JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 1, Number 4, 1994
Mary Ann Liebert, Inc., Publishers
Pp. 337–348

On the Complexity of Multiple Sequence Alignment

LUSHENG WANG[1] and TAO JIANG[2]

ABSTRACT

We study the computational complexity of two popular problems in multiple sequence alignment: multiple alignment with SP-score and multiple tree alignment. It is shown that the first problem is NP-complete and the second is MAX SNP-hard. The complexity of tree alignment with a given phylogeny is also considered.

Key words: multiple sequence alignment, evolutionary tree, SP-score, computational complexity, approximation algorithm

INTRODUCTION

MULTIPLE SEQUENCE ALIGNMENT is one of the most important and challenging problems in computational biology (Lander *et al.*, 1991; Karp, 1993). It plays an essential role in two related areas of molecular biology: finding highly conserved subregions among a set of biological sequences, and inferring the evolutionary history of some species from their associated sequences. A huge number of papers have been written on effective and efficient methods for constructing multiple sequence alignment. For a comprehensive survey, see Waterman (1989) and Chan *et al.* (1992).

Many score schemes have been suggested to measure the quality of a multiple alignment. Among them, *SP-score* seems to be very sensible and has received a lot of attention (Baconn and Anderson, 1986; Carrillo and Lipman, 1988; Schuler *et al.*, in press). (Here, SP stands for *sum of all pairs.*) The best algorithm to compute an optimal alignment under SP measure is based on dynamic programming and requires a running time that is in the order of the product of the lengths of input strings (Altschul and Lipman, 1989). Gusfield first proposed a polynomial-time approximation algorithm for this problem that achieves ratio $2 - \frac{2}{k}$ on k input sequences (*i.e.*, the algorithm always produces an alignment whose value is at most $2 - \frac{2}{k}$ times the optimum) (Gusfield, 1991, 1993). Pevzner improved Gusfield's algorithm to obtain a ratio of $2 - \frac{3}{k}$ (Pevzner, 1992). Recently Bafna *et al.* pushed the ratio to $2 - \frac{1}{k}$ (Bafna *et al.*, 1993) for any fixed l. However, it was not known if multiple alignment with SP-score is NP-hard (Pevzner, 1992). Here we show that this problem (actually, the decision version of it) is NP-complete.

The construction of an optimal evolutionary tree from a given set of sequences can also be viewed as a type of multiple sequence alignment, called *multiple tree alignment* or, simply, *tree alignment*. Foulds and Graham proved that a variant of tree alignment, where the distance between two sequences is defined as Hamming

[1]Department of Electrical and Computer Engineering, and [2]Department of Computer Science, McMaster University, Hamilton, Ontario L8S 4K1, Canada.

distance, is NP-complete (Foulds and Graham, 1982). Recently, Sweedyk and Warnow proved that tree alignment is NP-complete (Sweedyk and Warnow, 1992). Several approximate methods have been proposed in the literature (Sankoff *et al.*, 1976; Sankoff and Cedergren, 1983; Hein, 1989a,b). Gusfield showed that a minimum-cost spanning tree of the input sequences has a cost that is at most twice the cost of an optimal evolutionary tree (Gusfield, 1991, 1993). An interesting question is whether one can find efficient algorithms with approximation ratio better that 2. It is easy to see that the recent results of Zelikovsky and Berman and Ramaiyer on approximation of Steiner minimal trees imply that tree alignment can be approximated within a factor of 1.747 in polynomial time (Berman and Ramaiyer, 1993; Zelikovsky, 1993). But can we make the approximation ratio arbitrarily close to 1? In this paper, we will answer this negatively by showing that tree alignment is MAX SNP-hard. The concept of MAX SNP-hardness was introduced by Papadimitriou and Yannakakis (1991) for the study of non-approximability of NP-complete problems. Combining with the result in Arora *et al.* (1992), our result implies that tree alignment does not have a polynomial-time approximation scheme (PTAS). (A problem has a PTAS if for every fixed $\varepsilon > 0$, it can be approximated with ratio $1 + \varepsilon$ in polynomial time.) In other words, the approximation ratio can not arbitrarily approach 1.

An important variant of three alignment is that we are not only given the sequences of some species, but also the phylogenetic structure (*i.e.*, the tree structure). Although the problem seems easier, the algorithms proposed in Sankoff and Cedergren (1983), Altschul and Lipman (1989), and Hein (1989b) all run in exponential time in the worst case and again it was not known if this problem is NP-hard. Among the many possible phylogenetic structures, binary tree and star are the most common ones (Sankoff and Cedergren, 1983; Altschul and Lipman (1989). We will prove that tree alignment with a given phylogeny is NP-complete, even when the phylogeny is a binary tree. Furthermore, the problem is MAX SNP-hard if the phylogeny is a star. In contrast, when the given phylogeny is a binary tree, tree alignment is *not* MAX SNP-hard, for we have shown in a companion paper that it has a PTAS (Wang *et al.*, 1993).

BASIC DEFINITIONS

A *sequence* is a string over some alphabet Σ. For DNA sequences, the alphabet Σ contains four letters A, C, G, and T representing four distinct nucleotides, and for protein sequences, Σ contains 20 letters, each representing a unique amino acid. An *alignment* of two sequences s_1 and s_2 is obtained by inserting spaces into or at either end of s_1 and s_2 such that the two resulting sequences s_1' and s_2' are of the same length. That is, every letter in s_1' is opposite to a unique letter in s_2'. A space is viewed as a new letter and is denoted Δ throughout this paper. Two opposing identical letters form a *match* and two opposing nonidentical letters form a *mismatch*, which can also be viewed as a replacement. A space in one sequence opposite to a letter x in the other can be viewed as a deletion of x from the second sequence, or an insertion of x into the first sequence.

Suppose that l is the length of the sequences s_1' and s_2'. The value of the alignment is defined as $\sum_{i=1}^{l} s(s_1'(i), s_2'(i))$, where $s_1'(i)$ and $s_2'(i)$ denote the two letters at the ith column of the alignment, and $s(s_1'(i), s_2'(i))$ denotes the *score* of the two opposing letters under some given *score scheme* s. There are several popular score schemes for amino acids and for nucleotides (Jukes and Cantor, 1969; Schwartz and Dayhoff, 1979). A standard assumption about a score scheme s is that it satisfies *triangle inequality*, *i.e.*, for any three letters x, y, and z, $s(x, z) \leq s(x, y) + s(y, z)$. An *optimal alignment* of two sequences is one that *minimizes* the value over all possible alignments. The *edit distance* between two sequences is defined as the minimum alignment value of the two sequences.

The concept of an alignment can be easily extended to more than two sequences. A *multiple alignment* $\mathcal{A}$ of $k \geq 2$ sequences is obtained as follows: spaces are inserted into each sequence so that the resulting sequences have the same length l, and the sequences are arrayed in k rows of l columns each. Again, a score value is defined on each column under some score scheme and the value of $\mathcal{A}$ is simply the sum of the scores of all columns. A very popular score scheme, called *SP-score*, defines the score value of a column as the sum of the scores of all (unordered) pairs of the letters in the column. That is, the value of the alignment $\mathcal{A}$ is the sum of the values of pairwise alignments induced by $\mathcal{A}$. The SP-score has previously been studied extensively in the past (see, *e.g.*, Baconn and Anderson, 1986; Carrillo and Lipman, 1988; Altschul and Lipman, 1989; Pevzner, 1992; Gusfield, 1993).

In the analysis of genetic evolution, we are given a set X of k sequences, each stands for an extant species. Let Y be a set of hypothetical sequences, where $Y \cap X = \emptyset$. (Usually each sequence in Y could represent an extinct species.) An *evolutionary tree* $T_{X,Y}$ for X is a *weighted* (sometimes *rooted*) tree of $|X \cup Y|$ nodes, where each node is associated with a *unique* sequence in $X \cup Y$ (Gusfield, 1991, 1993). The *cost* of an edge is the edit distance between the two sequences associated with the ends of the edge. The cost $c(T_{X,Y})$ of the tree $T_{X,Y}$ is the total cost of all edges in $T_{X,Y}$. Given sequences X, the *optimal evolutionary tree* or *multiple tree alignment* or, simply, *tree alignment* problem is to find a set of sequences Y as well as an evolutionary tree $T_{X,Y}$ for X which minimizes $c(T_{X,Y})$ over possible sets Y and trees $T_{X,Y}$. Sometimes one might require that the given sequences of the extant species be only associated with the leaves in the evolutionary tree (Farach *et al.*, 1993). It is easy to verify that our result in Theorem 5 still holds in this case.

An important variant of tree alignment is that we are not only given the sequences of some species, but also the phylogenetic structure (*i.e.*, the tree structure). More precisely, we are given a set X of k sequences and a tree structure with k leaves, each of them associated with a unique sequence in X. Then we would like to find the hypothetical sequences Y and assign them to the internal nodes of the given tree so that the total cost is minimized (Sankoff, 1975). We will refer to this problem as *tree alignment with a given phylogeny.*

NP-COMPLETENESS OF MULTIPLE ALIGNMENT WITH SP-SCORE

In this section, we prove that the following decision version of multiple sequence alignment with SP-score is NP-complete.

INSTANCE: Set of sequences $S = \{s_1, s_2, \dots, s_k\}$, and positive integer c.

QUESTION: Is there a multiple alignment of S with value c or less?

The reduction is from the *shortest common supersequence* problem (Garey and Johnson, 1979):

INSTANCE: Finite set S of sequences over alphabet Σ and positive integer m.

QUESTION: Is there a sequence s with $|s| \leq m$ such that each $t = t_1 t_2 \cdots t_r \in S$ is a subsequence of s, *i.e.*, $s = s_0 t_1 s_1 t_2 s_2 \cdots t_r s_r$, for some $s_0, s_1, \dots, s_r$?

The problem remains NP-complete even if $|\Sigma| = 2$ (Middendorf, in press).

Theorem 1 *Multiple sequence alignment with SP-score is NP-complete.*

Proof. Obviously, multiple sequence alignment is in NP. We reduce the shortest common supersequence problem to multiple alignment with SP-score. Given a set S of sequences over alphabet $\{0, 1\}$, and a positive integer m, we construct a collection of sets $X = \{X_{i,j} | i, j \geq 0, i + j = m\}$, where $X_{i,j} = S \cup \{a^i, b^j\}$ and a and b are two new letters. Here we can assume that each sequence in S has length at most m. The score scheme is shown in Table 1. Clearly the score scheme satisfies triangle inequality. The positive integer c is defined as $c = (k - 1)\|S\| + (2k + 1)m$, where $\|S\|$ is the total length of all sequences in S.

To show that multiple alignment with SP-score is NP-hard, it is sufficient to show that: S has a supersequence s of length m if and only if some $X_{i,j}$ has an alignment with value at most c.

(*if*) Suppose that we have an alignment $\mathcal{A}$ of the $k + 2$ sequences in $X_{i,j}$ with value at most c, for some i, j. Consider the induced alignment of the k sequences in S. No matter what the alignment is, its score is always $(k - 1)\|S\|$. Thus, in $\mathcal{A}$, the total contribution of the pairwise alignments involving sequences a^i and/or b^j, is at most $(2k + 1)m$. Therefore, every 0 must be aligned with an a and every 1 must aligned with a b in $\mathcal{A}$. We

TABLE 1. SCORE SCHEME I

S	0	1	a	b	Δ
0	2	2	1	2	1
1	2	2	2	1	1
a	1	2	0	2	1
b	2	1	2	0	1
Δ	1	1	1	1	0

can obtain a supersequence s for S by assigning 0 to the columns containing a's and 1 to the other columns. The length of s is $i + j = m$.

(*only if*) Let s be a supersequence for S with length m. Let i be the number of 0's in s and j the number of 1's in s. Consider set $X_{i,j}$. For each sequence $t \in S$, there exists an alignment of t and s such that each 0 (or 1) in X_i matches a 0 (or 1, respectively) in s. Some 0's and 1's in s may correspond to spaces. To obtain the desired multiple alignment, we align each t in S with s as above and then align the a's in the sequence a^i with the 0's in s and the b's in b^j with the 1's in s. Obviously, in this alignment, the letters in a column are either 0, a, Δ, or 1, b, Δ. The value of the alignment (with sequence s removed) is c.

Therefore, by checking the value of an optimal alignment of $X_{i,j}$, $i + j = m$, we can answer if there is a supersequence s for X with length m in polynomial time. ■

MAX SNP-HARDNESS OF TREE ALIGNMENT

In this section, we show that constructing an optimal tree alignment is MAX SNP-hard. This implies that there is no polynomial-time approximation scheme (PTAS) for the problem, unless P = NP, by the result of Arora *et al.* (1992).

First, we review the definition of *L-reduction* introduced by Papadimitriou and Yannakakis (1991). Suppose that Π and Π' are two minimization problems. (The definition is analogous for maximization problems.) We say that Π *linearly reduces* (L-reduces) to Π' if there are polynomial-time algorithms f and g and constants α, $\beta > 0$ such that, for any instance I of Π,

(1) $OPT(f(I))) \leq \alpha \cdot OPT(I)$

(2) Given any solution of $f(I)$ with cost c', algorithm g produces in polynomial time a solution of I with cost c satisfying $|c - OPT(I)| \leq \beta|c' - OPT(f(I))|$.

It follows from the above definition that (i) the composition of two L-reductions is an L-reduction and (ii) if problem Π L-reduces to problem Π' and Π' can be approximated in polynomial time within a factor of $1 + \varepsilon$, then Π can be approximated within factor $1 + \alpha\beta\varepsilon$. In particular, if Π' has a PTAS, so does Π.

To prove the MAX SNP-hardness of tree alignment, we first prove a sequence of auxiliary MAX SNP-hardness results. We begin with the Vertex Cover-B problem, which is proved to be MAX SNP-complete in Papadimitriou and Yannakakis (1991).

Vertex Cover-B: Given a graph $G = (V, E)$ with degree bounded by B, find the smallest vertex cover, *i.e.*, a smallest subset $V' \subseteq V$ such that, for each edge $(u, v) \in E$, at least one of u and v belong to V'.

We then L-reduce Vertex Cover-B to the following more restricted version of itself:

Triangle-free Vertex Cover-B: Given a triangle-free graph $G = (V, E)$ with degree bounded by B, find the smallest vertex cover.

Now we L-reduce Triangle-free Vertex Cover-B to a restricted version of tree alignment:

Restricted Tree Alignment: Given two sets of sequences X and Y, find a subset $Y' \subseteq Y$ and an evolutionary tree $T_{X,Y'}$ with the smallest cost.

Finally this problem is L-reduced to the tree alignment problem, stated again below:

Tree Alignment: Given a set of sequences X, find a set of sequences Y and an evolutionary tree $T_{X,Y}$ with the smallest cost.

Now, we describe the required reductions.

Lemma 2 *Triangle-free Vertex Cover-B is MAX SNP-hard.*

Proof. For each edge (v_i, v_j) in the given graph G, we insert two vertices $u_{i,j}$ and $u_{j,i}$ into the edge. This should remove all the triangles. Call this new graph G'. Clearly, G has a vertex cover of size c if and only if G' has a vertex cover of size $c + |e|$. This is an L-reduction because $|E| \leq B \cdot c$. ■

Lemma 3 *Restricted Tree Alignment is MAX SNP-hard.*

Proof. Let $G = (V, E)$ be a triangle-free graph with degree bounded by B, where $V = \{1, 2, \dots, n\}$. Without loss of generality, we also assume that G is connected. Let 0_i denote the binary sequence of length n with a 0 at the ith position and 1's at the rest, and $0_{i,j}$ denote the binary sequence of length n with 0's at the ith and jth positions and 1's at the rest. We construct sets $X = \{0_{i,j} \mid (i, j) \in E\}$ and $Y = \{1^n\} \cup \{0_i \mid i = 1, 2, \dots, n\}$. The score scheme is defined in Table 2, which also satisfies triangle inequality.

Seven types of edges may appear in a restricted evolutionary tree. Their costs are:

(1) $c(1^n, 0_i) = 1.$

(2) $c(1^n, 0_{i,j}) = 2.$

(3) $c(0_i, 0_j) = 2s(1, 0) = 2.$

(4) $c(0_i, 0_{k,l}) = s(1, 0) = 1, \text{ if } i = k \text{ or } i = l.$

(5) $c(0_i, 0_{k,l}) = 3s(1, 0) = 3, \text{ if } i \neq k \text{ and } i \neq l.$

(6) $c(0_{i,j}, 0_{k,l}) = 2s(1, 0) = 2, \text{ if } \{i, j\} \cap \{k, l\} \neq \emptyset.$

(7) $c(0_{i,j}, 0_{k,l}) = 4s(1, 0) = 4, \text{ if } \{i, j\} \cap \{k, l\} = \emptyset.$

Now, we want to show that the reduction is indeed an L-reduction. Suppose that G has a vertex cover U of size c. We can connect each sequence $0_{i,j} \in X$ to some 0_k, where $k = i$ or j, and $k \in U$, and then connect the sequences $\{0_i \mid i \in U\}$ to 1^n. Each connection costs 1. This gives us a restricted evolutionary tree with cost $|E| + c$. Since the degree of G is bounded by B, $|E| \leq B \cdot c$. So condition (1) of L-reduction holds.

To see that condition (2) of L-reduction also holds, we need the following claim.

Claim 4 *Given a restricted tree alignment with cost c' we can find an evolutionary tree with cost not greater than c' in polynomial time such that all the edges are of type (1) or (4).*

Proof. Each edge $(1^n, 0_{i,j})$ can be replaced by two edges $(1^n, 0_i)$ and $(0_i, 0_{i,j})$. An edge $(0_i, 0_j)$ can be replaced by two edges $(0_i, 1^n)$ and $(1^n, 0_j)$. An edge $(0_i, 0_{k,l})$ of type (5), where $i \neq k$ and $i \neq l$, can be replaced by the edges $(0_k, 0_{k,l})$ and $(0_k, 0_i)$ with the same cost 3. An edge $(0_{i,j}, 0_{k,l})$ of type (6), where $\{i, j\} \cap \{k, l\} = \{m\}$, can be replaced by the edges $(0_{i,j}, 0_m)$ and $(0_{k,l}, 0_m)$ with the same cost 2. An edge $(0_{i,j}, 0_{k,l})$ of type (7), where $\{i, j\} \cap \{k, l\} = \emptyset$, can be replaced by the edges $(0_{i,j}, 0_i)$, $(0_{k,l}, 0_k)$, $(1^n, 0_i)$, and $(1^n, 0_k)$ with the same cost 4. ■

Given an evolutionary tree with cost c', we can construct a new evolutionary tree with the same cost c' using edges of types (1) and (4) only. The number of sequences of form 0_i in the new evolutionary tree is at most $c' - |E|$. This implies a vertex cover of G of size at most $c' - |E| + 1$. Therefore, setting $\beta = 1$ makes condition (2) hold. This completes the proof. ■

Theorem 5 *Tree Alignment is MAX SNP-hard.*

Proof. By Lemma 3, it suffices to show that given an evolutionary tree T for X with cost c, where X is the same as in the proof of Lemma 3, there is a polynomial-time algorithm to construct a restricted evolutionary tree for X and $Y = \{1^n\} \cup \{0_i \mid 1 \leq i \leq n\}$, with cost c or less. Observe that here X has the "triangle-free" property, *i.e.*, X does not simultaneously contain the sequences $0_{i,j}$, $0_{i,k}$, and $0_{j,k}$ for any i, j, k. We will give a method to modify the tree T so that every sequence not in X is of form 1^n or 0_i.

TABLE 2. SCORE SCHEME II

	0	1	Δ
0	0	1	2
1	1	0	2
Δ	2	2	0

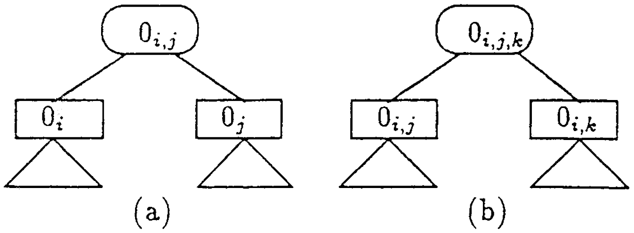

FIG. 1. a. Bad sequence of Type (a). b. Bad sequence of type (b).

A sequence that is not in X or of form 1^n or 0_i is a *bad sequence* and a node that is associated with a bad sequence is a *bad node*. Two sequences $0_{i,j}$ and $0_{k,l}$ are *adjacent* if $\{i, j\} \cap \{k, l\} \neq \emptyset$. Note that, as we have seen before, two adjacent sequences $0_{i,j}$ and $0_{i,k}$ can be connected through the sequence 0_i using two edges, each costing 1.

For convenience, we make without loss of generality a few assumptions about T. Here we view T as a rooted tree. We can assume that each edge in the tree T has cost 1. This is because we can delete any edge of cost 2 or more, find two adjacent sequences $0_{i,j}$ and $0_{i,k}$ one from each disconnected component, and reconnect them through 0_i. Since X is constructed from a connected graph G, such adjacent sequences always exist. Note that, this implies that all the sequences in the tree are of length n, because the score between Δ and other letters is 2. Moreover, we can assume that every bad node in T has two or more children. Otherwise, we can delete bad node and reconnect the two disconnected components as above without increasing the cost.

We will delete the bad nodes in T iteratively from the bottom to the top. Below we describe the steps involved in one iteration, which removes at least one bad node. Consider a bad node at the lowest level of the tree. The sequence, denoted s_1, associated with the node must be of form either $0_{i,j}$ (type (a)) or $0_{i,j,k}$ (type (b)), as shown in Fig. 1. Note that, in case (b), s_1 must have exactly two children due to the triangle-free property of X.

In the figure, an ellipse denotes a bad node, a rectangle denotes a *good* node, and a triangle denotes a subtree containing no bad nodes.

Suppose that s_1 is of type (a), say, $0_{i,j}$. Since s_1 has two children 0_i and 0_j, the parent of s_1 must have three 0's, say, $0_{i,j,k}$. Because each of 0_i and 0_j can appear at most once in T, s_1 must have a sibling of the form $0_{i,k}$, for some k, that has at most one child. That is, $0_{i,k}$ is not a bad node and $0_{i,k} \in X$ (see Fig. 2a). (In the figure, it is assumed that $0_{i,k}$ has one child.) Thus, we can delete s_1, move the subtree under 0_i to $0_{i,k}$ and relink the subtree under 0_j to the tree through some appropriate adjacent sequences (one from the subtree and one from the rest of the tree), as shown in Fig. 2b. This will not increase the cost.

Now suppose that s_1 is of type (b), say, $0_{i,j,k}$. Its parent, denoted s_2, contains either two 0's or four 0's. Suppose that s_2 contains two 0's, say, $s_2 = 0_{i,j}$. The assumptions made above force s_1 to have exactly two children $0_{i,k}$ and $0_{j,k}$. s_1 has a sibling of form 0_i or 0_j, then we can link $0_{i,k}$ (or $0_{j,k}$) to 0_i (or to 0_j, respectively) with cost 1, and thereby get rid of s_1. Thus, we assume that s_1 has a sibling s_3 with three 0's, say $0_{i,j,l}$, which

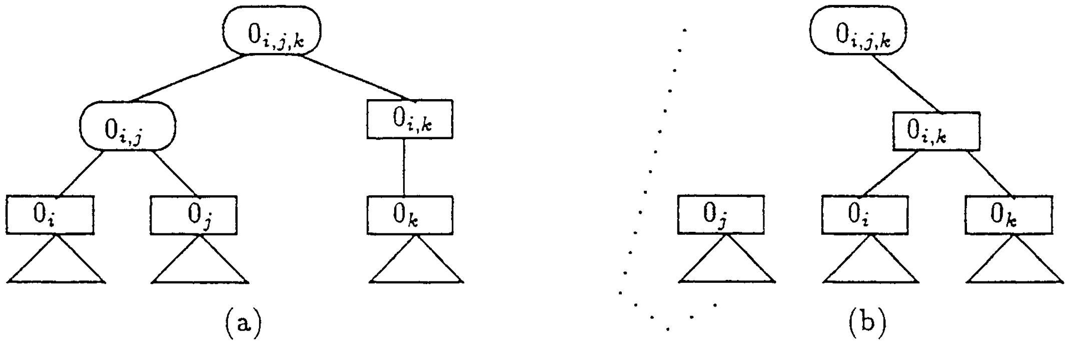

FIG. 2. When $s_1 = 0_{i,j}$.

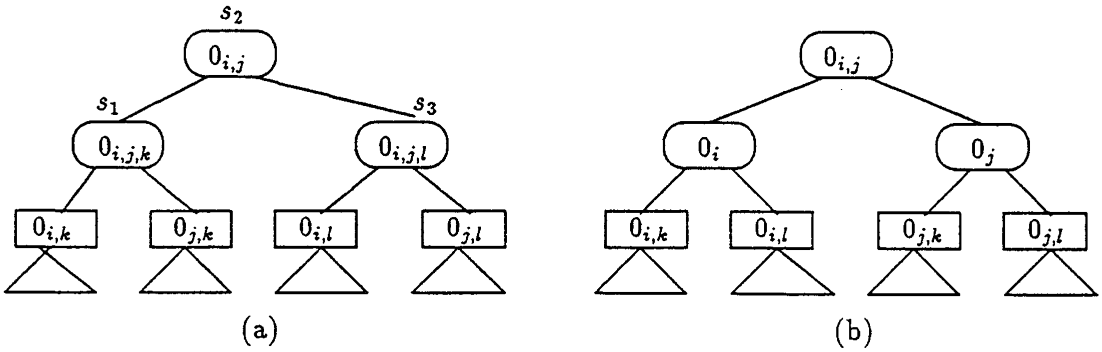

FIG. 3. When $s_1 = O_{i,j,k}$ and $s_2 = 0_{i,j}$.

is also a bad sequence of type (b). Similarly, s_3 must have two children $0_{i,l}$ and $0_{j,l}$. We can modify the two subtrees rooted at s_1 and s_3 to get rid of the bad nodes s_1 and s_3, as shown in Fig. 3.

Now suppose that s_2 contains four 0's, say, $s_2 = 0_{i,j,k,l}$. Without loss of generality, we assume that s_1 has a sibling $s_3 = 0_{i,j,l}$. We temporarily modify the subtrees under s_1 and s_2 as in Fig. 4.

Let s_4 be a sibling of s_2. Since the parent of s_2 has either three 0's or five 0's, s_4 has either four 0's, say, $0_{i,j,k,m}$, or two 0's, say, $0_{k,l}$. We consider two cases.

Case 1: $s_4 = 0_{i,j,k,m}$. Similar to $s_2 = 0_{i,j,k,l}$, s_4 has four descendants of form $0_{p,q}$, where $p, q \in \{i, j, k, l\}$. One of the descendants has to involve i, *e.g.*, it is of form $0_{i,k}$. Thus, we can link 0_i to $0_{i,k}$ with cost 1. This reconnects the component in Fig. 4b to the tree. Then we delete node s_2 and reduce the cost by 1.

Case 2: $s_4 = 0_{k,l}$. The component in Fig. 4 can be reorganized as in Fig. 5 without extra cost. Then we can link 0_k to s_4 with cost 1, and delete s_2 which reduces the cost by 1.

Therefore, we can gradually remove all the bad nodes from T. This completes the proof. ■

TREE ALIGNMENT WITH A GIVEN PHYLOGENY

In this section, we study the complexity of tree alignment when the phylogenetic tree structure is given. Two important structures are considered: binary tree and star.

Theorem 6 *It is NP-complete to construct an optimal evolutionary tree even when the given phylogeny is a binary tree.*

Proof. The reduction is again from the shortest common supersequence problem. Let $S = \{s_1, s_2, \ldots s_k\}$ be a set of sequences over $\{0, 1\}$ and m an integer. Now, we construct a binary tree T as in Fig. 6a, where

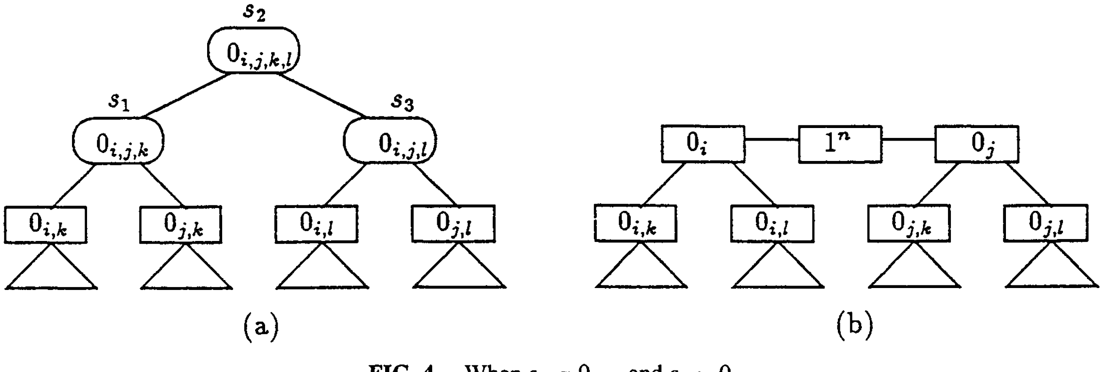

FIG. 4. When $s_1 = 0_{i,j,k}$ and $s_2 = 0_{i,j,k,l}$.

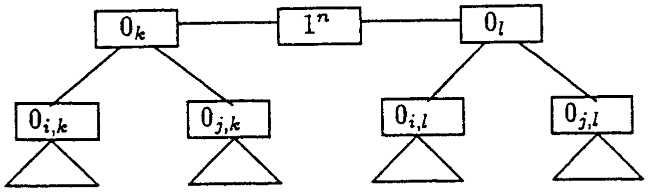

FIG. 5. Rearranging the component.

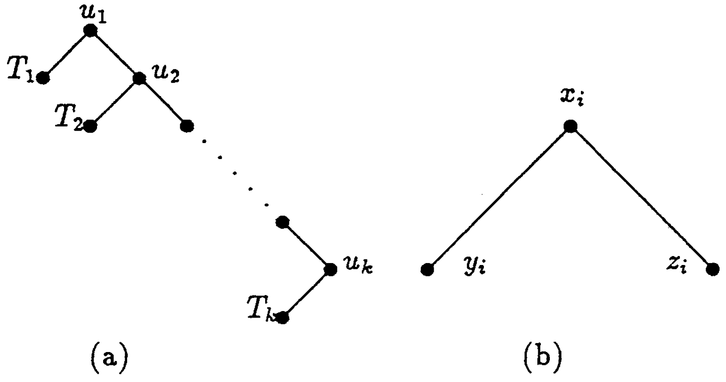

FIG. 6. a. The tree T. b. The subtree T_i.

each T_i is a subtree shown in Fig. 6b. Every leaf in T is associated with a sequence over $\Sigma = \{0, 1, a, b\}$. Let $l = \|S\| = \sum_{i=1}^{k} |s_i|$, and $D = 10^l 1$. D will be used as a delimiter in the construction. For each $i = 1, 2, \ldots, k$, let $w_i = D10^i$ if i is odd, $w_i = D0^i 1$ otherwise. We define $y_i = s_i w_i$ for $i = 1, 2, \ldots, k$, $z_i = a^{i-1} b^{m-i+1} w_i$, for $i = 1, 2, \ldots, k-1$, and $z_k = a^m w_k$.

The score scheme is defined in Table 3, which again satisfies triangle inequality.

Let $M = 2k - 1 + \sum_{i=1}^{k} 2m - |s_i|$. We will show that there is a common supersequence s with $|s| = m$ if and only if there is an evolutionary tree with cost M.

(*only if*) Assume that there is a supersequence s with $|s| = m$. To obtain the desired evolutionary tree, we assign sequence sw_i to every x_i and sequence $sD0^i$ to u_i, $i = 1, \ldots, k$.

(*if*) Suppose that there exists an assignment of sequences to the internal nodes of T such that the cost of the resulting evolutionary tree is M. For each i, let t_i be the sequence assigned to x_i. For any node v in T, let $T(v)$ denote the subtree rooted at v. First observe that, each edge in T costs at least 1 and because of the triangle inequality, the optimal cost of each $T(x_i)$ is at least the edit distance between the sequences at y_i and z_i, which is $2m - |s_i|$. Hence, the cost of $T(x_i)$ in this evolutionary tree is exactly $2m - |s_i|$, and t_i can be written as $t_i' D w_i$. Since now each edge of form (x_i, u_i) or (u_i, u_{i+1}) must cost 1, $t_1' = \ldots = t_k' = s$. The sequences b^m and a^m assigned to z_1 and z_k force s not to contain any a or b. Hence, in order for $T(x_i)$ to achieve score $2m - |s_i|$, the sequence s must be a supersequence of s_i and $|s| = m$. Therefore, we have a common supersequence for S with length m. ■

TABLE 3. SCORE SCHEME III

	0	1	a	b	Δ
0	0	1	1	1	1
1	1	0	1	1	1
a	1	1	0	2	2
b	1	1	2	0	2
Δ	1	1	2	2	0

TABLE 4. SCORE SCHEME

	#	\$	0	1	$*$	Δ	a	b
#	0	$2k$	$4k$	$4k$	$4k$	$4k$	$4k$	$4k$
\$	$2k$	0	$4k$	$4k$	$4k$	$4k$	$4k$	$4k$
0	$4k$	$4k$	0	1	0	1	1	1
1	$4k$	$4k$	1	0	0	1	1	1
$*$	$4k$	$4k$	0	0	0	0	0	2
Δ	$4k$	$4k$	1	1	0	0	0	2
a	$4k$	$4k$	1	1	0	0	0	2
b	$4k$	$4k$	1	1	2	2	2	2

Note that, in the above reduction all the internal nodes are labeled with distinct sequences. A simple modification of the proof shows that the NP-completeness result also holds if we relax the requirement and allow a sequence to appear several times in the evolutionary tree. (We just need remove everything after the delimiter D.)

To prove the MAX SNP-hardness of tree alignment when the given phylogeny is a star, we need the degree-bounded version of Max Cut problem.

Max Cut-B: Given a graph $G = (V, E)$ with degree bounded by B, find a partition of V which divides V into disjoint sets V_0 and V_1 such that the number of edges that go from V_0 to V_1 is the largest.

Max Cut-B is shown to be MAX SNP-complete in Papadimitriou and Yannakakis (1991). Now, we can prove our last result.

Theorem 7 *It is MAX SNP-hard to construct an optimal evolutionary tree when the given phylogeny is a star.*

Proof. The reduction is from Max Cut-B. Let $G = (V, E)$ be a graph with degree bounded by constant k, where $E = \{v_1, v_2, \ldots, v_n\}$. Define $\Sigma = \{0, 1, a, b, \#, *, \$\}$. The letters #, \$ will serve as delimiters and $*$ will be a kind of "wild card." For each $v_1 \in V$, we construct a sequence $s_i = z_{i,1}z_{i,2}\cdots z_{i,n}\$$, where

$$z_{i,j} = \begin{cases} D'_j0*^k \ D_j1*^k & \text{if } j \neq i,\ v_i \text{ and } v_j \text{ are adjacent} \\ D'_j*^{k+1} \ D_j*^{k+1} & \text{if } j \neq i,\ v_i \text{ and } v_j \text{ are not adjacent} \\ D'_j1^{k+1} \ D_j0^{k+1} & \text{if } j = i, \end{cases}$$

where $D'_1 = \$$, $D_1 = \#$, and $D'_i = D_i = \#$ for $i = 2, \ldots, n$.

Observe that, in general s_i has the form

$$(\$x^{k+1}\#y^{k+1})(\#x^{k+1}\#y^{k+1})\cdots(\#x^{k+1}\#^{k+1})(\#x^{k+1}\#y^{k+1}\$).$$

It contains n blocks of x^{k+1} and n blocks y^{k+1} in the sequence, where the ith x^{k+1} is 0^{k+1}, ith y^{k+1} is 1^{k+1}, and the rest of x^{k+1}'s and y^{k+1}'s are either $*^{k+1}$, $0*^k$, or $1*^k$. Intuitively, the construction forces the internal sequence to have the form

$$*^{k+1}(\#x^{k+1}\#*^{k+1})(\#x^{k+1}\#*^{k+1})\cdots(\#x^{k+1}\#*^{k+1}),$$

where there are n blocks of x^{k+1} in the sequence, each of them is either 0^{k+1} or 1^{k+1}. The 0/1 choice of the x^{k+1} blocks represents a partition of G and the value of the optimal alignment of s_i, and the internal sequence is expected to give the number of edges incident on v_i that are not cut. Note that since the degree of each node in G is bounded by k, for each i the segments $\{z_{i,j} \mid j \neq i\}$ totally contain at most k 0's and k 1's. Hence, the segment $z_{i,i}$ dominates the alignment between s_i and the internal sequence (*i.e.*, it determines how the blocks are lined up) because it contains blocks 0^{k+1} and 1^{k+1}.

Similarly, let $t_i = u_{i,1}u_{i,2}\cdots u_{i,n}\$$, where

$$u_{i,j} = \begin{cases} D'_j*^{k+1} \ D_j*^{k+1} & \text{if } j \neq i \\ D'_j1^{k+1} \ D_j0^{k+1} & \text{if } j = i. \end{cases}$$

Finally, define

$$\begin{aligned} X_0 &= \{s_i | i = 1, 2, \dots, n\}, \\ X_1 &= \{a^i(\# *^{k+1} \#)^n | i = 0, 1, \dots, 5n(k+1)\}, \\ X_2 &= \{a^i(\# b^{k+1}\#)^n | i = 1, 2, \dots, 3k\}, \\ X_3 &= \{a^i t_j | i = 1, 2, \dots, k,\ j = 1, 2, \dots, n\} \end{aligned}$$

and

$$X - X_0 \cup X_1 \cup X_2 \cup X_3.$$

The score scheme is given in Table 4. Note that the scores do not satisfy triangle inequality. The phylogency is a star (*i.e.*, a tree with only one internal node) with $|X|$ leaves, each is associated with a sequence in X.

First, we show that the internal sequence in an optimal evolutionary tree for X should be in the form

$$*^{k+1}(\# x^{k+1}\# *^{k+1})(\# x^{k+1}\# *^{k+1}) \cdots (\# x^{k+1}\# *^{k+1}),$$

where there are n blocks of x^{k+1}, each is either 0^{k+1} or 1^{k+1}. This is due to the following reasons.

1. The sequences in $X_1 = \{a^i(\# *^{k+1} \#)^n | i = 0, 1, \dots, 5n(k+1)\}$ force the internal sequence to contain exactly n #'s and no \$. Otherwise, the $5n(k+1)$ sequences in X_1 contribute a cost of $5n(k+1) \cdot 2k = 10k(k+1)n$ or more. However, if the internal sequence is in the form $(\# 1^{k+1}\# *^{k+1})^n$, the total cost of the tree is less than $5k^2n + 6kn < 10k(k+1)n$. (See the analysis below.)

2. The sequences in $X_2 = \{a^i(\# b^{k+1}\#)^n | i = 1, 2, \dots, 3k\}$ force the internal sequence to contain none of 0, 1, b at positions between the $2i$th # and $(2i+1)$th #. Otherwise, the existence of such letter would make the $3k$ sequences in X_2 contribute an extra cost of $3k$, while the contribution from the sequences in X_0 decreases by at most $k+1$ and the contribution from the sequences in X_3 decreases by at most k. Thus, we can always delete such letters without increasing the total cost.

3. The score scheme allows us to delete an a from the internal sequence without increasing the cost.

4. Since $s(b, b) = s(b, \Delta) = 2$ and $s(b, 1) = s(b, 0) = 1$, it is advantageous for the internal sequence to be of form

$$*^{k+1}(\#\{0,1\}^{k+1}\# *^{k+1})(\#\{0,1\}^{k+1}\# *^{k+1}) \cdots (\#\{0,1\}^{k+1}\# *^{k+1}),$$

where $\{0, 1\}^{k+1}$ denotes any binary string of length $k+1$. The leading and trailing *'s are used to absorb the 0's and 1's in the beginning or end of an s_i or an t_i (see Fig. 7.)

5. The kn sequences in $X_3 = \{a^i t_j | i = 1, 2, \dots, k,\ j = 1, 2, \dots, n\}$ allow us to modify the internal sequence into the form

$$*^{k+1}(\# x^{k+1}\# *^{k+1})(\# x^{k+1}\# *^{k+1}) \cdots (\# x^{k+1}\# *^{k+1}),$$

where there are n blocks of x^{k+1} in the sequence, each of them is either 0^{k+1} or 1^{k+1}.

Now, we prove condition (1) of L-reduction. Suppose that there is a partition V_0, V_1) of V, which cuts c edges. The internal sequence can be constructed as

$$*^{k+1}(\# x^{k+1}\# *^{k+1})(\# x^{k+1}\# *^{k+1}) \cdots (\# x^{k+1}\# *^{k+1}),$$

where there are n blocks of x^{k+1}, the ith block is 1^{k+1} if v_i is in V_1, and 0^{k+1} otherwise. In this case, the sequences in X_1 contributes no cost. Each sequence in X_2 contributes a cost of $(k+1)n$ and thus X_2 totally contributes $3k(k+1)n$. Each sequence in X_3 contributes a cost of $2k$ and totally X_3 contributes $2k^2n$.

Let $c(v)$ denote the number of edges incident upon v that are cut by the partition. For each $v_i \in V$, s_i

```
  $0***#1***#1111#0000#0***#1***$   $0***#1***#1111#0000#0***#1***$     s_i
****#1111#****#1111#****#0000#****    ****#0000#****#0000#****#1111#****   int. seq.
               (a)                               (b)
```

FIG. 7. a. v_i is in V_1. b. v_i is in V_0.

contributes $2k + d(v_i) - c(v_i)$. This can be observed as follows. Since there are $2n$ #'s in the internal sequence and $2n - 1$ #'s and 2 \$'s in each s_i, the delimiters in s_i always contribute a cost of $6k$. For each i, if the ith block of x^{k+1} of the internal sequence is 1^{k+1} (*i.e.*, $v_i \in V_1$), we align s_i with the internal sequence as in Fig. 7a, *i.e.*, the right end delimiter of the s_i is matched with a space, and if the ith block of x^{k+1} of the internal sequences is 0^{k+1} (*i.e.*, $v_i \in V_0$), we align s_i with the internal sequence as in Fig. 7b. If $v_i \in V_1$, then for each v_j adjacent to v_i, the segment $z_{j,i}$ of s_j, which is of the form $D_i'0*^k\ D_i1*^k$, will contribute 1 towards the cost if and only if $v_j \in V_1$ (see Fig. 7a). Similarly, if $v_i \in V_0$, then for each v_n adjacent to v_i, the segment $z_{j,i}$ of s_j will contribute 1 towards the cost if and only if $v_j \in V_0$ (see Fig. 7b. That is, all the edges that are not cut by the partition are counted here.

Therefore, the total cost of the tree is

$$3k(k+1)n + 2k^2n + \sum_{i=1}^{n} 6k + d(v_i) - c(v_i) = 5k^2n + 9kn + 2|E| - 2c.$$

Note that, the maximum number of edges that can be cut by a partition of G is at least $0.5|E|$ (Papadimitriou and Yannakakis, 1991). Since the degree of G is bounded, condition (1) of L-reduction holds.

By the same argument, it is not hard to show that, given an evolutionary tree for X with cost $c' = 5k^2n + 5kn + 2|E| - 2c$, we can easily construct a partition of G which cuts c edges, by looking at the 0/1 assignment to the x-blocks in the internal sequence. Thus, condition (2) of L-reduction also holds (with $\beta = 1/2$). ■

CONCLUDING REMARKS

It remains an interesting open question if the score scheme in the above proof can be made to satisfy triangle inequality. If so, then the result in Wang et al. (1993) that tree alignment with a given binary phylogeny has a PTAS implies that the degree of the tree makes a difference in the approximability of the problem.

ACKNOWLEDGMENTS

We thank the referee for many valuable comments and suggestions. This project was partially supported by NSERC Research Grant OGP0046613.

REFERENCES

Altschul, S., and Lipman, D. 1989. Trees, stars, and multiple sequence alignment. *SIAM J. Applied Math.* 49, 197–209.

Arora, S., Lund, C., Motwani, R., Sudan, M., and Szegedy, M. 1992. On the intractability of approximation problems. *Proc. 33rd IEEE Symp. Found. Comp. Sci.* 14–23.

Baconn, D., and Anderson, W. 1986. Multiple sequence alignment. *J. Mol. Biol.* 191, 153–161.

Bafna, V., Lawler, E., and Pevzner, P. 1994. Approximate methods for multiple sequence alignment. *Proc. 5th Combinational Pattern Matching Conference*, 43–53.

Berman, P., and Ramaiyer, V. 1993. Improved approximations for the Steiner tree problem. *Algorithmica* (in press).

Carrillo, H., and Lipman, D. 1988. The multiple sequence alignment problem in biology. *SIAM J. Appl. Math.* 48, 1073–1082.

Chan, S.C., Wong, A.K.C., and Chiu, D.K.T. 1992. A survey of multiple sequence comparison methods. *Bull. Math. Biol.* 54, 563–598.

Farach, M., Kannan, S., and Warnow, T. 1993. A robust model for finding optimal evolutionary trees. *Proc. 25th ACM Symp. on Theory Computing*, 137–145.

Foulds, L.R., and Graham, R.L. 1982. The Steiner problem in phylogeny is NP-complete. *Adv. Appl. Math.* 3, 43–49.

Garey, M.R., and Johnson, D.S. 1979. *Computers and Intractability: A Guide to the Theory of NP-completeness*, W. H. Freeman, San Francisco, CA.

Gusfield, D. 1991. Efficient methods for multiple sequence alignment with guaranteed error bounds. Tech. Report, CSE-91-4, UC Davis.

Gusfield, D. 1993. Efficient methods for multiple sequence alignment with guaranteed error bounds. *Bull. Math. Biol.* 55, 141–154.

Hein, J.J. 1989a. A tree reconstruction method that is economical in the number of pairwise comparisons used. *Mol. Biol. Evol.* 6, 669–684.

Hein, J.J. 1989b. A new method that simultaneously aligns and reconstructs ancestral sequences for any number of homologous sequences, when the phylogeny is given. *Mol. Biol. Evol.* 6, 649–668.

Jukes, T.H., and Cantor, C.R. 1969. Evolution of protein molecules, 21–132. In Munro, H.N., ed., *Mammalian Protein Metabolism*, Academic Press, San Diego, CA.

Karp, R.M. 1993. Mapping the genome: Some combinatorial problems arising in molecular biology. *ACM STOC'93*, 278–285.

Lander, E.S., Langridge, R., and Saccocio, D.M. 1991. Mapping and interpreting biological information. *Commun. ACM* 34, 33–39.

Middendorf, M. More on the complexity of common superstring and supersequence problems. *Theoret. Comp. Sci.* (in press).

Papadimitriou, C.H., and Yannakakis, M. 1991. Optimization, approximation, and complexity classes. *J. Computer Syst. Sci.* 43, 425–440.

Pevzner, P. 1992. Multiple alignment, communication cost, and graph matching. *SIAM J. Appl. Math.* 56, 1763–1779.

Sankoff, D. 1975. Minimal mutation trees of sequences. *SIAM J. Appl. Math.* 28, 35–42.

Sankoff, D., Cedergren, R.J., and Lapalme, G. 1976. Frequency of insertion-deletion, transversion, and transition in the evolution of 5S ribosomal RNA. *J. Mol. Evol.* 7, 133–149.

Sankoff, D., and Cedergren, R. 1983. Simultaneous comparisons of three or more sequences related by a tree, 253–264. In Sankoff, D., and Kruskal, J., eds., *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, Addison Wesley, Reading, MA.

Schuler, G.D., Altschul, S.F., and Lipman, D.J. A workbench for multiple alignment construction and analysis, in *Proteins: Structure, Function and Genetics* (in press).

Schwarz, R., and Dayhoff, M. 1979. Matrices for detecting distant relationships, 353–358. In Dayhoff, M., ed., *Atlas of Protein Sequences*, National Biomedical Research Foundation, Washington, DC.

Sweedyk, E., and Warnow, T. 1992. The tree alignment problem is NP-hard.

Waterman, M.S., 1989. Sequence alignments, 53–92. In Waterman, M.S., ed. *Mathematical Methods for DNA Sequences*, CRC, Boca Raton, FL.

Wang, L., Jiang, T., and Lawler, E. 1993. Approximation algorithms for tree alignment with a given phylogeny. (submitted to Algorithmica)

Zelikovsky, A.Z. 1993. The 11/6 approximation algorithm for the Steiner problem on networks. *Algorithmica* 9, 463–470.

Address reprint requests to:
Dr. T. Jiang
Department of Computer Science and Systems
McMaster University
Hamilton, Ontario L8S 4K1 Canada
jiang@maccs.mcmaster.ca

Received for publication August 25, 1993; accepted as revised October 12, 1993.