Fixed Topology Alignment with Recombination

Bin Ma* Peking University

Lusheng Wang† City Univ. Hong Kong

Ming Li‡ Univ. Waterloo

Abstract

In this paper, we study a new version of multiple sequence alignment, *fixed topology alignment with recombination*. We show that it can not be approximated within any constant ratio unless $P = NP$. For a more restricted version, we show that the problem is MAX-SNP-hard. This implies that there is no PTAS for this version unless $P = NP$. We also propose approximation algorithms for a special case, where each internal node has at most one recombination child and any two *merge paths* for different recombination nodes do not share any common node.

Key words: approximation algorithm, computational biology, evolutionary tree, multiple sequence alignment, phylogeny, recombination.

1 Introduction

Multiple sequence alignment is the most critical cutting-edge tool for extracting and representing biologically important commonalities from a set of sequences. It plays an essential role in the solution of many problems such as searching for highly conserved subregions among a set of biological sequences and inferring the evolutionary history of a family of sequences [5, 24]. Many versions have been proposed [1, 5, 24]. *Tree alignment* is one of the most famous versions. It was first proposed by Sankoff in [14].

For tree alignment, we are given k sequences and a tree $Tree$ of k leaves, each of which is labeled with a unique given sequences. The goal is to construct a sequence for each internal node in $Tree$ such that the cost of the tree is minimized. The cost of an edge in a tree is defined as the edit distance between the two sequences assigned to the two ends of the edge. The cost of a tree is the total cost of edges in the tree. Once each internal node is assigned a sequence, one can produce a multiple sequence alignment by optimally aligning two sequences on every edge. An alternative definition of tree alignment is to find an optimal

*Current address: Department of Mathematics, Peking University, Beijing 100871, P.R. China

†Supported in part by UGC 9040297. Current address: Department of Computer Science, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon, Hong Kong. E-mail: lwang@cs.cityu.edu.hk

‡Supported in part by the NSERC Operating Grant OGP0046506 and CGAT. E-mail: mli@math.uwaterloo.ca Current address: Department of Computer Science, University of Waterloo, Waterloo, Ont. N2L 3G1, Canada.

multiple sequence alignment with *tree score*. Given a multiple sequence alignment $\mathcal{A}$, the score $\mu(s_1(i), s_2(i), \ldots, s_k(i))$ on the i-th column of $\mathcal{A}$, where $s_j(i)$ is the letter (possibly a space) from sequence s_j in the i-th column of $\mathcal{A}$, is defined as follows:

Let tree $Tree = (V, E)$, where V and E are the sets of nodes and edges in $Tree$, have k leaves. Let $k+1$, $k+2$, ..., $k+m$ be the internal nodes on T. For each internal node l, reconstruct a letter (possibly a space) $s_l(i)$ such that $\sum_{(p,q)\in E} \mu(s_p(i), s_q(i))$, where $\mu((s_p(i), s_q(i))$ is the score for the pair of letters $s_p(i)$ and $s_q(i)$, is minimized. The score $\mu(s_1(i), s_2(i), \ldots, s_k(i))$ of the i-th column is thus defined as:

$$\mu(s_1(i), s_2(i), \ldots, s_k(i) = \sum_{(p,q)\in E} \mu(s_p(i), s_q(i)). \tag{1}$$

The tree score for $\mathcal{A}$ is the total scores of all columns in $\mathcal{A}$. It is known that the two definitions of tree alignment are equivalent.

Tree alignment was proved to be NP-hard [20]. Many heuristic algorithms have been proposed in the literature [1, 6, 15, 16]. Some approximation algorithms with guaranteed relative error bounds have been reported recently. In particular, a polynomial time approximation scheme (PTAS) is presented in [21] and improved versions are given in [22, 23].

Multigene families, viruses, and alleles from within populations experience recombinations. When recombinations occur, the evolutionary history can not be represented as a tree. Hein first studied the method to reconstruct the history of sequences subject to recombination [7, 8]. Hein observed that the evolution of a sequence with k recombinations could be described by k recombination points and $k+1$ trees describing the evolution of the $k+1$ intervals, where two neighboring trees were either identical or differed by one subtree transfer operation [7, 8, 9, 3, 2]. A heuristic method was proposed to find the most parsimonious history of the sequences in terms of mutation and recombination operations. Another strike was given by Kececioglu and Gusfield [12]. They introduced two new problems, *recombination distance*, and *bottleneck recombination history*. They tried to include higher-order evolutionary events such as block insertions and deletions [4], and tandem repeats [11, 13].

In this paper, we propose a model called *fixed topology alignment with recombination* (FTAR for short) which is analogous to tree alignment. The difference is that in the given topology, some nodes (called *recombination nodes*) have two parents instead of one. A recombination node obtains its ancestral material from both parents. Moreover, the given topology may have more than one root. As for tree alignment, there are two ways to define the problem. One is to reconstruct a sequence for each internal node such that the total cost of the topology is minimized. Another is to construct a multiple sequence alignment with minimum score. The first version is called fixed topology history with recombination (FTHR) and the second version is called fixed topology alignment with recombination (FTAR). When recombination is considered, the two versions are no longer equivalent. We show that, in general, both versions do not have a constant ratio approximation algorithm. Even for a special case, where each internal node has at most one recombination child and there are at most 6 parents of recombination nodes in any path from a root to a leaf in the given topology, they do not has a PTAS. The above results show that the new problems are much harder than the normal tree alignment problem, in terms of approximation. However, since recombination rarely occur in practice, it is interesting to study special cases. We design a ratio-3 algorithm for both FTAR and FTHR in a more restricted case, where each internal node has at most one recombination child and any two merge paths for different recombination nodes do not share any common node. We then extend the algorithm into a PTAS for FTAR in the case, where each internal node has at most one recombination child, any two merge paths for different recombination nodes do not share any common node, and there is a constant number of crossovers for each recombination node.

2 Fixed topology alignment with recombination – the model

When recombination happens, the ancestral material on the present sequence s_1 is located on two sequences s_2 and s_3. s_2 and s_3 can be cut at k locations (break points) into $k+1$ pieces, where $s_2 = s_{2,1}s_{2,2}\ldots s_{2,l+1}$ and $s_3 = s_{3,1}s_{3,2}\ldots s_{3,l+1}$. s_1 can be represented as $s_{\hat{2},1}s_{\hat{3},2}s_{\hat{2},3}\ldots s_{\hat{2},i}s_{3,\hat{i}+1}\ldots$, where subsequences $s_{\hat{2},i}$ and $s_{3,\hat{i}+1}$ differ from the corresponding $s_{2,i}$ and $s_{3,i+1}$ by insertion, deletion, and substitution of letters. k, the number of times s_1 switches between s_2 and s_3, is called the number of *crossovers*. The cost of the recombination is

$$dist(s_{1,1}, s_{\hat{1},1}) + dist(s_{2,2}, s_{\hat{2},2}), \ldots dist(s_{1,i}, s_{\hat{1},i}) + dist(s_{2,i+1}, s_{2,\hat{i}+1}) + \ldots + k\chi,$$

where $dist(s_{2,i+1}, s_{2,\hat{i}+1})$ is the edit distance between the two sequences $s_{2,i+1}$ and $s_{2,\hat{i}+1}$, k is the number of crossovers and χ is the crossover penalty. The *recombination* distance to produce s_1 from s_2 and s_3 is the cost of a recombination that has the smallest cost among all possible recombinations. We use $r\_dist(s_1, s_2, s_3)$ to denote the recombination distance. For more details, see [12, 19].

When recombination occurs, the given topology is no longer a binary tree. Instead, some nodes, called *recombination nodes*, in the given topology may have two parents[7, 8]. In a more general case as described in [12], the topology may have more than one root. The set of roots is called a *protoset*. The edges incident to recombination nodes are called *recombination* edges. See Figure 5 (b). A node/edge is *normal* if it is not a recombination node/edge.

The cost on a pair of recombination edges is the recombination distance to produce the sequence on the recombination node from the two sequences on its parents. The cost on other normal edges is the edit distance between two sequences. A topology is *fully labeled* , if every node on the topology is labeled. For a fully labeled topology, the cost of the topology is the total cost of edges in the topology. Each node in the topology with degree greater than 1 is an internal node. Each leaf/terminal (degree 1 node) in the topology is labeled with a given sequence. The goal here is to construct a sequence for each internal node such that the cost of the topology is minimized. We call this problem *fixed topology history with recombination* (FTHB).

Obviously, this problem is a generalization of tree alignment. The difference is that the given topology is no longer a binary tree. Instead, there are some recombination nodes which have two parents instead of one. Moreover, there may be more than one root in the topology.

As for tree alignment, there is an alternative definition. Let the topology $T = (V, E)$, where V and E are the sets of nodes and edges in T, have k leaves (nodes without any child). Let $k+1$, $k+2$, ..., $k+m$ be the internal nodes on T. Let $E(q_1, \ldots, q_m)$ be the subset of edges in E such that each pair of recombination edges has exactly one edge in $E(q_1, \ldots, q_m)$ and the choices are indicated by q_i's (q_i indicates that either the left or right edge is selected). Given a multiple sequence alignment $\mathcal{A}$ of N columns, the score $\mu(s_1(i), s_2(i), \ldots, s_k(i), q_1, \ldots q_m)$ on the i-th column, where $s_j(i)$ is the letter (possibly a space) from sequence s_j in the i-th column, is defined as follows:

For each internal node l, reconstruct a letter (possibly a space) $s_l(i)$ such that

$$\sum_{(p,q)\in E(q_1,\ldots,q_m)} \mu(s_p(i), s_q(i)),$$

where $\mu((s_p(i), s_q(i))$ is the score for the pair of letters $s_p(i)$ and $s_q(i)$, is minimized. The score $\mu(s_1(i), s_2(i), \ldots, s_k(i), q_1, \ldots q_m)$ of the i-th column with the choice $(q_1, q_2, \ldots, q_m)$ is thus defined as:

$$\mu(s_1(i), s_2(i), \ldots, s_k(i), q_1, \ldots q_m) = \sum_{(p,q)\in E(q_1,\ldots,q_m)} \mu(s_p(i), s_q(i)). \tag{2}$$

The cost of $\mathcal{A}$ is defined as:

$$\min_{q_1(1),\ldots q_m(1), q_1(2), \ldots q_n(2), \ldots, q_1(N), \ldots, q_m(N)} \{\sum_{i+1}^{N} \{\mu(s_1(i), s_2(i), \ldots, s_k(i), q_1(i), \ldots q_m(i)) + f(i)\chi\}\}, \tag{3}$$

where $f(1) = 0$ and $f(i) = c$ if there are c $q_l(i)$'s that are different from $q_l(i-1)$'s. This version is called *fixed topology alignment with recombination* (FTAR).

Unlike tree alignment, the above two definitions are not equivalent. That is, the cost of an optimal solution for FTHR might be smaller than that for FTAR. For example, the topology and the sequences are given as in Figure 6 (a). If a mismatch costs 1 and the crossover penalty is $\chi = 0.5$, the sequences assigned to internal nodes should be as shown in Figure 6 (a). Thus, the cost of an optimal solution for FTHR is $3 + \chi$. However, an optimal solution for FTAR costs $4 + \chi$ as shown in Figure 6 (a).

The reason is that the alignment in Figure 6 (b) imposes that a crossover produces an intermediate sequence s4':AABBB. (See Figure 6 (c).) Moreover, the example demonstrates that a solution for FTHR might not lead to an alignment.

3 Hardness results

In this section, we show that FTHR and FTAR cannot be approximated within any constant performance ratio in general. We then consider a restricted version, where each internal node has at most one recombination child and there is at most 6 parents of recombination nodes in any path from the root to a leaf in the given topology. We show that the restricted version for both FTHR and FTAR do not have a PTAS.

Theorem 1 *FTHR and FTAR cannot be approximated within any constant performance ratio unless $P = NP$.*

Now, we consider a more restricted case, where each internal node has at most one recombination child and there are at most 6 parents of recombination nodes in any path from the root to a leaf in the given topology.

Theorem 2 *The restricted version for both FTHR and FTAR is MAX-SNP-hard. That is, there is no polynomial time approximation scheme unless $P = NP$.*

The reduction is from vertex cover. We will give the reduction in the full version of the paper.

The hardness results in this section show that the new problem is much harder than tree alignment in terms of approximation. However, recombination occur infrequently. So, it is interesting to study some restricted cases. A *merge node* of recombination node v is the lowest common ancestor of v's two parents. The two different paths from a recombination node to its merge node are called *merge paths.* In the rest part of the paper, we study the case, where

- (C1) each internal node has at most one recombination child and
- (C2) any two merge paths for different recombination nodes do not share any common node.

4 Ratio-3 approximation algorithms for FTHR and FTAR

We first consider FTHR. At the end of the section, we give the algorithm for FTAR with fixed number of crossovers. The basic idea of our approximation algorithms is to combine a method to deform the layout of the given topology with the uniform lifting method for tree alignment in [22].

Deforming the layout Given a topology T satisfies conditions (C1) and (C2), we extend some of the edges in T such that the two parents of a recombination node are at the same level. Figure 7 gives an example.

Uniform Lifting For a deformed layout of the topology, ignoring the recombination edges, treating the topology as a set of binary trees (denoted as *forest* F), and keeping the level positions of all binary trees in the layout, we can uniformly lift the sequences to internal nodes. The lifting choice for all the nodes (even in different binary trees) at the same level is uniform. Since we do not delete any node, each internal node gets a sequence. We thus get a fully labeled topology. Figure 7 (b) gives an example. A topology obtained in this way is called a *uniform lifted* topology.

We can show that there is a uniform lifted topology with cost at most 3 times of the optimum. First, let us consider some definitions.

Given a binary tree $Tree$, we arbitrarily extend the edges in $Tree$ to form an *extended layout* such that the two parents of every recombination node are at the same level. We then uniformly lift the sequences from leaves to the internal nodes in the layout. The fully labeled tree obtained in this way is called an *extended uniform lifted* tree.

Let $Tree^{min}$ be an optimal fully labeled tree for a binary tree $Tree$. Let d be the depth of the extended layout of $Tree$, there are 2^d extended uniform lifted trees $Tree(1)$, $Tree(2)$, ..., $Tree(2^d)$. Let $v(i)$ be the sequence lifted to v in the extended uniform lifted tree $Tree(i)$ for $i = 1, 2, \ldots, 2^d$. $v(\hat{i})$ is the sequence lifted to v's child which is different from $v(i)$ in $Tree(i)$. For each node v, $S(v)$ denotes the set of given sequences assigned to the descendent leaves of v. Let μ_v^s be the cost of the path in $Tree^{min}$ from the internal node v to its descendent leaf which is labeled with the given sequence $s \in S(v)$. Let $V(Tree)$ be the set of internal nodes (degree-3) in $Tree$.

Similar to the uniform lifted trees in [22], the total cost of the 2^d extended uniform lifted trees is bounded as follows:

Lemma 3

$$\begin{aligned}
\sum_{i=1}^{2^d} C(Tree(i)) &\le \sum_{i=1}^{2^d} \{2 \times \{ \sum_{v \in V(Tree)} \mu_v^{v(i)}\} + \mu_r^{r(i)} + \mu_r^{r(\hat{i})}\} \\
&= \sum_{i=1}^{2^d} \{2 \times \{ \sum_{v \in V(Tree)} \mu_v^{v(\hat{i})}\} + \mu_r^{r(i)} + \mu_r^{r(\hat{i})}\} \\
&= 2 \times 2^d C(Tree^{min}) - \sum_{i=1}^{2^d} \{\mu_r^{r(i)} + \mu_r^{r(\hat{i})}\} \qquad (4) \\
&= 2 \times 2^d C(Tree^{min}) - 2 \times \sum_{i=1}^{2^d} \mu_r^{r(i)}. \qquad (5)
\end{aligned}$$

Let T be a topology for tree alignment with recombination. T^{min} denotes an optimal fully labeled topology.

Theorem 4 *There is a uniform lifted topology with cost at most* 3 *times of the optimum.*

Now, let us focus on the computation of an optimal uniform lifted topology. Conditions (C1) and (C2) allow us to design a dynamic programming algorithm. Let v be a node in a merge path of a recombination node. The *parallel* node u of v is a node in the other merge path of the recombination node such that u and v are at the same level. We use $dist[s, s']$ to denote the edit distance between two sequences s_1 and s_2. We classify the internal nodes in T into five cases:

Case 1: Let v be a node such that (1) there is no descendent recombination node or (2) it reaches a merge node first before it reachs the corresponding recombination node if there is any descendent recombination node of v. Note that, v could be a normal node or a recombination node. Define $d[v, s]$ as the smallest cost of the sub-topology rooted at v such that sequence s is uniformly lifted to v.

$$d[v,s] = \min_{all\ possible\ s_2} d[v_1,s_1] + d[v_2,s] + dist(s_1,s), \tag{6}$$

where v_1 and v_2 are the two children of v, v_1 is assigned sequence s_1 and v_2 is assigned the sequence s. The choice of s_1 may not be unique. (See Figure 8 (a).)

Case2: Let v be the parent of a recombination node. We define $d[v,s]$ as the smallest cost of the sub-topology such that the two parents are the only nodes at the top level of the sub-topology and s is uniformly lifted to v. Let v' be the other parent of the recombination node and sequence s' be the lifted sequence on v'. $d[v,s]$ can be computed as follows:

$$d[v,s] = \min_{all\ possible\ s'\ and\ s_3} d[v_1,s] + d[v_2,s'] + d[v_3,s_3] + r\_dist(s,s',s_3), \tag{7}$$

where v_3 is the recombination node, v_1 is the normal child of v, v_2 is the normal child of v', and s_3 is uniformly lifted to v_3. Note that, the sequences s and s' are also lifted to v_1 and v_2, respectively. (See Figure 8 (b).)

Case 3: Let v be a normal node between the merge node and the parent of the recombination node such that v has a parallel node v'. Define $d[v,s]$ as the smallest cost of the sub-topology with nodes v and v' at the top and containing all the descendent nodes of v and v'. (See Figure 8 (c).) $d[v,s]$ can be computed as follows:

$$\begin{aligned} d[v,s] \quad = & \min_{all\ possible\ s,\ s_1,s_2,s_3\ and\ s_4} dist(s,s_1) + dist(s,s_2) + dist(s',s_3) + dist(s',s_4) \\ & + d[v_1,s_1] + d[v_2,s_2] + d[v_3,s_3], \end{aligned} \tag{8}$$

where v_1 and v_2 are the two children of v that are assigned s_1 and s_2, respectively; v_3 and v_4 are the two children of v' that are assigned s_3 and s_4, respectively; and v' is assigned the sequence s'. Note that s is either s_1 or s_2 and s' is either s_3 or s_4.

Case 4: Let v be the node in a merge path of a recombination node such that v does not have a parallel node. Corresponding to v, there is an extended edge in the other merge path of the recombination node. Let v' be the node at the lower end of the extended edge. (See Figure 8 (d).) Define $d[v,s]$ as the smallest cost of the sub-topology rooted at nodes v and v' and containing all their descendent nodes.

$$d[v,s] = \min_{all\ possible\ s_1\ ands_2} d[v_1,s_1] + d[v_2,s_2] + dist(s_1,s_2), \tag{9}$$

where v_1 and v_2 are the two children of v that are assigned sequences s_1 and s_2, respectively. s is either s_1 or s_2.

Case 5: Let v be a merge node. $d[v,s]$ be the smallest cost of sub-topology rooted at v such that v is assigned the sequence s. $d[v,s]$ can be computed as:

$$d[v,s] = \min_{all\ possible\ s_1\ and\ s_2} d[v_1,s_1] + dist(s,s_1) + dist(s,s_2). \tag{10}$$

where v has two children v_1 and v_2 which are assigned s_1 and s_2, respectively. v_1 is the child that is not lower than the other child of v. Again, s is either s_1 or s_2.

Note that, if v is a leaf in T and v is assigned a sequence s, then $d[v, s] = 0$ and $d[v, s'] = \infty$ for any $s' \neq s$. We can compute all the $d[v, s]$'s bottom up. The algorithm is shown in Figure 9.

The running time depends on the time to compute each $d[v, s]$. From equations (6)–(10), the runing time could be as high as $O(k^6)$ if the involved $dist(s, s')$s and $r\_dist(s, s_1, s_2)$'s are previously computed. In fact, using the data structure called *extended tree* in [22], the running time of the algorithm purely depends on the number of $dist(s_1, s_2)$'s and $r\_dist(s_1, s_2, s_3)$'s used in (6)-(10). Similar to [22], we can reduce the total number of $dist(s_1, s_2)$'s and $r\_dist(s_1, s_2, s_3)$'s used in the algorithm to $O(kd)$, where k is the number of given sequences and d is the depth of the topology. Thus, we can get an $O(kd)$ algorithm, if all involved $dist(s_1, s_2)$'s and $r\_dist(s, s_1, s_2)$'s have been previously computed. The computation of each $dist(s_1, s_2)$ needs $O(n^2)$ time if the length of sequences is n. The computation of $r\_dist(s, s_1, s_2)$ needs $O(n^3)$ time and $O(n^2)$ space using the method in [12].

Now, we can conclude that

Theorem 5 *When the given topology satisfies (C1) and (C2), there is a ratio-3 approximation algorithm that takes $O(kd + kdn^3)$ time.*

FTAR with fixed number of crossovers

The above algorithm does not always give an alignment. Thus it only works for FTHR. However, the bound developed in Theorem 4 also holds for FTAR. Now, we modify Algorithm 1 to work for FTAR when each recombination contains at most a constant number, say, c, of crossovers. The key observation is that when the top node v has a parallel node v', i.e., Cases 2 and 3, the recombination imposes c fixed pairs of indexes for the two sequences s and s' that are assigned to v and v', respectively, in the resulting alignment. Thus, the recurrence equations are similar to those in (6)-(10) except that we have to keep c pairs of fixed indexes.

For simplity, we give the recurrence equations for the case, where each recombination contains one crossover. For Case 1, the equation is the same as (6).

Case 2: Let $d[v, s, s', i, j]$ be the smallest cost of sub-topology rooted at v such that v is assigned the sequence s and v' is assigned the sequence s' and the crossover breaks s and s' into two parts, $s[1...i]$ and $s[i+1...n]$, and $s'[1...j]$ and $s'[j+1...n]$. (See Figure 8 (b).)

$$d[v, s, s', i, j] = \min_{allpossible s_3} d[v_1, s] + d[v_2, s'] + d[v_3, s_3] + r\_dist(s_3, s, s', i, j), \tag{11}$$

where $r\_dist(s_3, s, s', i, j)$ is the recombination distance such that there is one crossover and it breaks s and s' into two parts, $s[1...i]$ and $s[i+1...n]$, and $s'[1...j]$ and $s'[j+1...n]$.

Case 3: Let $d[v, s, s', i, j]$ be the smallest cost of sub-topology rooted at v such that v is assigned the sequence s and v' is assigned the sequence s' and the crossover divides s_1 and s_2 into two parts $s_1[1...i]$ and $s[i+1...n]$, and $s_2[1...j]$ and $s_2[j+1...n]$. (See Figure 8 (c).)

$$\begin{aligned} d[v, s, s', i, j] &= \min_{all\ possible\ s,\ s_1, s_2, s_3\ ,\ s_4,\ i'\ and\ j'} dist(s, s_1, i, i') + dist(s, s_2) + dist(s', s_4) \\ &+ dist(s', s_3, j, j') + d[v_1, s_1, s_4, i', j'] + d[v_2, s_2] + d[v_3, s_3], \end{aligned} \tag{12}$$

where $dist(s_1 s_2, i, j) = dist(s_1[1..i], s_2[1...j]) + dist(s_1[i+1...n], s_2[j+1...n])$.

Case 4: Let $d[v, s, s', i, j]$ be the smallest cost of sub-topology rooted at v such that v is assigned the sequence s and v' is assigned the sequence s' and the crossover breaks s and s' into two parts, $s[1...i]$ and $s[i+1...n]$, and $s'[1...j]$ and $s'[j+1...n]$. (See Figure 8 (d).)

$$d[v, s, , s', i, j] = \min_{all\ possible\ s_1\ s_2\ ,and\ i'} d[v_1, s_1, s', i', j] + d[v_2, s_2] + dist(s_1, s_2), \tag{13}$$

where v_1 and v_2 are the two children of v that are assigned sequences s_1 and s_2, respectively; v' is assigned the sequence s'; and s is either s_1 or s_2.

Case 5: Let v be a merge node. $d[v, s]$ be the smallest cost of sub-topology rooted at v such that v is assigned the sequence s. $d[v, s]$ can be computed as:

$$d[v, s] = \min_{all\ possible\ s_1\ ,\ s_2,\ i\ and\ j} \{d[v_1, s_1, s_2, i, j] + dist(s_1, s_2, i, j)\}. \tag{14}$$

where v has two children v_1 and v_2 which are assigned s_1 and s_2, respectively. v_1 is the child that is not lower than the other child of v. Again, s is either s_1 or s_2.

Again, using the techniques in [22], computing the value in (12) requires the highiest time complexity, i.e., $O(n^2)$. Therefore, we can get an algorithm with time complexity $O(kdn^3 n^{4c})$ for the case where each recombination contains at most c crossovers.

5 A PTAS

We can extend the ratio-3 algorithm for FTAR with bounded number of crossovers to a PTAS. The basic idea is to partition the extended topology into a set of slides, each of which contains t levels of nodes. We number the levels from top to bottom. The root is at level 1. The top slide can contain 1, 2, ..., t levels of nodes. Thus, there are t different partitions P_1, P_2, ..., P_t, where the top slide of P_p contains p levels. (See Figure 10.) Each P_p partitions the topology into *small topologies*, each of which contains at most t levels of nodes. A *boundary node* for P_p is a node at the j-th level where j mod t = p. Since in the extended topology some edges cross more than one level, they might be cut by all the t different partitions. (This case happens when an edge crosses more than t level.) If so, it will result in a constant ratio.

An edge that crosses more than one level is called a *long* edge, whereas other edges are called *short* edges. For a fixed partition P_p, if two small topologies obtained from P_p share a common long edge, we construct an edge (u, v), where u stands for the upper small topology and v stands for the lower small topology. In this way, we can get a graph $G(p)$, called a *component* graph for partition P_p, to describe the relationships among small topologies. From (C2), it is easy to see that $G(p)$ is a forest (a set of trees). Thus, we can organize the nodes in $G(p)$ level by level. For each $G(p)$, we further cut the nodes (small topologies in T) for every t levels and obtain t different partitions $P_{p,q}$ for $q = 1, 2, \ldots, t$. Here we number the levels of $G(p)$ from top to bottom and the root is at level 1. Therefore, we have t^2 different partitions. A *boundary node* v for $P_{p,q}$ is a boundary node for P_p that is not incident to any long edge, or if v is incident to a long edge then the small topology that v is in is at level j

in $G(p)$ such that $j \mod \mathrm{t} = \mathrm{q}$. A *boundary edge* for $P_{p,q}$ is an edge such that at least one of its ends is a boundary node for $P_{p,q}$.

We then (1) uniformly lift the sequences to the internal nodes in T, (2) keep the lifted sequences on boundary nodes for $P_{p,q}$, and (3) reconstruct the sequences for the rest of the nodes in the topology such that the total cost of the topology is minimized.

A fully labeled topology thus obtained is called a $(t-t)$-uniform lifted topology. Since there are 2^d uniform lifted choices and t^2 different partitions , there are $2^d \times t^2$ different $(t-t)$-uniform lifted topologies. We use $T_{p,q}(i)$ to denote the $(t-t)$-topology obtained from partition $P_{p,q}$ and uniform lifting choice i. The total cost of those $2^d \times t^2$ $(t-t)$-uniform lifted topologies is bounded as follows:

Theorem 6

$$\sum_{p=1}^{T}\sum_{q=1}^{t}\sum_{i=1}^{2^d} C(T_{p,q}(i)) \leq 2^d \times t^2 C(T^{min}) + 4 \times 2^d \times tC(T^{min}). \tag{15}$$

Corollary 7 *There exists a $(t-t)$-uniform lifted topology with cost at most $(1+4/t)C(T^{min})$.*

Note that, Theorem 6 and Corollary 7 hold for both FTHR and FHAR. However, it is hard to find a polynomial time algorithm to compute an optimal $(t-t)$-uniform lifted topology. Fortunately, we can give a dynamic programming algorithm to compute an optimal $(t-t)$-uniform lifted topology for FTAR when each recombination contains a bounded number of crossovers. The algorithm is described in Appendix 2.

Theorem 8 *There is a polynomial time approximation scheme (PTAS) for FTAR when every recombination contains a bounded number of crossovers and the given topology satisfies (C1) and (C2).*

6 Remarks

The time complexity for the PTAS proposed here is too high so that we can not try the PTAS even for short sequences of 100 letters. It might be possible to improve the ratio of PTAS to $1 + \frac{5}{t} - \frac{4}{t2^t}$ or $1 + \frac{4}{t+1} + \frac{1}{t}$ with the same time complexity using the methods in [22, 23]. However, even if we can do so, the time complexity is still too high and it is impossible to try the PTAS for sequences of 100 letters.

7 Acknowledgement

We thank Tao Jiang for discussions on the topics studied in this paper.

Appendix 1: proofs

proof of Theorem 1:

Proof. The reduction is from the dominating set problem.

Dominating set: Instance: Graph $G =< V, E >$ and an integer K.
Question: is there a dominating set V' of size K or less for G, i.e., a subset $V' \subseteq V$ such that for all $u \in V - V'$ there is a $v \in V'$ for which $(u, v) \in E$.

It is known that dominating set cannot be approximated within any constant performance ratio unless $P = NP$ [10].

Let $G = (V, E)$ be an instance of dominating set, where $V = \{v_1, \cdots, v_n\}$, and $E = \{e_1, \cdots, e_m\}$. d_v denotes the degree of node v.

For each $v \in V$, we construct two sets of nodes $\{u_0^v, \cdots, u_{d_v+1}^v\}$ and $\{w_1^v, \cdots, w_{d_v}^v\}$. We then (i) link all u_0^v's to a root by normal edges; (ii) add edges (u_i^v, u_{i+1}^v) and (w_i^v, w_{i+1}^v); (iii) add edge (u_1^v, w_1^v); and (iv) attach a leaf a_v to every u_0^v and a leaf b_v to every $u_{d_v+1}^v$. See Figure 1(a). For each $e = (v, v') \in E$, select i and i' such that $u_i^v, u_{i'}^{v'}$ has only one child each at present, select j and j' such that $w_j^v, w_{j'}^{v'}$ has only one parent each at the present, then add edges $(u_i^v, w_{j'}^{v'})$ and $(u_{i'}^{v'}, w_j^v)$ in T. Note that, adding edges $(u_i^v, w_{j'}^{v'})$ and $(u_{i'}^{v'}, w_j^v)$ creates two recombination nodes. See Figure 1 (c). As an example, Figure 1 (b) gives an instance of dominating set and Figure 1 (c) gives the constructed instance.

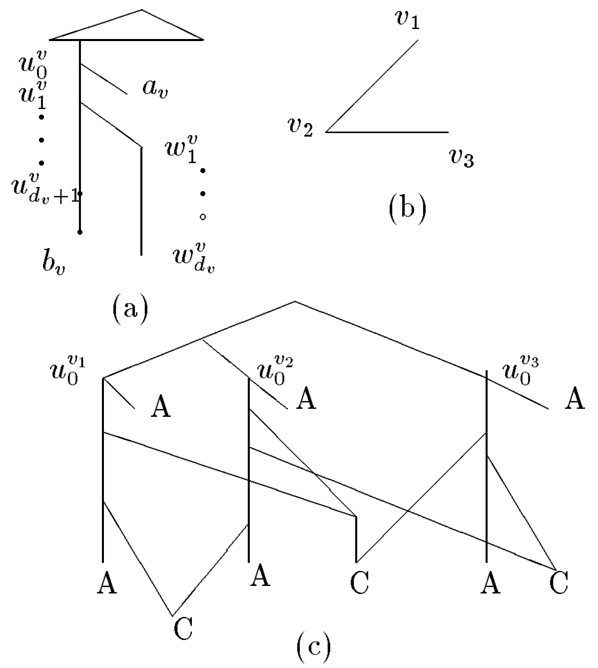


Figure 1: (a) The basic component. (b) The graph G. (c) The constructed instance T

It's easy to see that w_i^v's are recombination nodes and the rest of nodes are normal nodes. In the constructed topology, a_v's and b_v's are normal leaves and $w_{d_v}^v$'s are recombination leaves. Now, label a_v, b_v with A, and $w^v_{d_v}$ with AC. Define the score scheme as follows: $d(A,C) = 1$, $d(\Delta, A) = \frac{1}{2}$ and $d(\Delta, C) = \frac{1}{2}$. The crossover penalty is 0.

Let $OPT(T)$ be the cost of an optimal solution for fixed topology alignment with recombination. Let $OPT(G)$ be the size of the dominating set problem. The following lemma can be proved:

Lemma 9

$$OPT(T) \leq 2timesOPT(G). \tag{16}$$

Moreover, given a solution for fixed topology alignment with recombination of cost c, we can find a dominating set of size $2c$.

Proof. Given a dominating set V' of size c, we can get a solution for FTHR by assigning letter C to u^v_i's, where $i = 1, 2, \ldots, d_v + 1$ and $v \in V'$, and A to the rest of internal nodes. Thus, each path u^v_1. u^v_2, …, $u^v_{d_v+1}$, where $v \in V'$, contributes cost 2 and the rest of edges has cost 0. That is, the total cost for FTHR is $2c$. Therefore, $OPT(T) \leq 2 \times OPT(G)$.

Given a solution for FTHR with cost c, there are c $A - C$ mismatches or equivalently there are c pairs of $A - \Delta$ and $C - \Delta$ mismatches. We can construct dominating set V' as follows: (1) If any u^v_i ($i = 0, 1, \ldots, d_v + 1$) is assigned a sequence containing C, (In this case, it contributes cost by at least 1.) then node v is in the dominating set v'. (2) Consider a recombination node, say, $w^{v'}_j$. At least one of its parents (possibly two parents) is of u^v_i type. If neither of its parents is assigned a sequence containing C, (In this case, getting a C costs at least 0.5.) then v is in V', where u^v_i for some i is one of the parents. Thus, the size of V' is at most $2c$. ∎ From Lemma 9 and the result in [10], it follows that fixed topology alignment with recombination cannot be approximated within any constant ratio.

It is easy to see that the construction also works for FTHR. ∎

proof of Lemma 3:

Proof. Induction on the depth of the extended layout d. ∎

proof of Theorem 4:

Proof. Let T be the given topology and d be the depth of the deformed layout of T. There are 2^d uniform lifted topologies $T(i)$'s ($i = 1, 2, \ldots, 2^d$). If we ignore the recombination edges, the remaining part of $T(i)$ is a set of binary trees, i.e., a forest $F(i)$. Let F^{min} be the forest obtained by deleting all the recombination edges from T^{min}. Consider the non-zero cost of edges in forest $F(i)$. **(i)** The cost of each normal edge (u, v) in $F(i)$, where v is the parent of u, and v and u are assigned different sequences $v(i)$ and $\hat{v(i)}$, corresponds to the total cost of two paths from v to the two different descendent leaves $v(i)$ and $\hat{v(i)}$ in F^{min}. (See Figure 2.)

Let $C(F(i))$ be the cost of $F(i)$. From Lemma 3, the total cost of the 2^d uniform lifted forests is bounded as follows:

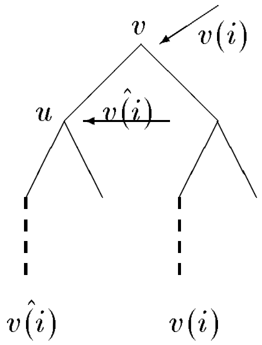


Figure 2: The bound for an ordinary edge (v, u), where u is assigned sequence $\hat{v(i)} = u(i)$ and v is assigned the sequence $v(i)$. The dashed-lines lead to the leaves in T.

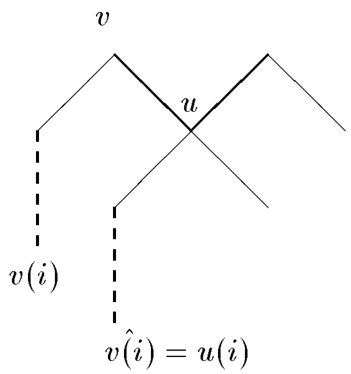


Figure 3: The bound for recombination edge (u, v) (heavy edge), where u is assigned sequence $\hat{v(i)} = u(i)$ and v is assigned the sequence $v(i)$. The dashed-lines lead to the leaves in T.

$$\sum_{i=1}^{2^d} C(F(i)) \leq 2 \times 2^d C(F^{min}) - \sum_{i=1}^{2^d} \sum_{all\ roots\ r_j's\ in\ forest\ F(i)} \{\mu_{r_j}^{r_j(i)} + \mu_{r_j}^{\hat{r_j(i)}}\}\}. \tag{17}$$

(ii) The cost of each recombination edge (v, u) in $T(i)$, where v is the parent of u, and v and u are assigned sequences $v(i)$ and $\hat{v(i)}$, respectively, can be upper bounded by the total cost of the two paths from v to $v(i)$ and $\hat{v(i)}$ in T^{min}. (See Figure 3.) The cost $C(P(v, \hat{v(i)})$ of path $P(v, \hat{v(i)})$ from v to $\hat{v(i)}$ is further divided as follows:

$$C(P(v, \hat{v(i)})) \leq C(v, u) + C(P(u, \hat{v(i)})), \tag{18}$$

where $C(v, u)$ is the cost of edge (v, u) in T^{min} and $C(P(u, \hat{v(i)}))$ is the cost of the path $P(u, \hat{v(i)})$ from u (a root of a binary tree in $F(i)$) to $\hat{v(i)}$ in T^{min}. Note that, $\hat{v(i)} = u(i)$. For each recombination node u, the cost $C(P(u, u(i)))$ should be charged twice, once for each of the two recombination edges incident to u in a uniform lifted topology. Moreover, in (17), there are two terms $\mu_{r_j}^{r_j(i)}$ and $\mu_{r_j}^{\hat{r_j(i)}}$ for each root r_j (recombination node) in $F(i)$. Consider all 2^d uniform lifted topologies, from (5) the total of all $C(P(u,\hat{v(i)}))$'s in (18) is exactly the total of all $\{\mu_{r_j}^{r_j(i)} + \mu_{r_j}^{\hat{r_j(i)}}\}$'s in (17).

That is,

$$\sum_{i=1}^{2^d} \sum_{u \in Rec(T)} 2 \times C(P(u,u(i))) \leq \sum_{i=1}^{2^d} \sum_{r_j inF} \{\mu_{r_j}^{r_j(i)} + \mu_{r_j}^{\hat{r_j(i)}}\}, \tag{19}$$

where $Rec(T)$ is the set of all recombination nodes in T.

Note that v, the parent of a recombination node, has only one child in $F(i)$. (See Figure 3.) This kind of nodes are at the middle of edges in binary trees of $F(i)$. The cost $C(P(v,v(i)))$ can be upper bounded by $C(P(w,v(i)))$, where w is the unique parent of v in T. From (C2), w has only one child that is a parent of a recombination node. Let v be a parent of a recombination node. Again, from Lemma 3, the total of all $C(P(v,v(i)))$'s among 2^d uniform lifted topologies is bounded by

$$\sum_{i=1}^{2^d} \{ \sum_{v \in Par\_Rec(T)} \mu_v^{v(i)} \} \leq \sum_{i=1}^{2^d} \{ \sum_{v \in V(T) \cup \{r\}} \mu_v^{v(i)} \} \leq 2^d C(F^{min}) \leq 2^d C(T^{min}), \tag{20}$$

where $Par\_Rec(T)$ is the set of all parents of recombination bodes in T and r is the root of T. From (17), (19) and (20), we have:

$$\sum_{i=1}^{2^d} C(T(i) \leq 3 \times 2^d C(T^{min}). \tag{21}$$

Therefore, we can conclude that there is a uniform lifted topology whose cost is at most three times of the optimum. ■

proof of Theorem 6:

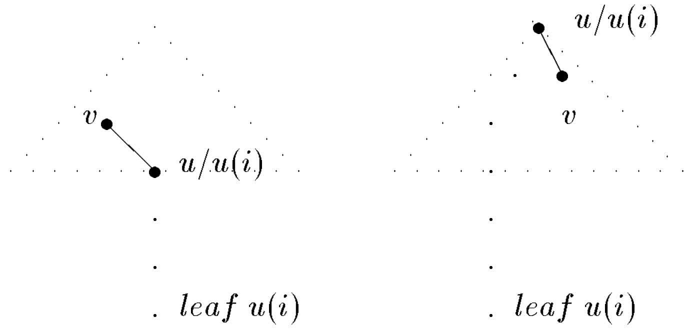
Proof. Let $u(i)$ be the descendent leaf of u whose associated sequence is lifted to u for the uniform lifted choice i. For a fixed partition $P_p/P_{p,q}$ of T, consider the cost $C(u,v)$ of a short boundary edge (u,v) in $T_{p,q}(i)$, where u is a boundary node for $P_p/P_{p,q}$ that is assigned a sequence $u(i)$ and v is assigned the corresponding sequence in the optimal solution T^{min}.

$$C(u,v) \leq \rho(u,v) + \mu_u^{u(i)}, \tag{22}$$

where $\rho(u,v)$ is the cost of edge (u,v) in the optimal solution T^{min}, and $\mu_u^{u(i)}$ is the cost of the path from u to $u(i)$ in the optimal solution. (See Figure 4.)

Let $C(Short_{p,q}(i))$ denote the total cost of all the short edges in $T_{p,q}(i)$, and $C(Short^{min})$ be the cost of the short edges in T^{min}. From (22), we have the following bound.

$$\begin{aligned} \sum_{i=1}^{2^d} C(Short_{p,q}(i)) &\leq 2^d \times C(Short^{min}) + 3 \times \sum_{u \in B_{p,q}} \sum_{i=1}^{2^d} \mu_u^{u(i)} & (23) \\ &\leq 2^d \times C(Short^{min}) + 3 \times \sum_{u \in B_p} \sum_{i=1}^{2^d} \mu_u^{u(i)}, & (24) \end{aligned}$$



(a) (b)

Figure 4: The bound of cost ($C(u, v)$ in $T_{p,q}(i)$. The light dotted lines stand for a merged component, the heavy dotted lines stand for the paths from u to the leaf $u(i)$.

where $B_{p,q}$ is the set of boundary nodes for partition $P_{p,q}$ and B_p is the set of boundary nodes for partition P_p. The coefficient 3 in (23) is from the facts that (1) each normal node in T has degree at most 3 and thus each contributes 3 paths and (2) every recombination node has two children and each of them contributes one path, and every recombination node has two parents and the two parents together contributes one path.

Let $C(Long_{p,q}(i))$ be the cost of all long edges in $T_{p,q}(i)$. Consider the $t \times 2^d$ uniform lifted topologies.

$$\begin{aligned}
\sum_{q=1}^{t}\sum_{i=1}^{2^d} C(T_{p,q}(i)) &= \sum_{q=1}^{t}\sum_{i=1}^{2^d}\{C(Short_{p,q}(i)) + C(Long_{p,q}(i))\} && (25)\\
&\leq \sum_{q=1}^{t}\{2^d \times C(T^{min}) + 3\sum_{u\in B_p}\sum_{i=1}^{2^d}\mu_u^{u(i)}\} + \sum_{q=1}^{t}\sum_{u\in B_{p,q}}\sum_{i=1}^{2^d}\mu_u^{u(i)} && (26)\\
&\leq t\times 2^d C(T^{min}) + 3\sum_{q=1}^{t}\sum_{u\in B_p}\sum_{i=1}^{2^d}\mu_u^{u(i)} + 2^d C(T^{min}). && (27)
\end{aligned}$$

The first two terms in (26) are from (24). The last term in (26) is from the fact that each long edge contributes one more path. (27) is from Lemma 3. Now, consider different P_p's for $p = 1, 2, \ldots, t$. From (27), we have

$$\begin{aligned}
\sum_{p=1}^{t}\sum_{q=1}^{t}\sum_{i=1}^{2^d} C(T_{p,q}(i)) &\leq t^2 2^d C(T^{min}) + 3\sum_{p=1}^{t}\sum_{q=1}^{t}\sum_{i=1}^{2^d}\sum_{u\in B_{p,q}} \mu_u^{u(i)} + t \times C(T^{min}) && (28)\\
&\leq t^2 2^d C(T^{min}) + 3t \times C(T^{min}) + t \times C(T^{min}) && (29)\\
&= t^2 2^d C(T^{min}) + 4t \times C(T^{min}). && (30)
\end{aligned}$$

(29) is again from Lemma 3. ■

Appendix 2: The description of the PTAS

Now we give an outline of a dynamic programming algorithm to compute an optimal $(t-t)$-uniform lifted topology for FTAR when every recombination contains one crossover. Note that, the boundary nodes in $P_{p,q}$ partition the topology into *merged* components each of which contains at most t^2 levels of nodes and at most 2^t long edges. Let v be a root of a merged component MT_v. Define top_v be the subtopology with v at top containing every node in MT_v and every node below v that has a descendent node in MT_v. Note that top_v could contain $O(k)$ nodes in the worst case, where k is the number of given sequences.

Similar to the ratio-3 algorithm, we consider the same five cases.

Cases 1 and 5: We define $d_{p,q}[v,s]$ as the cost of the sub-topology with v at the top level such that s is uniformly lifted to v and every boundary node for $P_{p,q}$ is assigned a uniform lifted sequence and every other node is assigned a sequence computed by the exact algorithm for small components. $d_{p,q}[v,s]$ can be computed as

$$d_{p,q}[v,s] = \min_{all\ possible\ s_u} \min_{i_u\ and\ j_u} \{cost(top_v) + \sum_{u\in bottom(top_v)} \hat{d}_{p,q}[u,s_u]\}, \tag{31}$$

where $cost(top_v)$ is the cost of top_v, $bottom(top_v)$ is the nodes in top_v that does not have any descendent node in top_v, and only one node in each pair of parallel nodes is in $bottom(top_v)$. Moreover,

$$\hat{d}_{p,q]}[u,s_u] = d_{p,q}[u,s_u]$$

if u is Case 1 or Case 5;

$$\hat{d}_{p,q}[u,s_u] = d_{p,q}[u, s_u, s'_u, i_u, j_u]$$

if u is is Case 2, 3, or 4. ($d_{p,q}[u,s_u,s'_u,i_u,j_u]$ is defined in case 2, 3, and 4.)

Case 2: Let $d_{p,q}[v,s,s',i,j]$ be the smallest cost of sub-topology rooted at v and v' for $P_{p,q}$ such that v is assigned the sequence s and v' is assigned the sequence s' and the crossover breaks s and s' at i and j. (See Figure 8 (b).)

$$d_{p,q}[v,s,s',i,j] = cost(top_v,i,j) + \sum_{u\in bottom(top_v)} \hat{d}_{p,q}[u,s_u], \tag{32}$$

where $cost(top_v,i,j)$ is the cost of top_v such that the crossover breaks the two sequences on the two roots of top_v at i and j, respectively.

Case 3: Let $d[v,s,s',i,j]$ be the smallest cost of sub-topology rooted at v and v' for $P_{p,q}$ such that v is assigned the sequence s and v' is assigned the sequence s' and the crossover divides s_1 and s_2 at i and j, respectively. (See Figure 8 (c).) **(3.1)** if the top components MT_v and $MT_{v'}$ are the same for $P_{p,q}$, i.e., the top component contains the recombination node, the equation is the same as in (32). **(3.2)** Otherwise, we use the following equation:

$$\begin{aligned} d[v,s,s',i,j] \quad = \quad & \min_{all\ possible i_u,\ j_u\ and\ s_u\ for\ u\in bottom(top_v)\ and\ i_{u'},\ j_{u'}\ and\ s_{u'}\ for\ u'\in bottom(top_{v'})} \\ & cost(top_{v,s},i) + cost(top_{v',s'},j) + \sum_{u\in bottom(top_v)} \hat{d}_{p,q}[u,s_u] \\ + \quad & \sum_{u'\in bottom(top_{v'})} \hat{d}_{p,q}[u',s_{u'}]. \end{aligned} \tag{33}$$

Case 4: Let $d[v, s, s', i, j]$ be the smallest cost of sub-topology rooted at v such that v is assigned the sequence s and v' is assigned the sequence s' for $P_{p,q}$ and the crossover breaks s and s' at i and j. (See Figure 8 (d).) **(4.1)** If MT_v contains the recombination node, then the equation is the same as (32). **(4.2)** If $P_{p,q}$ leads to one top component, (the other is a long edge,) then we use:

$$\begin{aligned} d[v, s, s', i, j] &= \min_{all\ possible i_u,\ j_u\ and\ s_u\ for\ u \in bottom(top_v)} cost(top_{v,s}, i) \\ &+ \sum_{u \in bottom(top_v)} \hat{d}_{p,q]}[u, s_u]. \end{aligned} \tag{34}$$

(4.3) Otherwise, we use (33).

We can compute all the cells bottom up in polynomial time. We then select the smallest $d_{p,q}[r, s, i, j]$ as the cost of the optimal $(t-t)$-uniform lifted topology. A standard back-tracking process can give the sequence on each node in T. Using the techniques in [22], one can obtain an $O(t^2 k d n^{t^2})M(t)$ algorithm with ratio $1 + \frac{4}{t}$, where k is the number of given sequences, d is the depth of the given topology, n is the length of the given sequences, $M(t)$ is the time required to compute the top_v , and t is the parameter that controls the accuracy of the approximation solution.

The computation of top_v

The computation of top_v involves an exact algorithm for FTAR under condition (C1) and (C2) when each recombination contains a bounded number of crossovers. For simplity, we give an outline for the case, where each recombination contains one crossover.

Let $s_1, s_2, \ldots, s_k$ be the given sequences and T be the given topology. Recall that, for fixed topology alignment we use $d[i_1, i_2, \ldots, i_k]$ to indicate the cost of the alignment for subsequences $s_j[1, i_j]$'s $(j = 1, 2, \ldots, k)$, where $s_j[1, i_j]$ is the subsequence of s_j containing the first i_j characters [14, 16]. Our algorithm for the new problem is similar. We consider different configurations of the present column in the alignment. For the recombination nodes in T, we can choose one of the parents in the construction of the present column. After we make a choice, the topology becomes a binary tree. Similar to tree alignment, we can compute the cost of the alignment for the subsequences $s_j[1, i_j]$ $(j = 1, 2, \ldots, k)$. Thus, the difference here is that we need an extra index f_q for each recombination node q in T. Each f_q can be either 0 and 1. 0 means that the left parent is selected and 1 means that the right parent is selected.

Let $d[i_1, i_2, \ldots, i_k, f_1, f_2, \ldots, f_m]$ be the minimum cost of the alignment for the subsequences $s_j[1, i_j]$'s $(j = 1, 2, \ldots, k)$ such that the parent of each recombination node q is chosen according to the value of f_q. We can compute $d[i_1, i_2, \ldots, i_k, f_1, f_2, \ldots, f_m]$ as follows:

$$d[i_1, i_2, \ldots, i_k, f_1, f_2, \ldots, f_m] = \min_{possible\ i'_j s\ and\ f'_q s} \{ d[i'_1, i'_2, \ldots, i'_k, f'_1, f'_2, \ldots, f'_m] \tag{35}$$

$$+ \quad cost(last) + \sum_{i=1}^{m} |f_i - f'_i| \chi, \} \tag{36}$$

where i'_j is either i_j or $i_j - 1$, f'_q can be f_q or different from f_q, χ is the crossover penalty and $cost(last)$ is the cost for the last column that contains k letters depending on the choice of i_j's. Note that $cost(last)$ depends on not only the k letters assigned to the leaves, but also the tree structure determined by the choices of f_q's. Once the letter on each leaf as well as the tree structure is fixed, we try to put all possible letters on internal nodes of the tree structure such that the cost of the tree for this column is minimized. There are $|\Sigma|^{k-1}$ ways to put letters on internal nodes, where $|\Sigma|$ is the size of the alphabet and $k-1$ is the number of internal nodes in the tree. In fact, a dynamic algorithm algorithm can compute $cost(last)$ in $O(k|\Sigma|)$ time [16]. Since each i_j and f_q can take two values, there are at most 2^{2k} choices in (35). Thus, the time required for each $d[i_1, i_2, \ldots, i_k, f_1, f_2, \ldots, f_m]$ is $O(2^{2k}k|\Sigma|)$, where k is the number of given sequences and $|\Sigma|$ is the size of the alphabet. We can compute all the $d[i_1, i_2, \ldots, i_k, f_1, f_2, \ldots, f_m]$'s in a specific order. Since there are $2^m n^k$ $d[i_1, i_2, \ldots, i_k, f_1, f_2, \ldots, f_m]$'s, the total time required for our algorithm is $O(2^{2k}2^m n^k k|\Sigma|)$, where k is the number of given sequences in T, n the length of sequences, m the number of recombination nodes in T.

The cases, where recombinations impose break points at some specific positions can be solved by imposing that each "optimal path" in the matrix must pass some specific cells.

The cost of top_v contains two parts: the cost of MT_v and the costs of those small components in P_p involved in top_v. To make sure that we can eventually get a sequence alignment, each long edge imposes a break point. Thus, we need at most 2^t break points. Therefore, each top_v can be computed in $O(2^{3\times 2^{t\times t}} n^{2^{t\times t}+2^t} 2^{t\times t}|\Sigma|k)$ time.

Theorem 10 *The PTAS runs in $O(2^{t^2} kdn^{2^{t\times t+t}} 2^{3^{t\times t}} k|\Sigma|)$ time for fixed t.*

Appendix 3: figures in the text

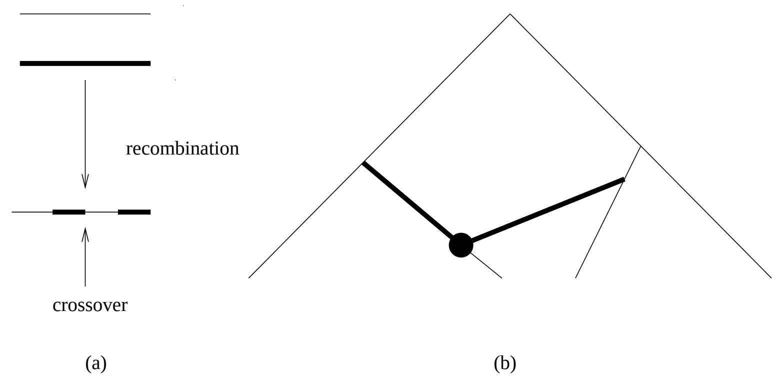


Figure 5: (a) Recombination operation. (b) The topology. The dark edges are recombination edges. The circled node is a recombination node.

```
                /  s1:AABABB
               /\ s1
              /  \
          s2 /\  /\ s3
            /  \/  \
           /        \
          /          \
         / s4: AAABBB \
        /              \                 s1: AABABB        AAABB  CABBB
                                         s2: AA ABB             \ /
                                         s3: CAB BB              | s4': AABBB  cost  0.5
                                        s4': AAB BB        s4:AAABBB           cost   1
s2: AAABB               s3:CABBB          s4:AAAB BB
             (a)                              (b)                         (c)
```

Figure 6: An optimal alignment.

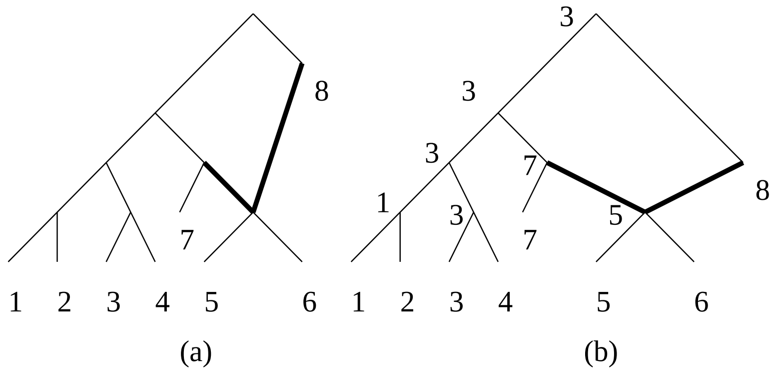


Figure 7: (a) The original topology. (b) The deformed layout. The numbers stand for given sequences and they are uniformly lifted to their ancestors. Since we ignore recombination edges, 7 is lifted to its parent.

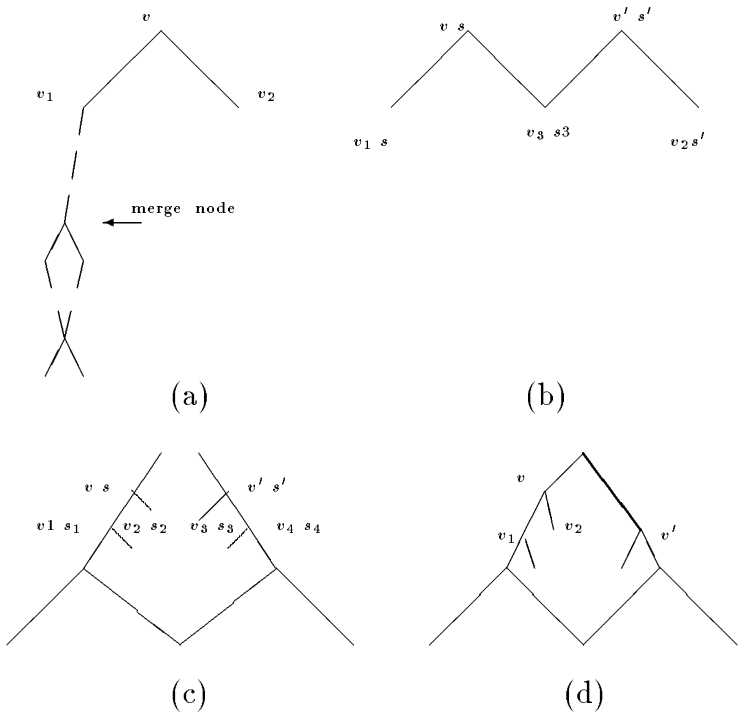


Figure 8: (a) Case 1. (b) Case 2. (c) Case 3. (d) Case 4. The heavy edge is an extended edge.

```
begin
for  each level of T, with the bottom level first do
    for each node v at the level do
    begin
       for each sequence s ∈ S(v),
         Compute d[v, s].
    end
Select an s ∈ S(r) such that d[r, s] is minimized.
Compute the uniform lifted topology with cost d[r, s] by back-tracing.
end.
```

Figure 9: **Algorithm 1**

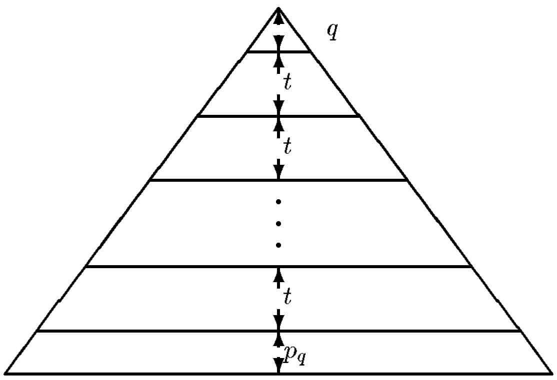


Figure 10: The partition P_q.

References

[1] S. Altschul and D. Lipman, Trees, stars, and multiple sequence alignment, *SIAM Journal on Applied Math.*, 49 (1989), pp. 197-209.

[2] B. DasGupta, X. He, T. Jiang, M. Li, J. Tromp and L. Zhang, On distances between phylogenetic trees, *Proc. 8th Annual ACM-SIAM Symposium on Discrete Algorithms*, Jan. 1997, New Orleans.

[3] B. DasGupta, X. He, T. Jiang, M. Li, and J. Tromp, On the linear-cost subtree-transfer distance, submitted to *Algorithmica*, 1997.

[4] Z. Galil and R. Ciancarlo, 11Speeding up dynamic programming with applications to molecular biology", Theoretical Computer Science **64**, pp. 107-118, 1989.

[5] D. Gusfield, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*, Cambridge University Press, 1997.

[6] J. Hein, 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 (1989), 649-668.

[7] J. Hein, Reconstructing evolution of sequences subject to recombination using parsimony, *Math. Biosci.* 98(1990), 185-200.

[8] J. Hein, A heuristic method to reconstruct the history of sequences subject to recombination, *J. Mol. Evo.* 36(1993) 396-405.

[9] J. Hein, T. Jiang, L. Wang, and K. Zhang, On the complexity of comparing evolutionary trees, *Discrete Applied Mathematics*, 71(1996), 153-169.

[10] D. Hochbaum, *Approximation Algorithms for NP-hard Problems*, PWS, to appear.

[11] S. K. Kannan and E. W. Myers, “An algorithm for locating non-overlapping regions of maximum alignment score”, CPM93, pp. 74-86, 1993.

[12] J. Kececioglu and D. Gusfield, “ Reconstructing a history of recombinations from a set of sequences”, 5th Annual ACM-SIAM Symposium on Discrete Algorithms, Arlington, Virginia, pp. 471-480, January 1994

[13] G. M. Landau and J. P. Schmidt, “An algorithm for approximate tandem repeats, CPM'93, pp. 120-133, 1993.

[14] D. Sankoff, Minimal mutation trees of sequences, *SIAM J. Applied Math.* 28(1975), 35-42.

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

[16] D. Sankoff and R. Cedergren, Simultaneous comparisons of three or more sequences related by a tree, In D. Sankoff and J. Kruskal, editors, *Time warps, string edits, and macromolecules: the theory and practice of sequence comparison*, pp. 253-264, Addison Wesley, 1983.

[17] F. W. Stahl, “Genetic recombination”, Scientific American, 90-101, February 1987.

[18] D. Swofford and G. Olson, Phylogenetic reconstruction, in *Molecular Systemtics*, D. Hillis and C. Moritz (eds), Sinauer Associates, Sunderland, MA, 1990.

[19] J. D. Watson, N. H. Hopkins, J. W. Roberts, J. A. Steitz, A. M. Weiner, “Molecular Biology of the gene, 4th edition, Benjamin-Cummings, Menlo Park, California, 1987.

[20] L. Wang and T. Jiang, On the complexity of multiple sequence alignment, *Journal of Computational Biology*, 1(1994), 337-348.

[21] L. Wang, T. Jiang and E.L. Lawler, Approximation algorithms for tree alignment with a given phylogeny, *Algorithmica*, 16(1996), 302-315.

[22] L. Wang and D. Gusfield, Improved approximation algorithms for tree alignment, *Journal of Algorithms*, vol. 25, pp. 255-173, 1997.

[23] L. Wang, T. Jiang, and Dan Gusfield, A more efficient approximation scheme for tree alignment, *Proceedings of the First Annual International Conference on Computational Molecular Biology*, pp. 310-319, 1997.

[24] M.S. Waterman, *Introduction to Computational Biology: Maps, sequences, and genomes*, Chapman and Hall, 1995.