ACADEMIC PRESS

Journal of Computer and System Sciences 65 (2002) 494–507

JOURNAL OF COMPUTER AND SYSTEM SCIENCES

www.academicpress.com

Methods for reconstructing the history of tandem repeats and their application to the human genome

Deep Jaitly,[a,1] Paul Kearney,[a,1] Guohui Lin,[b,2] and Bin Ma[c,*,3]

[a] *Department of Computer Science, University of Waterloo, Waterloo, Ont., Canada N2L 3G1*
[b] *Department of Computing Science, University of Alberta, Edmonton, Alta., Canada T6G 2E8*
[c] *Department of Computer Science, University of Western Ontario, London, Ont., Canada N6A 5B7*

Received 27 April 2001; received in revised form 13 October 2001

Abstract

The genomes of many species are dominated by short sequences repeated consecutively called tandem repeats. An understanding of the biological mechanisms that create and extend tandem repeats would be facilitated by reconstructing the history of duplication events that generated the tandem repeats. This paper explores the computational problem of reconstructing the duplication history of a tandem repeat. Specifically, the problem of reconstructing the minimum-cost duplication history is proved to be NP-hard even if the lengths and boundaries for the duplication events are fixed. When the lengths and boundaries are fixed, the minimum-cost duplication history can actually be represented by a tree. A non-trivial extension of the tree-alignment algorithms from [Wang et al. (Algorithmica 16 (1996) 302; SIAM J. Comput. 30 (1) (2000) 283)] gives a polynomial time approximation scheme (PTAS) for this special case. Experiments on more than 9000 tandem repeats from human chromosomes 1 and 22 demonstrate that our PTAS generates less costly histories in acceptable time than other heuristic methods. We also note that our PTAS works for any metric space. Therefore, our algorithm is also a PTAS for constructing a *minimum Steiner tree* (*MST*) when the order of all the regular nodes on the output Steiner tree is known.

Assigned to patriarchal poetry too sue sue sue sue shall sue sell and magnificent can as coming let the same shall shall shall shall let it share is share is share shall shall shall shall shell shell shall share is share shell can shell be shell be shell moving in in in inner moving move inner in in inner in meant meant might might may collect collected recollected to refuse what it is is it.

–Patriarchal Poetry, Gertrude Stein

*Corresponding author.

E-mail addresses: njaitly@math.uwaterloo.ca (D. Jaitly), pkearney@math.uwaterloo.ca (P. Kearney), ghlin@cs.ualberta.ca (G.-H. Lin), bma@csd.uwo.ca (B. Ma).

[1] Supported by CITO and NSERC.

[2] Supported in part by Startup Grant G227120195 from the University of Alberta.

[3] Supported in part by NSERC RGP0238748.

1. The tandem repeat history problem

It is estimated that over 10% of the human genome, the totality of human genetic information, consists of tandemly repeated sequences. In some other species, tandemly repeated sequences can even dominate the whole genome. For example, in the kangaroo rat (*Dipodomys ordii*) more than half of the genome consists of three tandemly repeated sequences: AAG (2.4 billions repetitions), TTAGGG (2.2 billion repetitions) and ACACAGCGGG (1.2 billion repetitions) [14].

An instance of a human tandem repeat appears below (Genbank[4] accession number 10120313):

```
CCTCCTCCTCCACCTCCTCCTCCTCCTCCTCCTCCTCCGCCTTCTCATCCTCCTCCACTT

CCTCCTCCTCCTCCTCCTCCCCTTCTCATCCTCCTCCTCTTCATCTACCC
```

This tandem repeat consists of 35 approximate copies of the *pattern sequence* CCT. Tandem repeats differ from other repeated genomic elements such as *short interspersed elements* (*SINEs*) which do not necessarily occur in tandem (i.e. consecutively). Tandem repeats vary greatly in the pattern sequence and the number of repeats [10] and are often approximate in the sense that the pattern sequence is not always repeated exactly.

The high variability among individuals in the number of repeated elements in a tandem repeat is the basis of DNA fingerprinting [6,7], which is a technique for identification of individuals using DNA. In fact, the American Federal Bureau of Investigation (FBI) has developed a database called *CODIS* that identifies convicted felons based on variability within tandem repeats. Variability within tandem repeat sequences is also linked to several genetic diseases including *spinobulbar muscular atrophy*, *fragile X mental retardation*, *myotonic dystrophy* and *Huntington's disease*. (See [10, p. 291] for references.)

Since the initial discovery of tandemly repeated elements [15], many theories on the biological mechanisms that create and extend tandem repeats have been proposed including *slipped-strand mispairing*, *unequal sister-chromatid exchange* and *unequal genetic recombination during meiosis* (see [1, p. 45] for details). With the completion of the human genome project, and many other genome sequencing projects, scientists will be in possession of the raw data needed to investigate the biological mechanisms of tandem repeat creation and extension. Due to the vast quantities of genomic data to be analyzed, computational tools are required to explore these data.

In this paper, we examine the problem of reconstructing the history of duplication events producing a tandem repeat. Consider the duplication history presented in Fig. 1. In Fig. 1, the sequence of duplication events that result in the tandem repeat (bottom sequence) is depicted by a sequence of trapezoids. The top of each trapezoid underlines the parent sequence that is duplicated and the bottom of each trapezoid overscores the child sequences produced. Hence, through a series of duplication events the ancestral sequence (top) is extended to the tandem repeat. Notice that duplication events have the following complexities:

- Duplication is not exact. The children may not be exact copies of the parent due to biological units mutation, insertion, and/or deletion.
- Duplication can occur anywhere along the parent sequence. That is, the boundaries of the duplication vary.

[4] Access to Genbank is available from the NCBI web page: http://www.ncbi.nlm.nih.gov/.

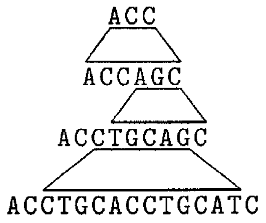


Fig. 1. The duplication history of a tandem repeat.

- The length of the parent sequence in each duplication event can be any integral multiple of the basic pattern size.

Informally, the Tandem Repeat History Problem (TRH) is to reconstruct the history of duplication events that generated a known tandem repeat. Formally, we utilize the definitions introduced by Benson and Dong [1] for consistency: Let S be a tandem repeat consisting of n approximate copies $S_1, S_2, \ldots, S_n$. Through a multiple sequence alignment, we assume that each S_i is a string of length m over the alphabet $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}, —\}$, where—indicates a gap in the alignment. Such a multiple sequence alignment can be obtained from S using tools such as the Tandem Repeats Finder [2]. Correspondingly, we assume that S is a string of length nm over Σ. A *contraction* of S is the replacement of a length $2pm$ substring of S by a new substring of length pm, where p is a positive integer. The contraction can be viewed as the opposite of a tandem duplication and can begin anywhere along the length of S. For example,

$\ldots\mathtt{AC}\underline{\mathtt{GGTAGATA}}\mathtt{CGCG}\ldots$

is contracted to

$\ldots\mathtt{AC}\underline{\mathtt{GGTA}}\mathtt{CGCG}\ldots$

by merging consecutive substrings GGTA and GATA. A contraction cost function assigns a cost to every contraction. This function typically depends on the length and similarity of the substrings being merged. For the purposes of this paper the contraction cost function used will be the sum of edit distances between the parent substring and the two child substrings.

Tandem Repeat History (TRH)
Instance: A string $S = S_1 S_2 \cdots S_n$, where $S_1, S_2, \ldots, S_n$ are length m strings over alphabet $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}, -\}$ and come from a multiple sequence alignment of the copies of a tandem repeat; A contraction cost function.
Objective: Find a series of contractions that reduce S to an ancestral sequence of length m and have the minimum cost.

Due to the complexities of TRH as listed in the above, the authors of [1] also studied a restricted version of the problem. In that restricted version, all duplication events must have length exactly m and can only begin at positions $1, m+1, \ldots$, and $(n-1)m+1$ in S. We call this restriction with fixed lengths and boundaries the Restricted Tandem Repeat History (RTRH) problem. In the RTRH problem the duplication history of a tandem repeat can be conveniently represented by a *duplication tree* as depicted in Fig. 2. In this example, the tandem repeat sequence is ATCTTCCTACGTACATTGGTAGGT, which is obtained by reading the leaf sequences from left to right. It is generated from the ancestral sequence ACCT in five duplication

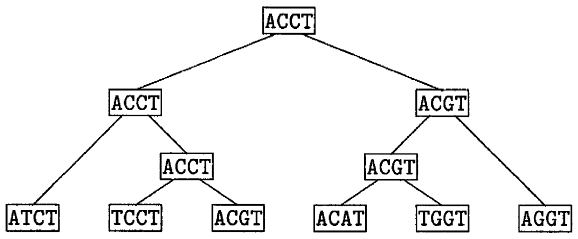


Fig. 2. The duplication history of a restricted tandem repeat.

events (each internal node of the tree represents a duplication event where the parent sequence is duplicated to produce two child sequences).

For ease of presentation, the sequence labeling with node v of the duplication tree T is denoted $l^T(v)$. The cost $c(e)$ of an edge $e = (u, v)$ in a duplication tree T takes the edit distance between the two sequences labeling with u and v. The cost of T, denoted by $c(T)$, is the sum of the edge costs in T. We give the formal definition of RTRH as follows:

Restricted Tandem Repeat History (RTRH)

Instance: A string $S = S_1S_2\cdots S_n$ where $S_1, S_2, \ldots, S_n$ are length m strings over alphabet $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}, —\}$.

Objective: Find an ordered binary tree T where each node is labeled by a length m sequence over $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}, —\}$, the leaves are labeled, from left to right, by $S_1, S_2, \ldots, S_n$, respectively, and $c(T)$ is minimized.

For the reason that an understanding of the duplication history of tandem repeats may lead insights into the biological processes that generate tandem repeats [4,5,8,9] (See [1, p. 45]), there is a need to analyze the duplication histories for all tandem repeats within entire genomes. Such a study requires the development of accurate and efficient computational techniques for recovering the duplication histories.

2. Previous results and our contributions

Benson et al. [1] developed greedy algorithms Greedy-TRHist for TRH and Greedy-TRHist-Restricted for RTRH. Both of them look for the contraction of minimum cost at each round, implement the contraction found. The difference between them is that the unrestricted algorithm permits duplications to occur anywhere along the length of the tandem repeat and the lengths of the duplications can be any integral multiple of the length of the basic pattern. Greedy-Trhist runs in time $O(mn^3)$ and Greedy-TRHist-Restricted runs in time $O(mn^2)$, where m is the length of the ancestral sequence and mn is the length of the tandem repeat. Unfortunately, neither Greedy-TRHist nor Greedy-TRHist-Restricted is guaranteed to produce a history with provable accuracy with respect to the optimal history. Nonetheless, Benson et al. demonstrated by a simulation study that Greedy-TRHist-Restricted typically produces a significantly better duplication tree than the approximation algorithms based on minimum order spanning tree, which guarantee a performance ratio of 2.

It is noteworthy that RTRH is actually a special Steiner Tree Problem under edit-distance, where $S_1, S_2, \ldots, S_n$ are regarded as n regular points. The labels for the internal nodes in the output duplication tree can be viewed as Steiner points. The specialty lies in that there is one additional constraint to the standard Steiner Tree Problem that those regular points $S_1, S_2, \dots,$ and S_n must be in a left-to-right order in the output tree.

The Steiner Tree Problem is known to be MAX-SNP hard [3] in an arbitrary metric space. Therefore, it does not admit a PTAS. On the positive aspect, the authors of [16] proved that when the topology of the Steiner tree is known, construction of minimum Steiner trees with the same topology (known as Fixed Topology Steiner Tree Problem) admits a PTAS. However, this result cannot be employed directly to solve RTRH since in an instance of RTRH we know only the order of the *nodes* but not the topology of the history.

We contribute the following results:

- RTRH is NP-hard. This hardness result establishes that approximation algorithms and heuristics are in need to reconstruct tandem repeat histories feasibly.
- RTRH admits a polynomial time approximation scheme (PTAS), which allows to compute an approximation to the optimal history with arbitrary given accuracy in polynomial time. The PTAS improves upon the *lifted tree* method for Tree Alignment [16].[5]
- The PTAS for RTRH is implemented to recover the duplication histories for over 9000 tandem repeats in human chromosomes 1 and 22. The histories recovered almost always have less cost than histories predicted by other methods.
- The PTAS for RTRH is more powerful in that it works for any distance that satisfies *Triangular Inequality*. Therefore, it is also a PTAS for constructing a minimum Steiner tree when the order of all the regular nodes on the tree is known, which strengthens the corresponding result for Fixed Topology Steiner Tree Problem.

3. RTRH is NP-hard

Obviously, if n is a constant, RTRH can be solved in polynomial time by exhaustive search. However, in general it is NP-hard as demonstrated below.

Theorem 1. *RTRH is NP-hard.*

Proof. We reduce the Max Cut problem, known to be NP-hard, to RTRH. When every node of the given graph has degree at most 3, Max Cut is still NP-hard [11]. Since every degree 1 or 2 node can be made to be degree 3 by attaching some small constant graphs, it is not hard to prove that Max Cut is NP-hard for regular graphs of degree 3 (also known as *cubic graphs*). We begin our reduction from such an instance of Max Cut. Suppose that $G = (V, E)$ is an undirected graph, where $V = \{v_1, v_2, \dots, v_n\}$ and $\deg(v_i) = 3$ for every i. We construct the corresponding instance I of RTRH as follows.

For every vertex $v_i \in V$, we construct 6 sequences $s_{i,1}, s_{i,2}, \dots, s_{i,6}$. This results in $6n$ sequences, $s_{1,1}, \dots, s_{1,6}, s_{2,1}, \dots, s_{2,6}, \dots, s_{n,1}, \dots, s_{n,6}$, which are the strings in the

[5] After the work and the manuscript of the paper had been done, we realized at RECOMB 2001 that Tang et al. [13] also considered the RTRH problem, motivated by another biological problem, *Tandem Gene Duplication*. Independently, they presented a 2-approximation algorithm and claimed that refinements on their algorithm lead to a PTAS.

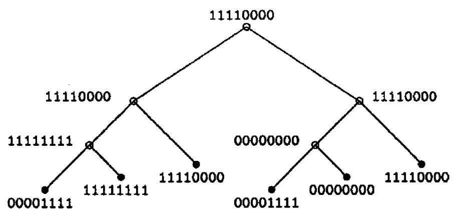
instance I. For every $v_i \in V$, each of the six corresponding sequences consists of four adjoint segments p_1, p_2, p_3, and p_4:

1. The first segment p_1 consists of n pieces, each piece is of length N_1. The ith piece consists of N_1 consecutive 1's. Except this piece, the other positions are 0's. This segment attempts to ensure that any optimal solution to I groups $s_{i,1}, s_{i,2}, \ldots s_{i,6}$ in a single subtree T_i for each $i = 1, 2, \ldots, n$. This can be achieved by a sufficient large number N_1.
2. The second segment p_2 consists of N_2 consecutive 1's for $s_{i,1}, s_{i,2}, s_{i,3}$, and N_2 consecutive 0's for $s_{i,4}, s_{i,5}, s_{i,6}$. This segment attempts to ensure that any optimal solution to I groups $s_{i,1}, s_{i,2}, s_{i,3}$ in a single subtree T_i^1 and $s_{i,4}, s_{i,5}, s_{i,6}$ in another single subtree T_i^2, for each i.
3. The third segment p_3 consists of n pieces again. *Each* piece has length 8. The ith piece is 00001111, 11111111, 11110000, 00001111, 00000000, and 11110000, for $s_{i,1}, s_{i,2}, \ldots, s_{i,6}$, respectively. At the other positions of this segment, $s_{i,1}, s_{i,2}, s_{i,3}$ take 0 and $s_{i,4}, s_{i,5}, s_{i,6}$ take 1. This segment will be used to force T_i^1 and T_i^2 to be isomorphic in an optimal solution to I, as illustrated later.
4. The construction of the fourth segment p_4 is more sophisticated. p_4 contains m positions, corresponding to the m edges. For the jth position, if $e_j = \langle v_i, v_{i'} \rangle$ for some $i < i'$, then let it be 0, 1, 1, 0, 0, 1 for $s_{i,1}, s_{i,2}, \ldots, s_{i,6}$, respectively; if $e_j = \langle v_i, v_{i'} \rangle$ for some $i > i'$, then let it be 1, 0, 0, 1, 1, 0 for $s_{i,1}, s_{i,2}, \ldots, s_{i,6}$, respectively; if e_j is not adjacent to v_i, then let it be 0, 0, 0, 1, 1, 1 for $s_{i,1}, s_{i,2}, \ldots, s_{i,6}$, respectively. This segment attempts to develop the relation between the solutions to the instance of Max Cut and I.

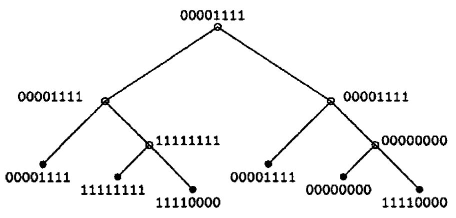
Let $N_1 = m^2n^2$ and $N_2 = mn$. A moment thinking shows that the attempted tasks of segments 1 and 2 listed in the above are achieved when setting N_1 and N_2 as those large values. The third segment will manage to do the following. Fig. 3 illustrates four possible topologies of T_i, which are named *left-style*, *right-style*, *inner-style* and *outer-style*, respectively. It is easy to verify that when T_i is either left-style or right-style, the costs contributed by the ith piece of segment p_3 in T_i is 16, which is 4 less than both inner-style and outer-style. Since for any index $j \neq i$, T_j^1 has the ith piece of p_3 be all 0's and T_j^2 has it be all 1's, when counting the costs contributed by the ith piece out of T_i, it does not matter if it is either 0 or 1 at the root of T_i. Thus, it is easy to check that the third segment will contribute 4 more to the cost if T_i is inner-style or outer-style. Consider the fourth segment p_4. Because the graph is cubic, it is not hard to verify that there are at most 3 positions which may contribute 1 less to the total cost when T_i is inner-style or outer-style. Thus, the third segment ensures T_i to be either left-style or right-style.

Now we come to the last segment. Till now, the topology of an optimal tree is almost fixed, except that the choice of the styles of T_i ($i = 1, 2, \ldots, n$), which is forced to be either left-style or right-style[6]. At the jth position of the last segment, suppose $e_j = \langle v_i, v_{i'} \rangle$. For any $k \notin \{i, i'\}$, $s_{k,1}, s_{k,2}, s_{k,3}$ all take 0 and $s_{k,4}, s_{k,5}, s_{k,6}$ all take 1. Therefore, the jth position contributes 1 for the cost in T_k, no matter what the root of T_k takes. Meanwhile, it is easy to verify that at T_i, $T_{i'}$ and the path linking T_i and $T_{i'}$, the jth position contributes 5 if both T_i and $T_{i'}$ are left-style (or right-style) (see Fig. 4(a)), and contributes 4 if they are in different styles (Fig. 4(b)). Thus, to reduce the cost, in an optimal solution, we must choose the styles of T_i's so that as many as possible pairs of adjacent vertices in G correspond to two trees with different styles.

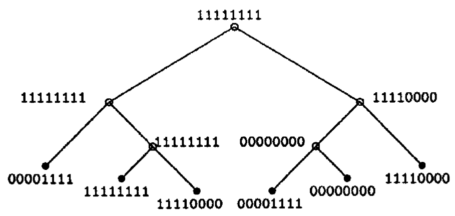
[6] Actually, the linkage of the T_i's is not fixed yet but it does not affect the total cost.



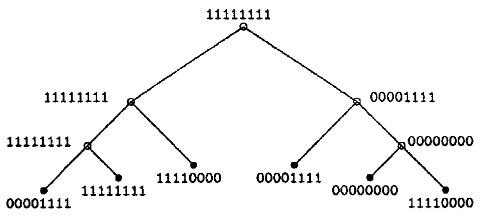
(a) left-style



(b) right-style

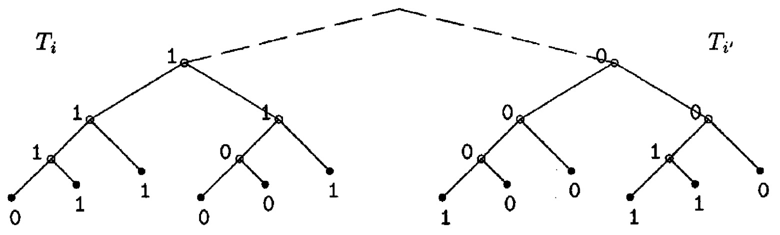


(c) inner-style

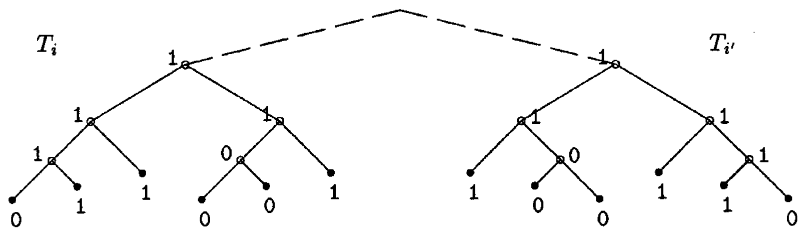


(d) outer-style

Fig. 3. Four possible topologies of the subtree T_i.



(a) T_i and $T_{i'}$ have same styles.



(b) T_i and $T_{i'}$ have different styles.

Fig. 4. Subtrees T_i and $T_{i'}$, and the path connecting them. The labels are the values at the positions corresponding to $e_j = \langle v_i, v_{i'} \rangle$, where $i < i'$.

It follows that if we respect the choice of the style of T_i to assign vertex v_i into either V_1 or V_2, the problem of finding a max-cut of graph G is reduced to find a minimum cost tree of I. This completes the proof. $\square$

4. PTAS for RTRH

The PTAS for the RTRH problem uses local optimization over subtrees of the duplication tree to efficiently approximate the minimal duplication tree. This approach is based upon the concept of an r-lifted tree which was introduced in the context of the *Tree Alignment* problem [16]. However, in the Tree Alignment application, the tree topology is known beforehand whereas here all that is known is the left to right ordering of the leaves.

Let T be a labeled tree. A node u of T is *lifted* from a descendant leaf v of u if v and u have the same sequence label. If a node is not lifted then it is called a *free* node. By default a leaf is a lifted node. The lifted nodes of T partition T into edge disjoint subtrees, which are called the *lifted components* of T. T is called a r-lifted tree if each lifted component of T has no more than r leaves (i.e. lifted nodes). Fig. 5 gives an example of a 3-lifted tree.

Suppose T is a labeled binary tree and $l^T(u)$ is the label of a node $u \in T$. For any distance $d(l^T(u), l^T(v))$ that satisfies *Triangular Inequality*, let $c'(T)$ be the sum of $d(l^T(u), l^T(v))$ for all pairs of adjacent nodes u and v. It is proved in [17] that there is an r-lifted tree T' with the same topology and leaf labels as T, so that $c'(T')$ is very close to $c'(T)$. More precisely, the following lemma is proved in [17]:

Lemma 2. *Let T be labeled binary tree. Assume $r = 2^{t-1} + 1 - q$ for $0 \leqslant q < 2^{t-2}$, then there is an r-lifted tree T' such that T' has the same topology and leaf labels, and satisfies that $c'(T') \leqslant (1 + \frac{2^{t-1}}{2^{t-2}(t+1)-q}) \times c'(T)$.*

In particular, if $r = 2^{t-1} + 1$, then $c'(T') \leqslant (1 + \frac{2}{t+1}) \times c'(T)$.

Moreover, for any positive integer r, a polynomial time algorithm was provided in [17] to construct an r-lifted tree satisfying Lemma 2. By Lemma 2, the algorithm is a PTAS for constructing a minimum-cost labeled tree under a given topology and given leaf labels. In their algorithm and proof, the authors used a special partitioning strategy to partition the given topology into lifted components. The partitioning strategy is an improvement from the one used in a previous paper [16], where the authors provided similar but weaker results. We note that the algorithms provided in [16,17] cannot be directly used here since their algorithms require the topology of the tree being given beforehand. Instead, we provide a new algorithm in this section to construct a minimum cost r-lifted tree under a given order of the leaves and given leaf labels. Since our algorithm does not rely on a specific partitioning strategy, we will be able to use Lemma 2 in the proof the approximation ratio.

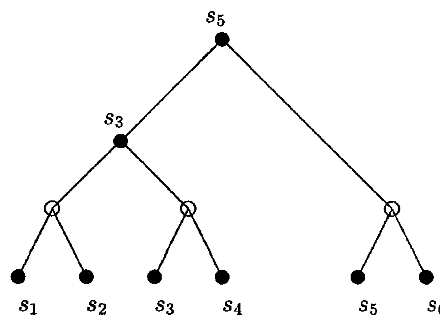


Fig. 5. A 3-lifted tree, where the solid nodes are lifted nodes since each of them has the same label as one of its descendent leaves.

Let $S = S_1S_2 \ldots S_n$ and m be an instance of the RTRH problem with minimum duplication tree T_{opt}. By Lemma 2, to obtain an approximation of T_{opt} it suffices to find a minimum cost r-lifted tree T where T becomes an arbitrarily good approximation of T_{opt} as r increases.

It turns out that for every r-lifted component with lifted nodes $v_1, v_2, \ldots, v_r$ where v_1 is the top lifted node, the optimal *local* duplication history can be constructed in polynomial time by trying all possible tree structures and using dynamic programming to assign the labels of the free nodes, as long as r is a constant. Denote the minimum cost of this component by $c(v_1, v_2, \ldots, v_r)$.

Therefore, it remains to be shown that all the lifted nodes of a minimum cost r-lifted tree can be obtained by dynamic programming. Define $D(i, k, j)$, where $i \leqslant k \leqslant j$, to be a minimum cost r-lifted tree such that

- the root of the tree is labeled by S_k and
- the leaves of the tree are labeled, from left to right, by $S_i, S_{i+1}, \ldots, S_j$.

It follows that the cost of a minimum cost r-lifted tree for $S = S_1S_2 \ldots S_n$ where the root is lifted is $\min_{1 \leqslant k \leqslant n} D(1, k, n)$. We note that a minimum cost r-lifted tree, where the root is not necessarily lifted, can be obtained by using the input $S' = S_0S_1S_2 \ldots S_n$ where S_0 is the sequence $\mathtt{XX} \cdots \mathtt{X}$ of length m, where $\mathtt{X}$ is a symbol not in $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}, -\}$. It is easy to see that there is a minimum cost r-lifted tree for S' where the left child of the root is the leaf labeled by S_0 and the root is lifted from S_0. Consequently, the cost of the right subtree of the root is the cost of a minimum cost r-lifted tree for S which is equal to

$$\min_{0 \leqslant k \leqslant n} D(0, k, n) - m.$$

Therefore, at the rest of this section, we concentrate on constructing the minimum cost r-lifted tree where the root is a lifted node.

Let T be a minimum cost r-lifted tree for $S_iS_{i+1} \cdots S_j$ where the root v is lifted from $S_k (i \leqslant k \leqslant j)$. It follows that T can be partitioned into two subtrees T_L and T_R that are rooted by v. The leaves $S_i, S_{i+1}, \ldots, S_{l-1}$ belong to T_L and $S_l, S_{l+1}, \ldots, S_j$ belong to T_R. See Fig. 6. Observe that the top lifted component of $T_L(T_R)$ may have $1 \leqslant t < r$ lifted nodes exclusive of v, denoted by $S_{k_1}, S_{k_2}, \ldots, S_{k_1}$, respectively. Each S_{k_l} leads a subtree with leaves $S_{p_{l-1}}, S_{p_{l-1}+1}, \ldots, S_{p_l}$, where the indices p_l's and k_l's satisfy

$$i = p_0 \leqslant k_1 < p_1 \leqslant k_2 < \cdots \leqslant k_t < p_t = l.$$

Obviously the cost of T is the sum of the costs of T_l and T_r. Since T is of the minimum cost, the labeling of nodes in T_l and T_r must be such that the cost of T_l and T_r are minimized. Observe that the cost of T_l consists of $c(S_k, S_{k_1}, S_{k_2}, \ldots, S_{k_t})$ and the costs of the subtrees led by $S_{k_1}, S_{k_2}, \ldots, S_{k_t}$, respectively. Let $f(i, S_k, l-1)$ be

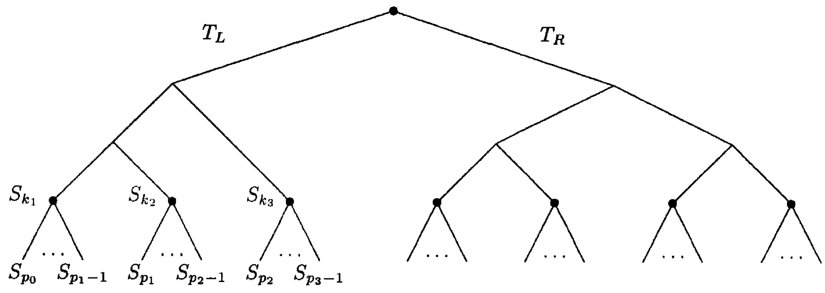


Fig. 6. The constituents of the cost of an r-lifted tree.

defined by the minimum of

$$c(S_k, S_{k_1}, S_{k_2}, \ldots, S_{k_t}) + \sum_{l=1}^{t} D(p_{l-1}, k_l, p_l - 1)$$

for all $2 \leqslant t < r$ and $i = p_0 \leqslant k_1 < p_1 \leqslant k_2 < \cdots < p_{t-1} \leqslant k_t < p_t = l$. Then $f(i, S_k, l-1)$ is the minimum cost of T_l. Similarly, $f(l, S_k, j)$ is the minimum cost of T_r.

The above observations justify the following recurrence for $D(i,k,j)$:

$$D(i,k,j) = \min_{i+1 \leqslant l \leqslant j} [f(i, l-1, S_k) + f(l, j, S_k)].$$

Dynamic programming can be used to compute the entries $D(i,k,j)$ for all $1 \leqslant i \leqslant k \leqslant j$. The detailed algorithm is given as follows:

```
                         ALGORITHM LIFTING
Input $S_1, S_2, \ldots, S_n \in \Sigma^m$, an integer $r$.
Output The minimum cost $r$-lifted tree with $S_1, S_2, \ldots, S_n$ as leaves.
1. for each $r$-element subset $\{ S_{i_1}, S_{i_2}, \ldots, S_{i_r} \}$ of the $n$ input strings do
                     Compute $c(S_{i_1}, S_{i_2}, \ldots, S_{i_r})$.
2. for $\Delta$ from 1 to $n$ do
        for $i$ from 1 to $n - k$ do
             for $k$ from $i$ to $i + \Delta$ do
                  (a) Let $j = i + \Delta - 1$.
                  (b) Let $D(i,k,j) = \min_{i+1 \leqslant l \leqslant j}[f(i, S_k, l-1) + f(l, S_k, j)]$.
3. Find the minimum $D(1,k,n)(1 \leqslant k \leqslant n)$. Reconstruct the $r$-lifted tree that has this
   minimum cost by backtracking.
```

Theorem 3. Algorithm Lifting *is a PTAS for* Restricted Tandem Repeat History.

Proof. Let $r \geqslant 3$ be a constant number. First let us estimate the time needed to compute $c(S_{i_1}, S_{i_2}, \ldots, S_{i_r})$. The number of different tree topologies with r leaves is also a constant number. If the distance between two labels is edit-distance, then for each topology, a dynamic programming can compute the optimal labels for all the internal nodes in $O(m^r)$ time [12]. Therefore, the time to compute $c(S_{i_1}, S_{i_2}, \ldots, S_{i_r})$ is bounded by $O(m^r)$. However, if the distance between two labels is Hamming-distance, then it is easy to see that we only need $O(m)$ time to compute $c(S_{i_1}, S_{i_2}, \ldots, S_{i_r})$. In any case, we only need polynomial time to compute $c(S_{i_1}, S_{i_2}, \ldots, S_{i_r})$. Since this time depends on which distance we use, we denote it by $T(r)$. Then step 1 of the algorithm needs $O(n^r T(r))$ time.

For any given $1 \leqslant i, l, k \leqslant n$, there are $O(n^{2r-3})$ different $(p_0, \ldots, p_t, k_1, \ldots, k_t)$ satisfying the conditions $2 \leqslant t < r$ and $i = p_0 \leqslant k_1 < p_1 \leqslant k_2 < \cdots < p_{t-1} \leqslant k_t < p_t = l$. Therefore, it takes $O(n^{2r-3})$ time to compute $f(i, S_k, l-1)$, provided that $c(S_k, S_{k_1}, S_{k_2}, \ldots, S_{k_t})$ and $D(p_{l-1}, k_l, p_l - 1)$ have been computed in the former steps. Similarly, it takes $O(n^{2r-3})$ time to compute $f(l, S_k, j)$. Thus, $DP(i,k,j)$ can be computed in $O(n \times n^{2r-3}) = O(n^{2r-2})$ time. Therefore, step 2 of the algorithm needs $O(n^3 \times n^{2r-2}) = O(n^{2r+1})$ time.

The backtracking needs no more time than the dynamic programming. Therefore, step 3 of the algorithm needs no more time than step 2. Totally, the algorithm needs $O(n^{2r+1} + n^r T(r))$ time.

From the discussion before the algorithm, ALGORITHM LIFTING outputs a minimum cost r-lifted tree T' with the leaf labels $S_1, S_2, \ldots, S_n$. By Lemma 2, this minimum cost r-lifted tree satisfies that

$$c(T') \leqslant \left(1 + \frac{2^{t-1}}{2^{t-2}(t+1) - q}\right) \times c(T_{\text{opt}}),$$

where $r = 2^{t-1} + 1 - q, 0 \leqslant q < 2^{t-2}$, and T_{opt} is the optimal solution for the instance $S_1, S_2, \ldots, S_n$ of RTRH. The ratio approaches 1 when r increases. Therefore, ALGORITHM LIFTING is a PTAS for RTRH. $\square$

Below we give a table of the approximation ratio and the running time for various values of t and r. In the table $T(r)$ denotes the time needed to compute $c(S_{i_1}, S_{i_2}, \ldots, S_{i_r})$.

r	3	4	5
t	2	3	3
Ratio	1.67	1.57	1.50
Time	$O(n^7 + n^3T(r))$	$O(n^9 + n^4T(r))$	$O(n^{11} + n^5T(r))$

Though the time complexity looks very high, in the next section it is demonstrated that the dynamic program can be implemented efficiently and applied to large datasets effectively. This is partly because our analysis of time complexity is not tight.

5. The PTAS applied to human genome data

In order to compare our PTAS designed for the RTRH problem to Benson and Dong's GREEDY-TRHIST-RESTRICTED algorithm, we implemented both algorithms and applied them to real examples of tandem repeats. We considered two sources of tandem repeat data. The first was a database of short tandem repeats maintained by the (American) National Institute of Standards and publicly available at http://www.cstl.nist.gov/div831/strbase/. However, we notice that this database focuses on short tandem repeat sequences which would limit the scope of the experimental study, and so, we choose not to use this source.

The second source of tandem repeats, that we elected to use in our experimental study, is the NCBI's Genbank. From Genbank we selected human chromosomes 1 and 22 which were completely sequenced at the time this experimental study was conducted. To the chromosomes we applied the publicly available tool TANDEM REPEAT FINDER [2] (http://c3.biomath.mssm.edu/trf.html) which extracted all tandem repeats from the raw chromosome data. For efficiency reasons, we extracted only those tandem repeats with 50 or less repeats and with pattern sequences of length no more than 100. As a result, 6016 tandem repeats were obtained from chromosome 1 and 3135 tandem repeats were obtained from chromosome 22.

The results of the experimental study appear in Figs. 7–9. In particular, for over 75% of the 6016 tandem repeats taken from chromosome 1 the PTAS recovers a less costly tandem repeat history than Benson's greedy algorithm whereas Benson's greedy algorithm produces a less costly tandem repeat history than the PTAS on less than 5% of the tandem repeats examined. For chromosome 22, the PTAS outperforms Benson's greedy algorithm on 80% of the 3135 tandem repeats

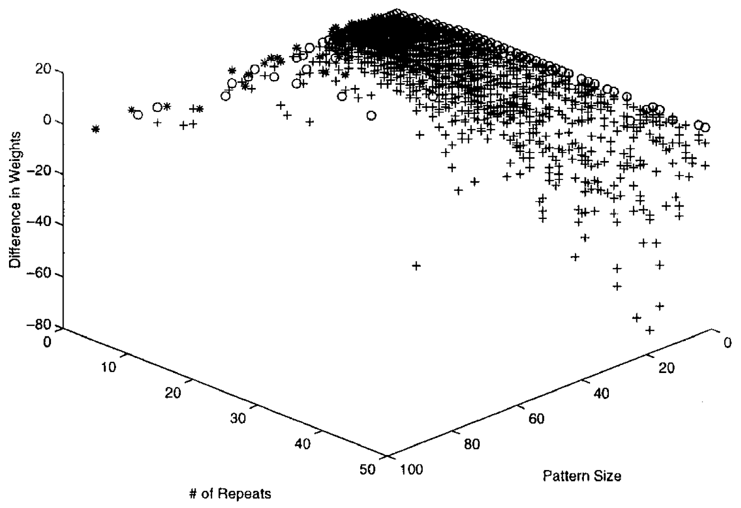


Fig. 7. Plot of difference between the PTAS and Greedy-TRHist-Restricted on human chromosome 1 versus the number and length of repeats (○: datapoints where difference is 0; +: datapoints where the PTAS produces a lower cost tandem repeat history; ∗: datapoints where Greedy-TRHist-Restricted produces a lower cost tandem repeat history).

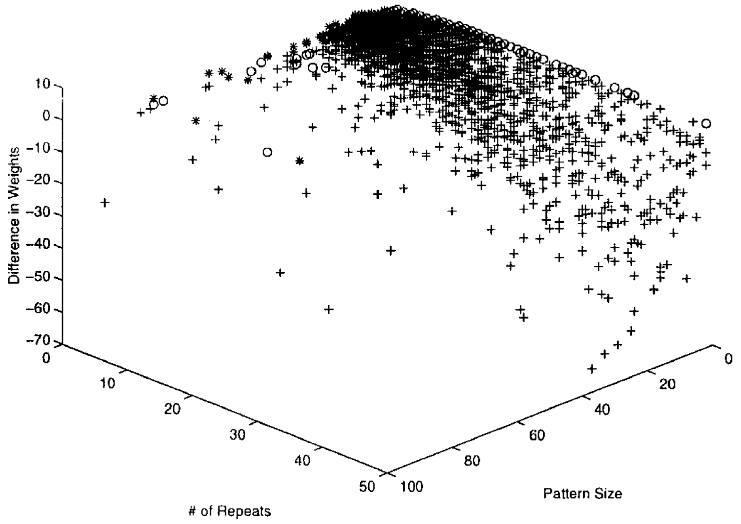


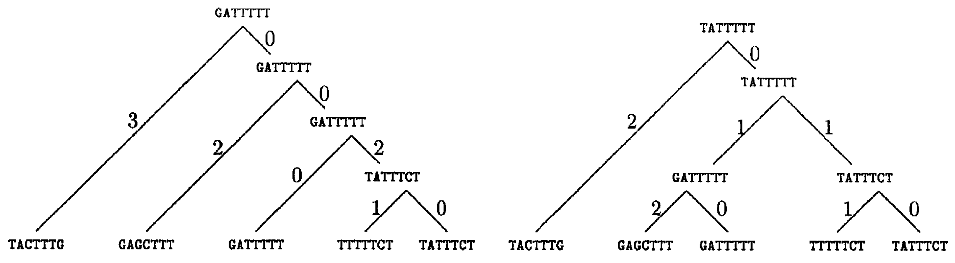
Fig. 8. Plot of difference between the PTAS and Greedy-TRHist-Restricted on human chromosome 22 versus the number and length of repeats (○: datapoints where difference is 0; +: datapoints where the PTAS produces a lower cost tandem repeat history; ∗: datapoints where Greedy-TRHist-Restricted produces a lower cost tandem repeat history).

whereas Benson's greedy algorithm outperforms the PTAS on less than 5% of the tandem repeats.

For illustration, an example of a tandem repeat where the PTAS produces a duplication tree of cost less than Greedy-TRHist-Restricted appears in Fig. 10.

Chromo.	Tot. Rpts.	−10%– 0%	0%	0%–10%	10%–20%	20%–30%	> 30%
1	6016	232	1235	1031	1725	1229	564
22	3135	131	492	605	965	690	252

Fig. 9. Summary of the differences between the PTAS and Benson's greedy algorithm on the chromosome 1 and 22 tandem repeats. Each row indicates on how many tandem repeats that PTAS reduces the costs by *x*–*y*% from the greedy algorithm.



(a) GREEDY tree, cost = 8. (b) PTAS tree, cost = 7.

Fig. 10. An example of a tandem repeat where the PTAS duplication tree is less costly than the GREEDY-TRHIST-RESTRICTED tree (8 versus 7).

6. Discussion

Although our PTAS is slower than GREEDY-TRHIST-RESTRICTED, it is sufficiently efficient to analyze over 9000 tandem repeats in human chromosomes 1 and 22 on a single CPU workstation when $r = 3$. In our experiments, we were able to perform the above analysis in 4 days on a machine running Linux, with 512 Mg RAM and a Pentium III CPU.

Our PTAS almost always produces tandem repeat histories with smaller cost than the tandem repeat histories produced by GREEDY-TRHIST-RESTRICTED. This result has two benefits. First, the PTAS provides a better upper bound for an instance to the unrestricted TRH problem than GREEDY-TRHIST-RESTRICTED (see [1] for details). Secondly, the PTAS can be used to do local optimization whenever the history constructed by other methods has a local component that is a tree structure. From Figs. 7 and 8 we can also see that the PTAS works better when the number of repeats is greater.

An interesting practical problem that arose during our experiments involved deciding how to compute distances between tandem repeat patterns. Since we were working with real data from Genbank our sequences had mutations. As a result the different repeat patterns in the tandem repeat sequences were not of exactly the same length as each other. This prevented us from using Benson and Dong's approach of comparing patterns character by character. We felt that a character by character comparison of repeat patterns might sacrifice local optimizations that would be possible by sequence alignment. As a result, we used sequence alignment with standard edit distance matrices to compute the distance between two patterns. To keep the experiments standard, our implementation of Benson's Greedy algorithm used the same metric to compute distances.

It is noteworthy that ALGORITHM LIFTING and the analysis do not depend on the edit-distance. Therefore, ALGORITHM LIFTING is also a PTAS for the minimum

Steiner tree problem when the order of the leaves is known. This improves the result in [16] where the authors proved that their algorithm is a PTAS for the minimum Steiner tree problem when the topology of the Steiner tree is known.

Acknowledgments

We thank Dr. Ming Li and Dr. Jonathan Badger for the valuable discussions on this topic. Paul Kearney thanks Gertrude Stein. Bin Ma's work was partially done at the University of Waterloo. Guohui Lin's work was partially done at the University of Waterloo and McMaster University.

References

[1] G. Benson, L. Dong, Reconstructing the duplication history of a tandem repeat, Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology (ISMB-99), 1999, pp. 44–53.
[2] G. Benson, Tandem repeats finder—a program to analyze DNA sequences, Nucleic Acids Res. 27 (1999) 573–580.
[3] M. Bern, P. Plassmann, The Steiner problem with edge lengths 1 and 2, Inform. Process. Lett. 32 (1989) 171–176.
[4] E. Eichler, H. Hammond, J. Macpherson, P. Ward, D. Nelson, Population survey of the human FRM1 CGG repeat substructure suggests biased polarity for the loss of AGG interruptions, Hum. Mol. Genet. 4 (1995) 2199–2208.
[5] A. Jeffreys, K. Tamaki, A. MacLeod, D. Monckton, D. Neil, J. Armour, Complex gene conversion events in germline mutation at human minisatellites, Nat. Gene. 6 (1994) 136–145.
[6] A.J. Jeffreys, V. Wilson, S.L. Thein, Individual-specific 'fingerprints' of human DNA, Nature 316 (1985) 76–79.
[7] A.J. Jeffreys, V. Wilson, S.L. Thein, Hypervariable 'minisatellite' regions in human DNA, Nature 314 (1985) 67–73.
[8] S. Kang, A. Jaworski, K. Ohshima, R. Wells, Expansion and deletion of CTG repeats from human disease genes are determined by the direction of replication in *E. coli.*, Nat. Genet. 10 (1995) 213–218.
[9] C. Kunst, S. Warren, Cryptic and polar variation of the fragile X repeat could result in predisposing normal alleles, Cell 77 (1994) 853–861.
[10] W.-H. Li, Molecular Evolution, Sinauer Associates, 1997.
[11] C.H. Papadimitriou, M. Yannakakis, Optimization, approximation, and complexity classes, J. Comput. System Sci. 43 (1991) 425–440.
[12] D. Sankoff, Minimal mutation trees of sequences, SIAM J. Appl. Math. 28 (1) (1975) 35–42.
[13] M. Tang, M.S. Waterman, S. Yooseph, Zinc finger gene clusters and tandem gene duplication, Proceedings of the Fifth Annual International Conference on Computational Molecular Biology, 2001, pp. 297–304.
[14] B. Widegren, U. Arnason, G. Akusjarvi, Characteristics of a conserved 1,579-bp highly repetitive component in the killer whale, Ornicus orca. Mol. Biol. Evol. 2 (1985) 411–419.
[15] A.H. Wyman, R. White, A highly polymorphic locus in human DNA, Proc. Natl. Acad. Sci. 77 (1980) 6745–6758.
[16] L. Wang, T. Jiang, E.L. Lawler, Approximation algorithms for tree alignment with a given phylogeny, Algorithmica 16 (1996) 302–315.
[17] L. Wang, T. Jiang, D. Gusfield, A more efficient approximation scheme for tree alignment, SIAM J. Comput. 30 (1) (2000) 283–299.