

# Multiple Sequence Comparison and HMMs

Christian N. S. Pedersen\*

September 12, 2001

Pages 23-30 from BRICS DS-00-4 C. N. S. Pedersen: Algorithms in Computational Biology

## 1 Introduction

A sequence family is a set of homologous sequences. Members of a sequence family diverge during evolution and share similarities, but similarities that span the entire family might be weak compared to similarities that span only few members of the family. When comparing any two members of the family the faint similarities that span the entire family are thus likely to be shadowed by the stronger similarities between the particular two members. To detect similarities that span an entire sequence family it is therefore advisable to use other methods than just pairwise comparisons of the members.

Comparison of several sequences is a difficult problem that involves many modeling choices. The comparison of several sequences is typically communicated using a multiple alignment that expresses how the sequences relate by substitutions, insertions, and deletions. In this section we focus on methods to compute multiple alignments and ways to extract a compact characterization of a sequence family based on a comparison of its members. Such a characterization can be used to search for unknown members of the family, and for comparison against the characterizations of other families.

## 2 Multiple Alignment

A *multiple alignment* of a set of strings  $S_1, S_2, \dots, S_k$  over an alphabet  $\Sigma$  is a natural generalization of a pairwise alignment. A multiple alignment is a  $k \times \ell$  matrix  $\mathcal{A} = (a_{ij})$ , where the entries  $a_{ij}$ ,  $1 \leq i \leq k$  and  $1 \leq j \leq \ell$ , are either symbols from the alphabet or the blank symbol “-”, such that the concatenation of the non-blank characters in row  $i$  yields  $S_i$ . Figure 1 shows a multiple alignment of five strings. Computing a good multiple alignment of a set of strings is a difficult and much researched problem. Firstly, it involves choosing

---

\*Basic Research In Computer Science (BRICS), Department of Computer Science, University of Aarhus, Ny Munkegade, 8000 Århus C, Denmark. E-mail: [cstorm@daimi.au.dk](mailto:cstorm@daimi.au.dk).

$$\begin{bmatrix} A & A & G & A & A & - & A \\ A & T & - & A & A & T & G \\ C & T & G & - & G & - & G \\ C & C & - & A & G & T & T \\ C & C & G & - & G & - & - \end{bmatrix}$$

Figure 1: A multiple alignment of five strings.

a score function that assigns a score to each possible multiple alignment describing its quality with respect to some criteria. Secondly, it involves constructing a method to compute a multiple alignment with optimal score.

The *sum-of-pairs* score function introduced by Carillo and Lipman in [4] defines the score of a multiple alignment of  $k$  strings as the sum of the scores of the  $k(k-1)/2$  pairwise alignments induced by the multiple alignment. It is difficult to give a reasonable biological justification of the sum-of-pairs score function but nonetheless it has been widely used, e.g. [2, 11, 20]. If a classical score function with linear gap cost is used to compute the score of the induced pairwise alignments, then an optimal sum-of-pairs multiple alignment of  $k$  strings of lengths at most  $n$  can be computed in time  $O(2^k \cdot n^k)$  and space  $O(n^k)$  by a generalization of the dynamic programming method for computing an optimal pairwise alignment. Despite the simplicity of this multiple alignment method, its steep running time and space consumption makes it impractical even for modestly sized sets of relatively short strings.

Wang and Jiang in [24] show that the problem of computing a multiple alignment with optimal sum-of-pairs score is NP hard. However, the need for good multiple alignments has motivated several heuristics and approximation algorithms for computing a multiple alignment with a good sum-of-pairs score. For example, Feng and Doolittle in [10] present a heuristic based on combining good pairwise alignments. Combining good pairwise alignments is also the general idea of the approximation algorithm presented by Bafna *et al.* in [3] which in polynomial time computes a multiple alignment of  $k$  strings with a sum-of-pairs score that for any fixed  $l < k$  is at most a factor  $2 - l/k$  from the optimal score. The approximation algorithm is a generalization of ideas presented by Gusfield in [15] and Pevzner in [21].

Many score functions other than sum-of-pairs, and corresponding methods for computing an optimal multiple alignment, have been proposed in the literature. For example, to construct a multiple alignment of biological sequences it seems natural to take the evolutionary relationships between the sequences into account. Hein in [16] presents a heuristic which simultaneously attempts to infer and use the evolutionary relationships between members of a sequence family to guide the construction of a multiple alignment of the members of the sequence family. Krogh *et al.* in [19] present a popular and successful heuristic for computing multiple alignments, which use profile hidden Markov models to describe the relationships between members of a sequence family. We return to

profile hidden Markov models in Section 3.

A multiple alignment of a set of strings is useful for many purposes. The relationships between strings expressed by a multiple alignment is used to guide many methods that attempt to infer knowledge such as evolutionary history, or common three-dimensional structure, from a set of biological sequences. On the other hand, knowledge about the evolutionary history, or the common three-dimensional structure, of a set of biological sequences can also be used to produce a good multiple alignment. As mentioned above, the method by Hein in [16] attempts to incorporate the correspondence between evolutionary history and multiple alignments into a single method for constructing a multiple alignment while reconstructing the evolutionary history.

In the rest of this section we will not focus on any specific application of multiple alignments, but instead focus on the problem of deriving a compact characterization of a set of strings from a multiple alignment of its members. If the set of strings is a biological sequence family, such a compact characterization has at least two interesting applications. Firstly, it can be used to search a sequence database for unknown members of the family. Secondly, it can be used to compare sequence families by comparing their characterizations rather than comparing the individual members of the families.

The *consensus string* of a set of strings  $S_1, S_2, \dots, S_k$  is a string that attempts to capture the essence of the entire set of strings. There is no consensus on defining a consensus string, but if a multiple alignment of the set of strings is available it seems natural to use the relationships expressed by the multiple alignment to construct the consensus string. Most often this is done by extracting the dominant character from each column in the multiple alignment. In the simplest case the dominant character is chosen as the most frequent occurring character, where ties are broken arbitrarily. Because blanks are not part of the alphabet of the strings, it is common to ignore columns where the dominant character is a blank. This implies that each position in the consensus string corresponds to a column in the multiple alignment, but that columns in the multiple alignment where the dominant character is blank do not correspond to positions in the consensus string. Using this definition a possible consensus string of the multiple alignment in Figure 1 is the string CTGAGG, where the sixth column does not correspond to a position in the consensus string because the most frequent character in this column is a blank.

If the aligned set of strings is a biological sequence family, then the extracted consensus string is usually referred to as the *consensus sequence* of the family. It is natural to interpret the consensus sequence of a family as a possible ancestral sequence from which each sequence in the family has evolved by substitutions, insertions, and deletions of characters. Each row in the multiple alignment of the family then describes how the corresponding sequence has evolved from the consensus sequence; a character in a column that corresponds to a position in the consensus sequence has evolved from that position in the consensus sequence by a substitution; a blank in a column that corresponds to a position in the consensus sequence indicates that the character in that position in the consensus sequence has been deleted; a character in a column that does not correspond to a position in the consensus sequence has been inserted.

	1	2	3	4	5	6	7
A	0.4	0.2		0.6	0.4		0.2
C	0.6	0.4					
G			0.6		0.6		0.4
T		0.4				0.4	0.2
-			0.4	0.4		0.6	0.2

Figure 2: The profile of a multiple alignment in Figure 1. The entries of the profile that are not filled are zero. An entry for a character in a column of the profile is the frequency with which that character appears in the corresponding column in the multiple alignment.

The consensus sequence is a very compact characterization of a multiple aligned sequence family. The simplicity of the consensus sequence characterization is attractive because it makes it possible to compare sequence families by their consensus sequences using any method for sequence comparison. However, the consensus sequence characterization of a multiple aligned sequence family is probably too coarse-grained because it abstracts away all information in the multiple alignment except for the dominant character in each column. The *profile* of a multiple alignment as introduced by Gribskov *et al.* in [14, 13] is a more fine-grained method to characterize a set of strings from a multiple alignment of its members that attempts to remedy this problem.

A profile of a multiple alignment describes for each column the frequency with which each character in the alphabet (and the blank character) appears in the column. Figure 2 shows the profile of the multiple alignment in Figure 1. Gribskov *et al.* in [14, 13] show how to compare a profile and a string in order to determine how likely it is that the string is a member of the set of strings characterized by the profile. The general idea of the method is similar to alignment of two strings. The profile is viewed as a “string” where each column is a “character”. The objective is to compute an optimal alignment of the string and the profile where the score reflects how well the string fits the profile. This is done by using a position dependent scoring scheme that defines the cost of matching a character from the string against a column in the profile as the sum of the costs of matching the character to each character in alphabet weighted with the frequency with which the character appears in the column of the profile. For example, the cost of matching character  $G$  to the second column of the profile in Figure 2 is  $0.2 \cdot d(A, G) + 0.4 \cdot d(C, G) + 0.4 \cdot d(T, G)$ , where  $d(x, y)$  is the cost of matching character  $x$  with character  $y$ . An optimal alignment of a string of length  $n$  and a profile of  $m$  columns can be computed in time  $O(|\Sigma|nm)$ , where  $|\Sigma|$  is the size of the alphabet of the string and profile. Gotoh in [12] shows how to compare two profiles which makes it possible to compare sequence families by their profile characterizations. The general idea of the method is once again similar to alignment of two strings.

### 3 Hidden Markov Models

One of the most popular and successful way to characterize sequence families is to use profile hidden Markov models, which are simple types of hidden Markov models. A *hidden Markov model*  $M$  over an alphabet  $\Sigma$  describes a probability distribution  $P_M$  over the set of finite strings  $S \in \Sigma^*$ , that is,  $P_M(S)$  is the probability of the string  $S \in \Sigma^*$  under the model  $M$ . A hidden Markov model  $M$  can be used to characterize a family of strings by saying that a string  $S$  is a member of the family if the probability  $P_M(S)$  is significant.

Similar to a Markov model, a hidden Markov model consists of a set of states connected by transitions. Each state has a local probability distribution, the *state transition probabilities*, over the transitions from that state. We use  $P_q(q')$  to denote the probability of a transition from state  $q$  to  $q'$ . The transition structure of a hidden Markov model is can be illustrated as a directed graph with a node for each state, and an edge between two nodes if the corresponding state transition probability is non-zero. Unlike a Markov model, a state in a hidden Markov model can generate a character according to a local probability distribution, the *symbol emission probabilities*, over the characters in the alphabet. We use  $P_q(\sigma)$  to denote the probability of generating character  $\sigma \in \Sigma$  in state  $q$ . A state that does not have symbol emission probabilities is a *silent state*.

It is convenient to think of a hidden Markov model as a generative model in which a *run* generates a string  $S \in \Sigma^*$  with probability  $P_M(S)$ . A run starts in a special start-state, and continues from state to state according to the state transition probabilities, until a special end-state is reached. Each time a non-silent state is entered, a character is generated according to the symbol emission probabilities of that state. A run thus follows a Markovian sequence of states and generates a sequence of characters. The name “*hidden Markov model*” is because the Markovian sequences of states, the path, is hidden while only the generated sequence of characters, the string, is observable.

The basic theory of hidden Markov models was developed and applied to problems in speech recognition in the late 1960’s and early 1970’s. Rabiner in [22] gives a very good overview of the theory of hidden Markov models and its applications to problems in speech recognition. Hidden Markov models were first applied to problems in computational biology in the late 1980’s and early 1990’s. Since then they have found many applications, e.g. modeling of DNA [5], protein secondary structure prediction [1], gene prediction [18], and recognition of transmembrane proteins [23]. Probably the most popular application, introduced by Krogh *et al.* in [19], is to use *profile hidden Markov models* to characterize a sequence family by modeling how the sequences relate by substitutions, insertions and deletions to the consensus sequence of the family. The prefix “profile” is because profile hidden Markov models address the same problem as profiles of multiple alignments.

A profile hidden Markov model is characterized by its simple transition structure. Figure 3 shows the transition structure of a small profile hidden Markov model. The transition structure consists of repeated elements of *match*, *insert*, and silent *delete* states. The number of repeated elements is the *length* of the model. Each element of a match, insert and delete state models a position

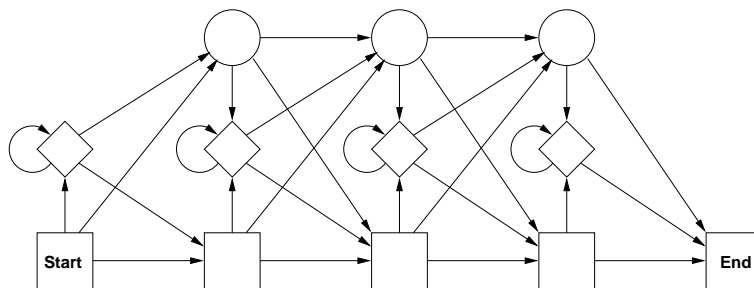


Figure 3: The transition structure of a profile hidden Markov model. The squares are the match-states, the diamonds are the insert-states and the circles are the silent delete-states.

in the consensus sequence of the sequence family, and describes how members of the family deviate from the consensus sequence at that position. The match state models that the generated character has evolved from the position in the consensus sequence. The insert state models that the generated character has been inserted between the two neighboring positions in the consensus sequence. The self-loop on the insert state models that several consecutive characters can be inserted between two positions in the consensus sequence. The delete state models that the position has been deleted from the consensus sequence.

The parameters of a profile hidden Markov model  $M$  (the length and the transition and emission probabilities) should be chosen to reflect the characteristics of the modeled sequence family (the length of the consensus sequence and how each member relates to the consensus sequence), such that the probability,  $P_M(S)$ , that it generates a string  $S$  can be used to distinguish between members and non-members of the family. The parameters can be chosen in consistence with an existing multiple alignment of members of the sequence family by setting the length of the model to the length of the consensus sequence of the multiple alignment, and by setting the transition and emission probabilities according to the frequency with which each character occurs in each column of the multiple alignment. This approach is similar to constructing a standard profile of a multiple alignment as discussed in the previous section. More interestingly, the parameters can also be estimated from an unaligned set of members of the sequence family. The estimation is done by setting the length of the model to the believed length of the consensus sequence, and by successively adjusting the transition and emission probabilities to maximize the probability of the model having generated the known members of the family.

Adjusting the parameters of a hidden Markov model  $M$  to maximize the probability,  $P_M(S)$ , that it generates a given string  $S$  is a fundamental and difficult problem. No exact method exists to decide, in general, the parameters that maximize  $P_M(S)$ . However, iterative methods that successively refine the parameters of the model, such that  $P_M(S)$  is guaranteed to converge to a local maximum, are available. Refining the parameters of a model  $M$  to maximize  $P_M(S)$  for a given string  $S$  is called *training* the model  $M$  with respect to  $S$ . Many training methods use the forward algorithm (which is described

below) to compute for any pair of state  $q$  and index  $i$ , the probability of being in state  $q$  having generated prefix  $S[1..i]$  of the string  $S$ . This set of probabilities is subsequently used to adjust the parameters of the model. Training methods are addressed in details by Rabiner in [22], and by Krogh *et al.* in [19].

## 4 Fundamental Algorithms for HMMs

Many applications of hidden Markov models in speech recognition and computational biology are variations of two fundamental problems and their solutions. The first problem is to determine the probability,  $P_M(S)$ , that a model,  $M$ , generates a given string  $S$ . The second problem is to determine the most likely path in a model  $M$  that generates a given string  $S$ . Because the ideas behind the algorithms solving these problems are fundamental to many applications of hidden Markov models, we present the algorithms in further details below. In the presentation we assume that the transition structure of a model contains no cycles of silent states. If the transition structure contains cycles of silent states, the presentation of the algorithms is more technical, but the asymptotic running times of the algorithms are unaffected by the presence of cycles of silent states. The full details are described in [22, 6].

The first problem, i.e. computing the probability  $P_M(S)$  that model  $M$  generates string  $S$ , is solved by the *forward algorithm*. The general idea of the forward algorithm is to build a table,  $A$ , indexed by states from  $M$  and indices from  $S$ , such that entry  $A(q, i)$  holds the probability of being in state  $q$  in  $M$  having generated the prefix  $S[1..i]$  of  $S$ . The entry indexed by the end-state and the length of  $S$  then holds the desired probability  $P_M(S)$  of being in the end-state having generate  $S$ . To explain the algorithm we call state  $q'$  a *predecessor* of state  $q$  in  $M$ , if the transition probability from  $q'$  to  $q$ ,  $P_{q'}(q)$ , is non-zero. The probability of being in state  $q$  having generated  $S[1..i]$  is then the sum over all predecessors  $q'$  of  $q$  of the probability of coming to state  $q$  via predecessor  $q'$  having generated  $S[1..i]$ . There are two cases. If  $q$  is a non-silent state, the last character of  $S[1..i]$  is generated in  $q$ . In this case  $A(q, i)$  is the sum over all predecessors  $q'$  of  $q$  of terms  $A(q', i - 1) \cdot P_{q'}(q) \cdot P_q(S[i])$ . If  $q$  is a silent state, no character is generated in  $q$ . In this case  $A(q, i)$  is the sum over all predecessors  $q'$  of  $q$  of terms  $A(q', i) \cdot P_{q'}(q)$ . In summary we get:

$$A(q, i) = \begin{cases} \sum_{q' \rightarrow q} A(q', i - 1) \cdot P_{q'}(q) \cdot P_q(S[i]) & \text{if } q \text{ is non-silent} \\ \sum_{q' \rightarrow q} A(q', i) \cdot P_{q'}(q) & \text{if } q \text{ is silent} \end{cases} \quad (1)$$

By using dynamic programming this recurrence yields an algorithm for computing  $P_M(S)$  with running time and space consumption  $O(mn)$ , where  $m$  is the number of transitions in  $M$ , and  $n$  is the length of  $S$ .

The second problem, i.e. computing the probability of the most likely path in model  $M$  that generates string  $S$ , and the path itself, is solved by the *Viterbi algorithm*. The only difference between the Viterbi algorithm and the forward

algorithm is that entry  $A(q, i)$  holds the probability of the most likely path to state  $q$  that generates  $S[1..i]$ . This probability is the maximum, instead of the sum, over all predecessors  $q'$  of  $q$  of the probability of coming to state  $q$  via predecessor  $q'$  having generated  $S[1..i]$ . The entry indexed by the end-state and the length of  $S$  holds the probability of the most likely path in  $M$  that generates  $S$ . The most likely path can be obtained by backtracking the performed maximization steps. The running time and space consumption of the Viterbi algorithm, including backtracking, is the same as the running time and space consumption of the forward algorithm.

The forward and the Viterbi algorithms are both useful when applied to profile hidden Markov models. The probability that a model generates a given string, which is computed by the forward algorithm, and the probability of the most likely path that generates a given string, which is computed by the Viterbi algorithm, are both plausible measures of how likely it is that a string is a member of the sequence family modeled by the profile hidden Markov model. Both measures can be used to search a sequence database for new members of the family, or inversely, to search a database of profile hidden Markov models (families) for the model (family) that most likely describe a new sequence.

Another application of the Viterbi algorithm is to construct a multiple alignment. The idea is to interpret the most likely path in a profile hidden Markov model  $M$  that generates the string  $S$  as an alignment of  $S$  against the consensus sequence of sequence family modeled by  $M$ . The interpretation is as follows; if the most like path passes the  $k$ th match state, such that  $S[i]$  is most likely generated by the  $k$ th match state, then  $S[i]$  should be matched against the  $k$ th character in the consensus sequence; if the most like path passes the  $k$ th insert state, such that  $S[i]$  is most likely generated by the  $k$ th insert state, then  $S[i]$  should be inserted between the  $k$ th and  $(k + 1)$ st character in the consensus sequence; finally, if the most likely path passes the  $k$ th delete state, then the  $k$ th character in the consensus sequence has been deleted from  $S$ . The most likely path thus explains how to construct  $S$  by matching, inserting and deleting characters from the consensus sequence, i.e. describes an alignment of  $S$  against the consensus sequence. Alignments of several sequences against the consensus sequence can be combined to a multiple alignment of the sequences.

Constructing multiple alignments using the Viterbi algorithm, combined with training methods to construct profile hidden Markov models from unaligned set of sequences, is one of the most popular and successful heuristics for multiple sequence alignment. It was also the primary application of profile hidden Markov models by Krogh *et al.* in [19]. Other applications of profile hidden Markov models are described by Durbin *et al.* in [6], and Eddy in [8, 9]. Software packages, e.g. SAM [17] and HMMER [7], that implement methods using profile hidden Markov models are widely used, and libraries, e.g. Pfam (available at <http://pfam.wustl.edu>), that store multiple alignments and profile hidden Markov models characterizations of sequences families are available and growing. In July 1999, the Pfam library contained multiple alignments, and profile hidden Markov models characterizations, of 1488 protein families. (In June 2000, it contained 2290 protein families characterization.)

## References

- [1] K. Asai, S. Hayamizu, and K. Handa. Prediction of protein secondary structure by the hidden markov model. *Computer Applications in the Biosciences (CABIOS)*, 9:141–146, 1993.
- [2] D. Bacon and W. Anderson. Multiple sequence alignment. *Journal of Molecular Biology*, 191:153–161, 1986.
- [3] V. Bafna, E. L. Lawler, and P. A. Pevzner. Approximation algorithms for multiple sequence alignment. In *Proceedings of the 5th Annual Symposium on Combinatorial Pattern Matching (CPM)*, Lecture Notes in Computer Science, pages 43–53, 1994.
- [4] H. Carrillo and D. Lipman. The multiple sequence alignment problem in biology. *SIAM Journal on Applied Mathematics*, 48:1073–1082, 1988.
- [5] G. A. Churchill. Stochastic models for heterogeneous DNA sequences. *Bulletin of Mathematical Biology*, 51:79–94, 1989.
- [6] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.
- [7] S. Eddy. *HMMER User's Guide*. Department of Genetics, Washington University School of Medicine, 2.1.1 edition, December 1998. (See <http://hmmmer.wustl.edu>).
- [8] S. R. Eddy. Hidden markov models. *Current Opinion in Structural Biology*, 6:361–365, 1996.
- [9] S. R. Eddy. Profile hidden markov models. *Bioinformatics*, 14:755–763, 1998.
- [10] D. Feng and R. Doolittle. Progressive sequence alignment as prerequisite to correct phylogenetic trees. *Journal of Molecular Evolution*, 25:351–360, 1987.
- [11] O. Gotoh. Alignment of three biological sequences with an efficient traceback. *Journal of Theoretical Biology*, 121:327–337, 1986.
- [12] O. Gotoh. Optimal alignment between groups of sequences and its application to multiple sequence alignment. *Computer Applications in the Biosciences (CABIOS)*, 9:361–370, 1993.
- [13] M. Gribskov, R. Lüthy, and D. Eisenberg. Profile analysis. *Methods in Enzymology*, 183:146–159, 1990.
- [14] M. Gribskov, A. D. McLachlan, and D. Eisenberg. Profile analysis: Detection of distantly related proteins. In *Proceedings of the National Academy of Science of the USA*, volume 84, pages 4355–4358, 1987.

- [15] D. Gusfield. Efficient methods for multiple sequence alignment with guaranteed error bounds. *Bulletin of Mathematical Biology*, 55:141–154, 1993.
- [16] J. Hein. Unified approach to alignment and phylogenies. *Methods in Enzymology*, 183:626–645, 1990.
- [17] R. Hughey and A. Krogh. SAM : Sequence alignment and modeling software system. Technical Report UCSC-CRL-95-7, University of California, Santa Cruz, January 1995. (The SAM documentation is regularly updated. See <http://www.cse.ucsc.edu/research/compbio/sam.html>).
- [18] A. Krogh. Two methods for improving performance of an HMM and their application for gene finding. In *Proceedings of the 5th International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 179–186, 1997.
- [19] A. Krogh, M. Brown, I. S. Mian, K. Sjölander, and D. Haussler. Hidden markov models in computational biology: Applications to protein modeling. *Journal of Molecular Biology*, 235:1501–1531, 1994.
- [20] M. Murate, J. Richardson, and J. Sussman. Simultaneous comparison of three protein sequences. *Proceedings of the National Academy of Science of the USA*, 82:3073–3077, 1985.
- [21] P. Pevzner. Multiple alignment, communication cost, and graph matching. *SIAM Journal of Applied Mathematics*, 52:1763–1779, 1992.
- [22] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77, pages 257–286, 1989.
- [23] E. L. L. Sonnhammer, G. von Heijne, and A. Krogh. A hidden Markov model for predicting transmembrane helices in protein sequences. In *Proceedings of the 6th International Conference on Intelligent Systems for Molecular Biology (ISMB)*, 1998.
- [24] L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1:337–348, 1994.