MUSCA: An Algorithm for Constrained Alignment of Multiple Data Sequences

Laxmi Parida parida@us.ibm.com
Aris Floratos floratos@us.ibm.com
Isidore Rigoutsos rigoutso@us.ibm.com

Bioinformatics and Pattern Discovery Group
Computational Biology Center
IBM Thomas J. Watson Research Center
Yorktown Heights, NY10598, USA

Abstract

Given a set of N sequences, the Multiple Sequence Alignment problem is to align these N sequences, possibly with gaps, that brings out the best *commonality* of the N sequences. MUSCA[1] is a two-stage approach to the alignment problem by identifying two relatively simpler sub-problems whose solutions are used to obtain the alignment of the sequences. We first *discover motifs* in the N sequences and then extract an appropriate subset of compatible motifs to obtain a "good" alignment. The motifs of interest to us are the *irredundant* motifs which are only polynomial in the input size. In practice, however, the number is much smaller (sub-linear). Notice that this step aids in a direct N-wise alignment, as opposed to composing the alignments from lower order (say pairwise) alignments and the solution is also independent of the order of the input sequences; hence the algorithm works very well while dealing with a large number of sequences. The second part of the problem that deals with obtaining a good alignment is solved using a graph-theoretic approach that computes an induced subgraph satisfying certain simple constraints. We reduce a version of this problem to that of solving an instance of a set covering problem, thus offer the best possible approximate solution to the problem (provided P$\neq$NP). Our experimental results, while being preliminary, indicate that this approach is efficient, particularly on large numbers of long sequences, and, gives good alignments when tested on biological data such as DNA and protein sequences. We introduce the the notion of an *alignment number* K ($2 \leq K \leq N$), a user-controlled parameter, that lends a useful flexibility to the aligning program: this additional requirement constrains the alignment to have at least K sequences agree on a character, whenever possible, in the alignment. The usefulness of the alignment number is corroborated by the users who view this as a natural constraint while dealing with a large number of sequences.

1 Introduction

Given a set of N sequences, the Multiple Sequence Alignment problem is to align these N sequences, possibly with gaps, that brings out the best *commonality* of the N sequences. Various alignment cost functions [2, 3, 4, 6, 8, 7, 14, 15, 12, 9], have been used in literature. The general approach to solving the pairwise ($N = 2$) sequence alignment problem has been a dynamic programming technique using different mechanisms of scores which is a function of the *edit distance*, along with *gap penalties*, to evaluate the similarity of the sequences. In [16, 13] the case of $N > 2$ has been handled by first doing a pairwise alignment for some or all possible pairs in some order and then building a N-wise alignment from these.

MUSCA[2] uses a two-stage approach to the alignment problem by identifying two relatively simpler sub-problems which deal separately with the two issues, one of identifying the "local similarities" and

[1] Musca is a constellation in the polar region of the Southern Hemisphere near Apus and Carina. Also, MUSCA is an anagram of the salient characters in Constrained Multiple Sequence Alignment.

[2] Musca is a constellation in the polar region of the Southern Hemisphere near Apus and Carina. Also, MUSCA is an anagram of the salient characters in Constrained Multiple Sequence Alignment.

the other of aligning the similarities appropriately. We first *discover motifs* in the N sequences, and then use these motifs to obtain a "good" alignment. Informally, a motif is a repeated pattern that appears more than once in a sequence. In the alignment context a motif is a pattern that appears in two or more input sequences A major point of criticism regarding using motifs is that they are usually very large in number (exponential in input size); however, we show that the motifs that are relevant to the alignment problem are the *irredundant motifs*, and the number of these motifs is polynomial in the input size [10]. Moreover, in practice, this number is much smaller (sub-linear). Thus, using motifs for the alignment helps in at least two ways: (1) it aids in a direct N-wise alignment, as opposed to composing the alignments from lower order (say pairwise) alignments and (2) the solution is independent of the order of the input sequences. We believe that, in practice, these have important consequences.

The second sub-problem of the alignment problem is that of obtaining a good alignment. Notice that any arbitrary set of motifs need not necessarily give rise to an alignment, under the premise that the alignment that uses a motif *does not introduce gaps in the motif*. Having obtained *all possible* motifs in the first stage, this stage involves pruning this set to obtain a (sub)set that gives an alignment. We solve this problem by mapping the motifs of the first stage to a suitable directed graph. Next we show that obtaining an alignment of the motifs is equivalent to solving a set-covering problem. Thus we present a very systematic way of aligning sequences based on motifs.

It is well known that the multiple sequence alignment problem, in addition to being a hard-to-solve problem, is also very hard to *model* to the satisfaction of evolutionary biologists, geneticists and other users. Does our approach have any theoretical contributions to the multiple sequence alignment problem in general? We introduce the the notion of an *alignment number* K ($2 \leq K \leq N$), a user-controlled parameter, that lends a useful flexibility to the aligning program: this additional requirement constrains the alignment to have at least K sequences agree on a character, whenever possible, in the alignment. This is particularly of interest when a large number of sequences are being aligned. The utility of the alignment number is corroborated by the users who view this as a natural constraint while dealing with a large number of sequences.

Roadmap. We describe our two-stage approach of motif discovery in Section 2 and aligning sequences in Section 3. We discuss the issues involved in using motifs for alignment and present a simple graph theoretic formulation in Section 3.1. We present heuristic algorithm for this problem by mapping it to a set covering problem in Section 3.2.

2 Motif Discovery (Stage 1)

We begin by giving a rigorous definition of a motif.

Definition 1 *(K-motif m, location list $\mathcal{L}_m$) Given a string s on alphabet Σ and an integer K, $2 \leq K < |s|$, a string m on $\Sigma \cup$ '.' is a K-motif with location list $\mathcal{L}_m = (l_1, l_2, \ldots, l_p)$, if all of the following hold:*

1. $m[0], m[|m|-1] \in \Sigma$,
   [First and last characters of the motif are *solid* characters; if "dont care" (the '.' character) characters are allowed at the ends, the motifs can be made arbitrarily long in size without conveying any extra information.]

2. $p \geq K$,

3. *there does not exist a location l, $l \neq l_i$, $1 \leq i \leq p$ such that m occurs at l on s (the location list is of maximal size), and,*
   [This ensures that any two distinct location lists must have distinct motifs associated with each.]

4. *for every "dont care" character at position j in m, there exist at least two distinct occurrences l_{i_1} and l_{i_2}, $1 \leq i_1, i_2 \leq p$, such that $s[l_{i_1} + j] \neq s[l_{i_2} + j]$.*
   [No "dont care" character can be replaced by a solid character, otherwise an arbitrary number of "dont care" characters could be introduced without conveying any extra information.]

If m is a string of $\Sigma \cup \{`.'\}$, m is called a *rigid motif*. In the rest of the discussion a K-motif is referred to as a motif.

We give the the definitions of maximality and irredundancy below.

Definition 2 *(Maximal Motif) Let p_1, p_2, ..., p_k be all the motifs in the sequence s. A motif p_i is maximal if and only if there is no p_j $(j \neq i)$ such that p_i is a substring of p_j, or, if p_i is a substring of p_j, then there exists at least one occurrence of p_i in s that is not covered by p_j in s.*

Definition 3 *(Redundant motif) A maximal motif m, with location list $\mathcal{L}_m$, is redundant if there exist maximal motifs m_i, $1 \leq i \leq p$, such that $\mathcal{L}_m = \mathcal{L}_{m_1} \cup \mathcal{L}_{m_2} \ldots \cup \mathcal{L}_{m_p}$.*
A motif that is not redundant is called an irredundant *motif.*

The motifs of interest to the sequence alignment problem are the *irredundant* motifs (see lemma 4) which are only quadratic in the input size and there exists a polynomial time algorithm to extract them from the input [10].

Similar definitions extend to discovering common motifs from multiple sequences. *Teiresias* is an efficient implementation of the motif discovery problem [11] and we use this in our experiments. Notice that a strip is a restricted version of the motif, in the sense that it is a motif of size one. In the following discussion we discuss motifs while bearing in mind that the results would hold also for strips (the results that are not critically dependent on the motif size being greater than one).

3 Sequence Alignment

We skip the process of discovering motifs from input sequences. Similar definitions extend to discovering common motifs from multiple sequences. TEIRESIAS is an efficient implementation of the motif discovery problem [11] and we use this in our experiments.

We obtain the irredundant motifs and the position of the motif in each sequence it appears in. The offset list, associated with each motif p_i, is a list of two-tuples (s_i, l_i), where s_i is a pointer to the sequence and l_i is the offset in the sequence. We assume that every offset list has exactly one occurrence of each sequence. At the end of the motif discovery step we have motifs, $p_1, p_2, \ldots, p_N$; N is the number of the motifs (not necessarily distinct), each with an offset list consisting of at least $k \geq 2$ (distinct) sequences and the position in the sequence where the motif appears.

Can *all* the motifs be used in an alignment? If the answer is yes, we form an alignment that respects all the motifs. If the answer is no, we remove the "offending" motifs in a manner that optimizes a cost function and gives an alignment. To investigate this further, we now explore the conditions under which *two* motifs can be used simultaneously in an alignment.

Definition 4 *(Motif Overlap) Two irredundant motifs p_i and p_j overlap if there exists a sequence s containing both these motifs and the following holds. Let n_i and n_j be the sizes of the motifs p_i and p_j and let l_i and l_j be the locations (offsets) in a sequence s respectively, then the motifs overlap if the intervals $[l_i, l_i + n_i]$ and $[l_j, l_j + n_j]$ have a non-empty intersection.*

Definition 5 *(Pairwise Compatible Motifs) Two motifs, p_1 and p_2, are pairwise compatible if there exists an alignment of the sequences that does not introduce gaps in the motifs p_1 and p_2.*

Lemma 1 *Two irredundant motifs p_i and p_j are pairwise compatible if and only if none of the following holds:*

1. *If p_i and p_j do not overlap in* all *the sequences, then p_i is to the left of p_j, without loss of generality (otherwise a* domain crossing mismatch *is said to have occurred).*

2. *If p_i and p_j overlap in* any *sequence, then p_i is at some fixed distance d to the left of p_j, without loss of generality (otherwise an* overlap mismatch *is said to have occurred).*

We define the alignment using a set of motifs as follows.

Definition 6 *(sequence alignment, compatible set) Given a set S of motifs, $v_{i_1}, v_{i_2}, \ldots, v_{i_n}$, a motif-alignment of the sequences, $s_1, s_2, \ldots, s_m$, is the alignment such that in all the sequences,* with no gaps in the motifs, *the motifs $v_{i_1}, v_{i_2}, \ldots, v_{i_n}$, are aligned (in all the sequences they appear). If such an alignment exists, the set $s_1, s_2, \ldots, s_m$, is called a compatible set.*

Definition 7 *(linear ordering of motifs) Given a set of compatible motifs, a consistent ordering of the motifs such that, in every sequence, the set of motifs that are present in the sequence appear in the left to right order and this ordering is called the linear ordering.*

Is it sufficient to just check for pairwise incompatibility of motifs while seeking an alignment?

Definition 8 *(domain crossing error) Given a set of motifs, $m_1, m_2, \ldots, m_n$, a domain crossing error is said to occur if there exists a linear ordering of the motifs $m_{i_1}, m_{i_2}, \ldots, m_{i_n}$, yet there exists no alignment that respects all the n motifs.*

Lemma 2 *A set of irredundant motifs $p_1, p_2, \ldots, p_n$ is feasible* if and only if *none of the following holds:*

1. *There exist distinct motifs p_i and p_j such that p_i and p_j are pairwise incompatible.*

2. *There exists a non-empty subset of the motifs* without *a linear ordering.*

3. *There exists a non-empty subset of the motifs that demonstrate* domain crossing *error.*

3.1 The Graph-theoretic Formulation

Next we wish to capture these conditions in a graph as follows. Construct a directed graph $G = (V, E)$ where every motif p_i corresponds to a vertex v_i, thus $N = |V|$. The directed edges are introduced as follows:

1. There is no edge between two vertices where the two corresponding motifs do not occur simultaneously in any sequence.

2. If p_i is to the left of p_j in *every* sequence that the two motifs are present, then a directed edge is placed from v_i to v_j. This is to indicate that in the alignment the motif p_i appears to the left of p_j. The edges are labeled as follows:

   (a) Label forbidden, if the motifs corresponding to v_1 and v_2 are not pairwise compatible.

   (b) Label overlap, if the motifs corresponding to v_1 and v_2 overlap.

   (c) Label nonoverlap, if the motifs corresponding to v_1 and v_2 are pairwise compatible but do not overlap.

The linear ordering of motifs is captured by checking for cycles in the graph. However the domain crossing mismatch requires a more careful handling as described below.

Handling domain crossing mismatches. We associate a *distance* $\mathcal{D}_{v_1v_2}$ with every edge that is *not* labeled forbidden. This is used to compute the feasibility of a collection of motifs corresponding to a solution; this *does not* contribute to the cost of the alignment. (We discuss the weight corresponding to the cost in the next section.) Let p_i and p_j be the two motifs corresponding to the vertices. Then if d is the *minimum* of the distance between the occurrences of the two motifs in every sequence that both of them appear in, $\mathcal{D}_{v_iv_j} = d$.

To detect the *domain crossing mismatches* of motifs (that are pairwise compatible), we define the notion of a *consistent graph w.r.t. a vertex.*

Definition 9 *Let $G = (V, E)$ be a labeled, weighted, directed, graph with weights on the edge uv given by $\mathcal{D}(u, v)$ and a label $\in \{forbidden, overlap, nonoverlap\}$. A path, $\mathcal{P}$, is valid if it has no edges labeled* forbidden. *Further, a valid path, $\mathcal{P}$, is called an overlap-path if all the edges in the path are labeled overlap. The weight of the valid path $\mathcal{P}$, $\mathcal{D}_{\mathcal{P}}$, is the sum of the weights of its constituent edges.*

Let $p \in V$. The graph is consistent w.r.t p if $\forall\ q \in V$, for all pairs of vertex-disjoint valid paths from p to q, $\mathcal{P}^1$ and $\mathcal{P}^2$,

1. *$\mathcal{D}_{\mathcal{P}^1} = \mathcal{D}_{\mathcal{P}^2}$, if $\mathcal{P}^1$ and $\mathcal{P}^2$ are both overlap-paths, or,*
2. *$\mathcal{D}_{\mathcal{P}^1} \geq \mathcal{D}_{\mathcal{P}^2}$, if $\mathcal{P}^1$ is an overlap-path and $\mathcal{P}^2$ is not.*

We now present the straightforward observation that relates the set of compatible motifs to a feasible subgraph.

Lemma 3 *The following two statements are equivalent:*

- *Given a subset of motifs $p_1, p_2, \ldots, p_n$ from the set of all motifs from m sequences of input, the subset is compatible, if the following holds:*
  1. *the motifs are not pairwise incompatible,*
  2. *there exists a linear ordering of $p_1, p_2, \ldots, p_n$, and,*
  3. *there is no domain crossing mismatch in $p_1, p_2, \ldots, p_n$.*
- *Given a subset of vertices $v_1, v_2, \ldots, v_n$, constructed as defined in this section. The induced subgraph on $v_1, v_2, \ldots, v_n$ is feasible, if the following holds:*
  1. *there is no edge labeled* forbidden *in the induced subgraph,*
  2. *the induced subgraph is acyclic, and,*
  3. *the induced subgraph is consistent w.r.t. every vertex v_i, $1 \leq i \leq n$.*

Lemma 4 *If p is a redundant motif, then using the motif p does not improve the cost of the optimization problem.*

Proof Sketch. Let p be rendered redundant by motifs $p_1, p_2, \ldots, p_n$, $n \geq 1$. By definition, motif p has less number of solid-characters than each of p_i, $1 \leq i \leq n$. Thus if an alignment can use motif p, it can certainly use all the motifs $p_1, p_2, \ldots, p_n$, giving a larger number of solid-characters; thus a higher cost for the optimization problem. □

3.2 Algorithm to compute the "best" alignment

Given a set of incompatible motifs, the set can be grouped into sets (not necessarily disjoint) such that each set violates *exactly one* of the three conditions of Lemma (2). These sets are called *basic incompatible sets*. Next, it can be easily shown that we can remove *exactly one* motif from a basic incompatible set to make it compatible.

The algorithm proceeds in the following three steps.

1. Detect the *basic infeasible* (sub)sets.

2. Eliminate motifs to obtain a feasible set that maximizes the cost.

3. Render the alignment.

Step 1. In this step we form subset of vertices that lead to incompatibility of the motifs. Using lemma (3), we compute the following sets:

1. Construct the sets F_1, F_2, ..., F_{n_f} where each set consists of two vertices which are the end points of an edge labeled Forbidden. This is done by simply scanning all the edges and collecting the end points of the edges labeled Forbidden.

2. Construct the sets C_1, C_2, ..., C_{n_c} where each set consists of vertices that form a directed cycle in the graph. All the cycles are captured by carrying out a depth first search (DFS) of the graph.

3. Construct the sets P_1, P_2, ..., P_{n_p} where each set consists of vertices that form a closed path in the graph. These are captured by carrying out a breadth first search (BFS) rooted at each vertex.

It is easy to see that the basic incompatible sets are $F_1, F_2 \ldots, F_{n_f}, C_1, C_2 \ldots, C_{n_c}, P_1, P_2 \ldots, P_{n_p}$.

Step 2. Set-covering problem. An instance $(X, \mathcal{Y})$ of the set-covering problem consists of a finite set X and a family $\mathcal{Y}$ of subsets of X, such that every element of X belongs to at least one subset in $\mathcal{Y}$: $X = \cup_{S \in \mathcal{Y}} S$. We say that a subset $S \in \mathcal{Y}$ *covers* its elements. The problem is to find a minimum-size subset $\mathcal{A} \subseteq \mathcal{Y}$ whose members cover all of X: $X = \cup_{S \in \mathcal{A}} S$. Any $\mathcal{A}$ satisfying this condition covers X. See [5] for details on this problem.

We construct an instance of a set cover problem (the dual of our problem) as follows. Let

$$\{v_1, v_2, \ldots, v_n\} = F_1 \cup F_2 \ldots \cup F_{n_f} \cup C_1 \cup C_2 \ldots \cup C_{n_c} \cup P_1 \cup P_2 \ldots \cup P_{n_p}$$

The elements of the the set (X of the set cover problem) are

$$F_1, F_2 \ldots, F_{n_f}, C_1, C_2 \ldots, C_{n_c}, P_1, P_2 \ldots, P_{n_p}.$$

The subset S_i corresponds to each v_i, $1 \leq i \leq n$ (where $\mathcal{Y} = \{S_1, S_2 \ldots, S_n\}$), and is defined as

$$S_i = \{F_l | v_i \in F_l, 1 \leq l \leq n_f\} \cup \{C_l | v_i \in C_l, 1 \leq l \leq n_c\} \cup \{P_l | v_i \in P_l, 1 \leq l \leq n_p\}.$$

Thus S_i denotes all the basic incompatible set that has a common element v_i, and removing the motif corresponding to v_i suffices to make all the corresponding basic sets compatible. Now, it is easy to see that a solution to the set-covering problem gives the minimum number of motifs that need to be removed so that the remaining set of motifs is compatible.

Associating weights to S_i which reflect the weight of each motif (depending on the cost function), gives a weighted set cover problem. The set-covering problem is known to be MAX SNP hard and the greedy algorithm is the best known approximation algorithm for the problem, assuming $P \neq NP$ [1]. To reflect the underlying cost function a weight is associated with every motif in the following manner. Let c be the number of solid characters in the corresponding motif, p_i, and let the number of sequences containing p_i be l, then for the k-MSA problem the associated weight is cl and for k-MSA$_{\max}$ it is c. We ignore the change in cost due to the common cost of a set of motifs [3] .

Step 3. This step consists of the following substeps:

[3] For a more accurate computation of the cost an appropriate common cost due to a set of overlapping motifs must be used.

3.1 We compute a linear ordering (see Definition 7) using the graph $\mathcal{G}'$ of Step 2, of the motifs p^i_{j1}, p^i_{j2}, …, $p^i_{j_i}$ for each sequence i. Such a linear ordering of the motifs exists, since the set of motifs is feasible.

3.2 From the original sequence s_i, we obtain the fillers (if any) between two consecutive motifs as f^i_0, f^i_{j1}, f^i_{j2}, …, $f^i_{j_i}$. f^i_0 is the leftmost portion, possibly empty. For example, let sequence $s_i = abcdefghijkl$ and let two motifs be as follows: $p^i_1 = cde$ and $p^i_2 = ghi$. Then $f^i_0 = ab$, $f^i_1 = f$, $f^i_2 = jkl$.

3.3 We obtain an alignment of the sequences by appropriately aligning each $(p^i_{jl} + f^i_{jl})$ and f^i_0, $l = 1, 2, \ldots, j_i$, filling the gaps with '-'. The symbol '+' denotes a string concatenation operation. The alignment is such that each motif of a sequence is perfectly aligned with the corresponding motif in all the other sequences.

For example let $s_1 = abcdefghijkl$ and $s_2 = cdexyzpqrghitu$. Then $p^1_1 = p^2_1 = cde$ and $p^1_2 = p^2_2 = ghi$ and $f^1_0 = ab$, $f^1_1 = f$, $f^1_2 = jkl$, f^2_0 =empty, $f^2_1 = xyzpqr$, $f^2_2 = tu$. Then the alignment of the sequences are as follows (the motifs are shown in bold):

(1)	a	b	**c**	**d**	**e**	f	–	–	–	–	–	**g**	**h**	**i**	j	k	l
(2)			**c**	**d**	**e**	x	y	z	p	q	r	**g**	**h**	**i**	t	u	

4 Summary

We have proposed a two-stage approach to the alignment problem by handling two relatively simpler sub-problems which deal separately with the two issues, one of identifying the "local similarities" and the other of aligning the similarities appropriately. In the first stage we identify *all possible* K-wise motifs, i.e., all motifs that appear simultaneously in at least K of the N input sequences ($2 \leq K \leq N$). In the second stage, we give plausible alignments of a carefully chosen subset of these motifs (that optimize certain cost functions). Using this approach for the alignment helps in at least two ways: (1) it aids in a direct N-wise alignment, as opposed to composing the alignments from lower order (say pairwise) alignments and (2) the resulting alignment is independent of the order of the input sequences. K is an input parameter and is called the *alignment number*. In practice, our approach works particularly well for alignment of a large set of (long) sequences. We have presented the result of running our alignment algorithm on biological data and the results look very promising. In the full version of the paper we discuss the computational complexity of the underlying optimization problem along with an analysis of the approximation factor of any alignment that results from the algorithm.

References

[1] Arora, S., Karger, D., Karpinski, M., Polynomial time approximation schemes for dense instances of NP-hard problems, *Proceedings of the Twenty-Seventh Annual ACM Symposium on the Theory of Computing*, ACM, 284–93, 1995.

[2] Altschul, S.F., Gap costs for multiple sequence alignment, *J. Theor. Biol.*, 138:297–309, 1989.

[3] Chao, K.M., Hardison, R., and Millter, W., Recent developments in linear-space alignment methods: a survey, *J. Comput. Biol.*, 3:271–291, 1994.

[4] Carrillo, H. and Lipman, D., The multiple sequence alignment problem in biology, *SIAM J. Appl. Math.*, 481073–1082, 1988.

[5] Cormen, T.H., Leiserson, C.E., and Rivest, R.L., *Introduction to Algorithms*, The MIT Press, Cambridge, Massachusetts, 1990.

[6] Gupta, S.K., Kececioglu, J.D., and Schaffer, A.A., Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment, *J. Comput. Biol.*, 2(3):459–472, 1995.

[7] Higgins, D.G. and Sharp, P.M., CLUSTAL: a package for performing multiple sequence alignment on a microcomputer, *Gene*, 73:237–244, 1988.

[8] Hirosawa, M., Totoki, Y., Hoshida, M., and Ishikawa, M., Comprehensive study on iterative algorithms of multiple sequence alignment, *Comput. Appl. Biosci.*, 11(1):13–18, 1995.

[9] Lawrence, C.E., Altschul, S.F., Boguski, M.S., Liu, J.S., Neuwald, A.F., and Wootton, J.C., Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment, *Science*, 262(5131):208–214, 1993.

[10] Parida, L., *Algorithmic Techniques in Computational Genomics*, Ph.D. thesis, Courant Institute of Mathematical Sciences, New York University, September, 1998.

[11] Rigoutsos, I. and Floratos, A., Motif discovery in biological sequences without alignment or enumeration, In *Proceedings of the Annual Conference on Computational Molecular Biology (RECOMB98)*, 221–227, ACM Press, 1998.

[12] Rocke, E. and Tompa, M., An algorithm for finding novel gapped motifs in DNA sequences, In *Proceedings of the Annual Conference on Computational Molecular Biology (RECOMB98)*, 228–233, ACM Press, 1998.

[13] Vihinen, M., An algorithm for simultaneous comparison of several sequences, *Comput. Appl. Biosci.*, 4(1):89–92, 1988.

[14] Waterman, M.S., Parametric and ensemble alignment algorithms, *Bull. Math. Biol.*, 56(4):743–767, 1994.

[15] Waterman, M.S., *An Introduction to Computational Biology: Maps, Sequences and Genomes*, Chapman Hall, 1995.

[16] Miller, W. and Zhang, Z., and He, B., Local multiple alignment vis subgraph enumeration, *Discrete Appl. Math.*, 71:337–365, 1996.