Reconstructing the Duplication History of a Tandem Repeat

Gary Benson[*] and **Lan Dong**[†]
Department of Biomathematical Sciences
The Mount Sinai School of Medicine
New York, NY 10029-6574

Abstract

One of the less well understood mutational transformations that act upon DNA is *tandem duplication*. In this process, a stretch of DNA is duplicated to produce two or more adjacent copies, resulting in a *tandem repeat*. Over time, the copies undergo additional mutations so that typically, multiple *approximate* tandem copies are present. An interesting feature of tandem repeats is that the duplicated copies are preserved together, making it possible to do "phylogenetic analysis" on a single sequence. This involves using the pattern of mutations among the copies to determine a minimal or a most likely history for the repeat. A history tries to describe the interwoven pattern of substitutions, indels, and duplication events in such a way as to minimize the number of identical mutations that arise independently. Because the copies are adjacent and ordered, the history problem can not be solved by standard phylogeny algorithms. In this paper, we introduce several versions of the tandem repeat history problem, develop algorithmic solutions and evaluate their performance. We also develop ways to visualize important features of a history with the goal of discovering properties of the duplication mechanism.

Keywords: tandem repeats, phylogeny algorithms

1 Introduction

One of the less well understood mutational processes for DNA molecules is *tandem duplication* in which a stretch of DNA is transformed into two or more adjacent *copies*. The following illustrates a tandem duplication in which the single occurrence of triplet CGG is transformed into three identical, adjacent copies.

$$...T\ \underline{CGG}\ A... \longrightarrow ...T\ \underline{CGG}\ \underline{CGG}\ \underline{CGG}\ A...$$

The result of a tandem duplication is termed a *tandem repeat*. Over time, individual copies within a tandem repeat may undergo additional, uncoordinated mutations (including new tandem duplications) so that typically, multiple *approximate* tandem copies are present.

Examination of a tandem repeat often suggests that the sequence was produced by a series of tandem duplications interspersed with point mutations. The real biological sequence shown in Fig. 1 is a typical example. It consists of 16 copies of an 8 nucleotide pattern. Copies are numbered and spaces inserted between the copies for clarity. A consensus for these 16 copies is $AAACTTAG$. Astrisks (*) flag differences between the copies and the consensus.

Careful observation reveals that G is periodically substituted for A. Such substitutions are unlikely to occur independently. It is more likely that a single common ancestor pattern is responsible for the A to G substitutions through duplication. Perhaps an 8 character unit, say $AAACTTAG$, was first duplicated and then mutated to $AGACTTAG$ and then the two copies were duplicated as a single 16 character unit $AAACTTAGAGACTTAG$. When the second through thirteenth copies are viewed in this way, 5 of the A to G substitutions are accounted for. Further observation suggests that the two starred Ts may have been the result of another duplication.

Tandem repeats are different from other types of duplicated sequences because the child copies of duplication are adjacent on the same sequence. This difference leads to complications in determining the parent copy of duplication. (See Fig. 2.)

Boundaries. It is not always possible to distinguish the boundaries of a duplicated pattern. Consider the two examples below in which a duplication changes three identical copies of $ABCD$ into four identical copies. Although the boundaries of the duplicated patterns (underlined) differ, the results are the same.

$$ABCD\underline{ABCD}ABCD \rightarrow ABCD\underline{ABCD}\ \underline{ABCD}ABCD$$
$$ABCDAB\underline{CDAB}CD \rightarrow ABCDAB\underline{CDAB}\ \underline{CDAB}CD$$

[*]Partially supported by NFS grant CCR-9623532 and a 1997 grant from the German Academic Exchange Service (DAAD).
[†]Partially supported by NFS grant CCR-9623532.

```
                 *   *                   *                   *
AAACTTAG  AAACTTAT  AGACTTAG  AAACTTAG  AGACTTAG  AAACTTAG  AGACTTAG  AAACTTAG
1         2         3         4         5         6         7         8
 *                *   *                   *         *   *                 *
AGACTTAG  AAACTTAT  AGACTTAG  AAACTTAG  AGACTTAG  AGACTCAG  AAACTTAG  AAAGCTTAG
9         10        11        12        13        14        15        16

--------------------------------------------------------------------------------

                 *
AAACTTAG  AAACTTATAGACTTAG  AAACTTAGAGACTTAG  AAACTTAGAGACTTAG  AAACTTAGAGACTTAG
1         2        3        4        5        6        7        8        9
       *                                  *   *                 *
AAACTTATAGACTTAG  AAACTTAGAGACTTAG  AGACTCAG  AAACTTAG  AAAGCTTAG
10       11       12       13       14       15       16
```

Figure 1: Top: Periodic nucleotide substitutions in a tandem repeat suggests a common ancestor. Bottom: Five of the A to G substitutions may be accounted for by a single A to G substitution followed by duplication.

Mutations add information. In the next example, the second copy of ABCD has been mutated to AXCY. Now, different duplication boundaries give different results.

$$ABCD\underline{AXCY}ABCD \rightarrow ABCD\underline{AXCY}\ \underline{AXCY}ABCD$$
$$ABCDAX\underline{CYAB}CD \rightarrow ABCDAX\underline{CYAB}\ \underline{CYAB}CD$$

Note that the boundaries are still not completely determined in the later two cases. The pattern in both could be shifted one character to the right and give the same results. We present a way to display this uncertainty in Section 6.

Duplication size. The size of the duplication unit can be any multiple of the basic pattern size. In the example below, four copies of a pattern of size 4 are changed into six copies by duplicating the middle 8 characters. Again, mutations in the original copies can help distinguish the size of the duplication unit from other possibilities.

$$ABCD\underline{AXCYABCZ}ABCD \longrightarrow ABCD\underline{AXCYABCZ}\ \underline{AXCYABCZ}ABCD$$

Several mechanisms have been proposed for the production of tandem repeats, including replication slippage and unequal crossing over (Wells 1996; Levinson & Gutman 1987; Schlotterer & Tautz 1992; Okumura, Kiyama, & Oishi 1987; Smith 1976). Biological studies (Strand *et al.* 1993; Weitzmann, Woodford, & Usdin 1997) have already provided support for one or the other of the mechanisms. Mathematical modeling has suggested mechanistic characteristics. For example in (Di Rienzo *et al.* 1994) accurate modeling of copy number variation at a polymorphic dinucleotide repeat locus has been obtained with a two-phase model which assumes predominantly single copy changes with rare multi-copy changes. In (Bell & Torney 1993) comparison of estimated rates of unequal crossing over and observed rates of microsatellite mutation lead to the conclusion that slipped strand mispairing is the major cause of length polymorphism in microsatellites. In (Charlesworth, Sniegowski, & Stephan 1994), modeling and simulation suggests that very low recombination rates (unequal crossing over) can result in very large copy number and higher order repeats.

Many unresolved questions can be asked about the mechanism of tandem duplication, among them: 1) Is the boundary of the duplication unit unique, is it confined to a few locations or is it seemingly unrestricted? 2) Is the duplication unit size unique, does it vary in a small range or is it unrestriced? Does pattern size affect the variability of duplication unit size? 3) Does duplication occur preferentially at one end or the other of the repeat or preferentially on the leading or lagging strand during replication (Kunst & Warren 1994; Kang *et al.* 1995; Eichler *et al.* 1995)?

Answers to these questions may suggest the presence of conformational structures, either within or adjacent to the tandem repeat (Jeffreys *et al.* 1994), which trigger duplication or may indicate that different mechanisms act on patterns of different sizes. An extensive analysis of the histories of many tandem repeats can provide data to support one or the other of the theoretical models and may reveal new mechanistic features not already anticipated. Additionally, comparison of related tandem repeats in different sequences could resolve important questions regarding evolution or mutation over short time scales. Such a capability would open up new opportunities to address questions of evolution and ancestry, including the study of human migration, rapid evolution of bacterial diseases, and the cascade of mutations that lead to cancer. With these purposes in mind, we have begun the development of algorithms to

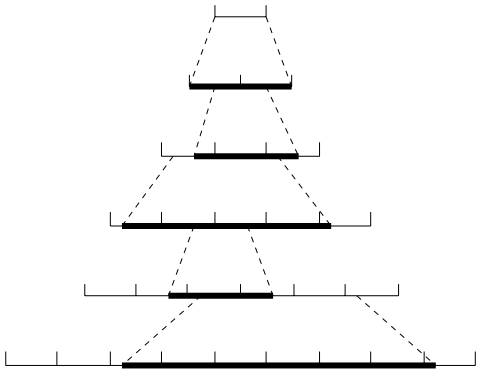

Figure 2: A tandem repeat history. Ancestral pattern sequence is at the top. Bottom sequence contains 9 descendant copies of the pattern. Dotted lines mark the boundaries of copies involved in a duplication. Parent copy is above, child copies (in bold) below. Note that 1) the boundaries of a parent need not coincide with the putative boundaries of the pattern, 2) a parent's length can be a multiple of the length of a single pattern, and 3) child copies can interact to form a parent in subsequent duplications.

reconstruct tandem repeat histories.

The remainder of this paper is organized as follows. Section 2 contains definitions and descriptions of the problems we investigate. Section 3 describes our greedy algorithms for the history problem. In Section 4 we develop upper and lower bounds on a restricted version of the history problem. In Section 5 we report the performance of the algorithms on simulated sequences. Finally, in Section 6 we give graphical presentations of our analysis of real biological sequences. The Appendix contains additional details on one of our algorithms.

2 Definitions and Problem Descriptions

For the purposes of the problems described below, we assume that a tandem repeat sequence consists of n approximate copies of a basic pattern of length k. We are given a multiple alignment, M, of the copies. M has n rows and k columns and the ith row in M contains the ith copy (left-to-right) in the tandem repeat. We let $M_{i,j}$ represent the ith row and jth column of M. Each $M_{i,j}$ contains one of the alphabet symbols $\{A, C, G, T, -\}$ where $-$ indicates a gap in the alignment. We use the notation

$$M_{i,j} \cdots M_{i',j'}, \quad 1 \leq i \leq i' \leq n, \quad 1 \leq j \leq j' \leq k$$

to represent a substring of characters in the multiple alignment starting at position (i, j), ending at position (i', j'), and wrapping around at the right edge of the multiple alignment if necessary.

Definition 1. A *pattern* is some string of nucleotides. A *tandem duplication* is a mutation that replaces one copy of a pattern with two or more adjacent, identical copies. A *contraction* is an algorithmic operation in which two or more adjacent, equal length substrings (the contraction copies) of a string are replaced by a single substring (the *merged copy*). A contraction can be thought of as the opposite of a tandem duplication. A *binary contraction* replaces *two* contraction copies with a merged copy. A *many-to-one contraction* replaces *two or more* contraction copies with a merged copy. A *contraction copy* is some substring of the multiple alignment M with length a multiple of k. For the purposes of contraction, each position in M is treated as a *character set* which is some subset of the alphabet $\{A, C, G, T, -\}$. An *ambiguous character set* is a character set which is not a singleton set, *e.g.*, $\{A, G\}$ is ambiguous but $\{A\}$ is not. The original multiple alignment M contains no ambiguous character sets, but a merged copy may contain ambiguous character sets. When a contraction is applied to a multiple alignment M, a new, shorter multiple alignment M' is produced.

In this paper, we consider the following problems.

- **Tandem repeat history problem (TRHIST).** Given a multiple alignment M of the copies of a tandem repeat, a cost function for contractions, and a rule for producing merged copies, find a least cost series of contractions which reduce M to a single merged copy.
- **TRHIST, fixed boundary, fixed duplication size.** Size and boundaries of contraction copies are fixed and remain the same across all contractions. Without loss of generality, the size is k and the left boundary is column 1 of M.
- **TRHIST, single column, fixed duplication size.** The history problem on a single column of M. Boundary is necessarily fixed and size of contraction copies is fixed without loss of generality at a single character.

3 Greedy algorithms for TRHIST.

Rule for producing merged copies. If contraction copies are not identical, the merged copy will contain ambiguous characters, represented by ambiguous character sets. This ambiguity may be resolved by some later contraction. Our rule is that the character set at position i in a merged copy is the intersection of the character sets at position i in the contraction copies if the intersection is non-empty. Otherwise, it is the union of the character sets. This is analogous to the method used by Sankoff (Sankoff 1975; Sankoff & Rousseau 1975).

The cost of a contraction. We let the cost function for contractions equal the number of changes that must be made in the contraction copies to make them identical. This is an edit distance type cost function where

First contraction:

	1	2	3	4	5		1	2	3	4	5
1	A	C	T	T	A		1	A	C	T	T	A
2	A	C	–	T	A		2	A	C	–	T	A
3	G	G	< T	T	A		3	G	G	T	T	{A/C}
4	G	A	T	T	A	$\Longrightarrow$	4	{A/G}	{A/C}	T	T	A
5	G	A	>< T	T	C		5	G	{A/C}	T	T	A
6	A	C	T	T	A							
7	G	C	> T	T	A							

$M_{3,3} \cdots M_{5,2}$ =	T	T	A	G	A	T	T	A	G	A
$M_{5,3} \cdots M_{7,2}$ =	T	T	C	A	C	T	T	A	G	C
merged copy =	T	T	{A/C}	{A/G}	{A/C}	T	T	A	G	{A/C}

Second contraction:

	1	2	3	4	5		1	2	3	4	5
1	A	C	T	T	A		1	A	C	T	T	A
2	A	C	– <	T	A		2	A	C	–	T	A
3	G	G	T ><	T	{A/C}	$\Longrightarrow$	3	G	{A/C/G}	T	T	A
4	{A/G}	{A/C}	T >	T	A		4	G	{A/C}	T	T	A
5	G	{A/C}	T	T	A							

$M'_{2,4} \cdots M'_{3,3}$ =	T	A	G	G	T
$M'_{3,4} \cdots M'_{4,3}$ =	T	{A/C}	{A/G}	{A/C}	T
merged copy =	T	A	G	{A/C/G}	T

Figure 3: An example of two binary contractions.

substitutions and indels have equal cost. The cost of a contraction equals the number of character sets that are formed in the merged copy by the union operation. To see why, note that in the case where the intersection is not empty, there is a character which makes the contraction copies identical at that position. In the union case though, there is no character that both contraction copies share and therefore, at least one of the copies must be changed at that position with a cost of one.

The operation of binary contraction is illustrated in Fig. 3. Many-to-one contraction works similarly. On the left in the first contraction, is a multiple alignment with $k = 5$ and $n = 7$. Two contraction copies, of length $2k$, are marked by $<$ and $>$. On the right is the new alignment with the merged copy. The contraction copies and merged copy are shown separately below the alignments. Braces indicate ambiguous character sets. The contraction cost is 4. In the second contraction, the contraction copies have size 1k. In the merged copy, two ambiguous character sets are eliminated and one set grows larger. The contraction cost is 1.

A binary contractions algorithm. Our first greedy algorithm, GREEDY-TRHIST, locally minimizes the binary contraction cost. Each contraction removes two contraction copies from a multiple alignment and replaces them with a merged copy to form a smaller multiple alignment. At each stage, the algorithm chooses the contraction with minimum *contraction cost ratio* defined as the contraction cost divided by Δ_c which is the reduction in size of the tandem repeat. (Here Δ_c equals the size of one contraction copy). Ties are broken arbitrarily except that larger Δ_c is chosen over smaller Δ_c.

We have implemented GREEDY-TRHIST as a $O(kn^3)$ algorithm. Note that the problem size is kn. At each stage, the cost for every possible contraction (size $1 \cdot k$ to size $(n/2) \cdot k$, starting at every position) is determined with a character to character comparison. This takes time $O(kn^2)$. There are at most $n-1$ contraction stages. Notice that it is possible to leave out of the calculation any columns that contain only a single letter. The number of such columns increases as the algorithm proceeds.

A many-to-one contractions algorithm. Our second greedy algorithm GREEDY-MANY-TRHIST locally minimizes the many-to-one contraction cost. Each contraction removes $k \geq 2$ contraction copies from a multiple alignment and replaces them with a single merged copy. At each stage, the algorithm chooses the contraction with minimum contraction cost ratio. (Here, Δ_c is $k-1$ times the size of a contraction copy.) Ties are broken as in GREEDY-TRHIST.

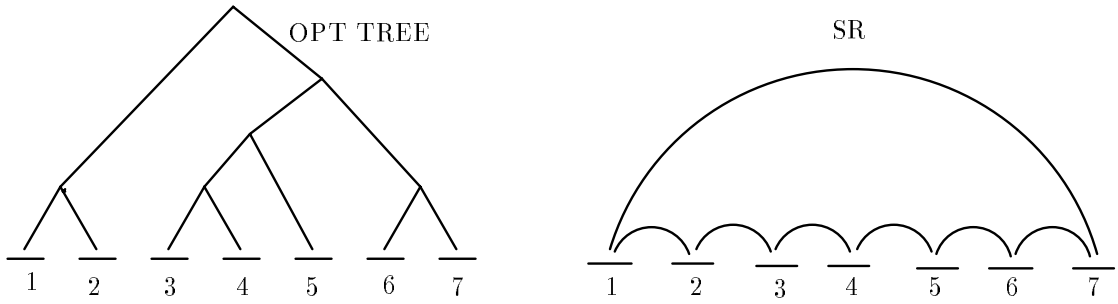

Figure 4: An optimal duplication tree and the cycle SR produced by shortcutting an inorder traversal of the tree.

GREEDY-
MANY-TRHIST is implemented as a $O(kn^3 \log n)$ algorithm. Within a single column, for contraction size $i \cdot k, i = 1, 2, \ldots, n/2$, there are $O(n^2/i)$ costs to be determined, each in constant time using earlier cost calculations. This leads to $O(n^2 \log n)$ cost calculations. For k columns and a maximum of $n-1$ contraction stages, the total is $O(kn^3 \log n)$. We do not report further on GREEDY-MANY-TRHIST in this paper.

Exploring the tree of solutions. The space of all possible history solutions for a tandem repeat can be explored as a tree of solutions in which we are seeking the minimal solution. The GREEDY algorithms follow only a single branch of this tree at each node (*i.e.*, only a single contraction is selected). In order to improve the chance of finding an optimal solution, we do a limited exploration of the tree of solutions. At each contraction stage, we generate a list of minimal cost (or near minimal) contraction choices (there are often several minimal cost choices) and using depth first search we explore each choice in turn.

Exploration of the solution tree provides a secondary benefit. It allows us to identify those features (boundaries/duplication sizes/duplication positions) that are strongly supported by the collection of minimal or near minimal histories.

4 Upper and lower bounds on the cost of a restricted problem.

It is difficult to evaluate the GREEDY-TRHIST algorithms' ability to find minimum cost solutions to a history problem because the minimum answer is not known. In order to test the method, we have used simulated data and a more restricted problem in which the boundaries and duplication sizes are fixed ahead of time. A greedy solution for the fixed boundary, fixed duplication size problem can be obtained with the algorithm GREEDY-TRHIST, in time $O(n^2 k)$, by restricting the chosen contractions to those with left boundary in column 1 of M and contraction copy size $= 1k$. We call this algorithm GREEDY-TRHIST-RESTRICTED.

Even with a restricted version of the history problem, we still do not know the minimum answer. Below, we develop several upper and lower bounds with which we compare the performance of the GREEDY algorithm.

4.1 Upper bounds

In contrast to the general problem, the duplication history of the restricted problem is always a tree. As with other Steiner tree problems which obey the triangle inequality, the fixed boundary, fixed duplication size problem can be bounded to within 2 times optimal (Kou, Markowsky, & Berman 1981). Unlike those other problems, a minimum spanning tree is *not* required. A minimum spanning tree will usually improve the $2 \cdot OPT$ solution, but because the leaves of the tree are ordered, a special type of minimum spanning tree, the *ordered minimum spanning tree* is required. Due to the the left-to-right ordering of the pattern copies imposed by the tandem repeat sequence any other Steiner tree approximation algorithm which depends in its proof or implementation on unordered trees does not apply. An example is the 11/6 Steiner tree approximation of Zelikovsky (Zelikovsky 1993) which assumes that edges can be removed and added on spanning trees whose leaves are unordered.

Definition 2 A *duplication tree* is a rooted, leaf and edge ordered tree. A depth-first traversal of the tree which follows the edge order at each node visits the leaves in order (Fig. 4, left). An *ordered spanning tree* is a spanning tree on an ordered set of nodes with the following property. With the nodes numbered in order, for any two edges (i_1, j_1) and $(i_2, j_2), i_1 < j_1, i_2 < j_2$ we have $(i_1 - i_2)(i_1 - j_2)(j_1 - i_2)(j_1 - j_2) \geq 0$. Alternately, an ordered spanning tree can be drawn on an ordered set of nodes arranged in a linear fashion, with every edge occupying the same half plane (the half planes established by the line through the nodes) and with no edges crossing (Fig. 5, left). Each tree is built on a multiple alignment with k columns. The ith leaf (duplication tree) or ith node (ordered spanning tree) is labeled with the ith row of M. In a duplication tree, the internal nodes are labeled with ancestor sequences also of length k. *Edge cost* in both types of tree is the

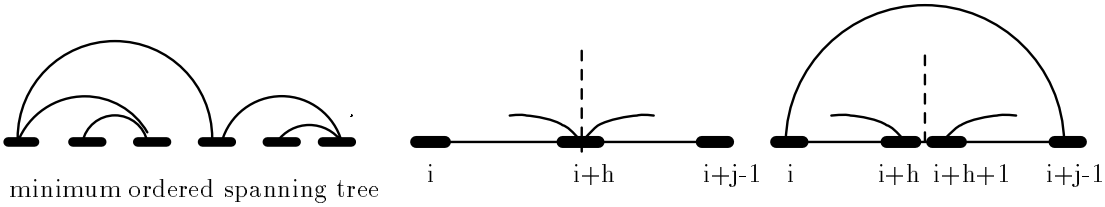

Figure 5: An ordered spanning tree, left. The recursion for minimal cost for an interval of length j starting at node i has two cases.

number of differences between the aligned substrings labeling the ends of the edge. The *cost of a tree* is the sum of its edge costs.

Lemma 1. Every ordered spanning tree can be transformed into a duplication tree of equal cost.

Proof: Sketch: To convert an ordered spanning tree to a duplication tree, we create a leaf for each original node in the spanning tree. We also create a root and internal nodes each of which has the same label as one of the leaves. New edges created either have a cost of zero because they connect nodes with the same label or they mimic the edges in the spanning tree. The ordering of the edges at each node preserves the ordering of the leaves. ∎

A $2 \cdot OPT$ approximation. An optimal duplication tree P for M will look something like the tree shown in Fig. 4. Each leaf is one of the rows of M. Intermediate nodes, are labeled with ancestral sequences. An inorder traversal of P starting at the root produce a cycle R of nodes in which each internal node appears twice and each leaf node appears once. Because R has two edges for every edge in P, $cost(R) = 2 \cdot cost(P)$. Eliminating all the internal nodes from R by short-cutting between leaf nodes produces a simple cycle SR containing only leaf nodes. It is important to note that *no matter what the original form of P, the graph SR always has the same form* shown in Fig. 4. The cost of SR depends only on the distance between the leaf nodes, that is, the rows in M. The triangle inequality guarantees that $cost(SR) \leq cost(R)$. Following the nodes in SR from leaf 1 produces the sequence $1, 2, 3, \ldots, n, 1$. By removing the most costly edge, an ordered spanning tree TR is produced, giving the inequality

$$cost(TR) \leq cost(SR) \leq cost(R) = 2 \cdot cost(P).$$

TR can be easily transformed into a duplication tree TR^* of equal cost (Lemma 4.1). Thus TR^* is a solution with cost no greater than $2 \cdot OPT$.

The minimum ordered spanning tree. TR is not necessarily a *minimum* ordered spanning tree. A true minimum tree MT can be computed using dynamic programming in $O(n^2k + n^3)$ time. We compute the minimum ordered spanning tree for all intervals of size $2, 3, \ldots, n$ where an interval of length j starting at node i contains the $i, \ldots, i + j - 1$ rows in M. For an interval of length 2, the cost of the edge is the distance $d(i, i + 1)$ between the copies. For an interval of length $j > 2$ starting at node i, there are two possible cases for the minimal cost (Fig. 5). In one case, a node $i + h$ splits the interval, with all nodes on the left side of the split connected to all nodes on the right side through node $i + h$. In the other case, a node $i + h$ splits the interval with all nodes on the left, including node $i + h$, connected to all nodes on the right through an edge between nodes i and $i + j - 1$. The recurrence for the cost is

$$DSP(i,j) = \min \begin{cases} \min\limits_{1 \leq h \leq j-2} \{DSP(i, h+1) + DSP(i+h, j-h)\} \\ \min\limits_{0 \leq h \leq j-2} \{d(i, i+j-1) + DSP(i, h+1) \\ \qquad + DSP(i+h+1, j-h-1)\}. \end{cases}$$

The minimum ordered spanning tree MT can be converted into a duplication tree MT^* of equal cost (Lemma 4.1) and this tree while no worse than TR^* is usually better (see section 5). As we also report in section 5, GREEDY-TRHIST-RESTRICTED produces a much better solution than either TR^* or MT^*.

4.2 Lower bounds

Our crudest lower bound is **character differences**, $\sum_j C_j - 1$, where C_j is the number of different characters in the column j of M. This bound implies that every pair of identical characters in a column can be merged at zero cost. Better bounds are possible for the fixed boundary, fixed duplication size problem. First, from the $2 \cdot OPT$ solution, $cost(SR)$ is easy to compute, so we have a simple lower bound of

$$cost(P) \geq cost(SR)/2.$$

Next, observe that in the restricted problem, the duplication tree for each column of the multiple alignment M is identical. A single column algorithm, when applied to each column separately provides a lower bound, referred to as **independent columns**. The minimum cost for a single column of n characters can be found in $O(n^3)$ time by dynamic programming. The algorithm is given in the Appendix. A better lower bound can be obtained by computing the optimal cost for every

Length=60, Duplications=10

Cost Differences	$p = .02$		$p = .03$		$p = .04$	
	MEAN	STD	MEAN	STD	MEAN	STD
$TR^* - MT^*$	6.8	3.6	7.6	4.0	8.2	4.0
$MT^* - GREEDY$	16.9	4.4	24.7	4.7	31.2	5.2
$GREEDY - ColPair$	1.0	1.3	1.3	1.4	1.3	1.5
Generating Cost	53.3	6.4	79.6	9.1	105.7	10.7

Table 1: Cost differences between the three solution methods, TR^*, MT^*, and GREEDY-TRHIST-RESTRICTED and the best lower bound, column pairs. Generating cost is the number of character changes during "mutation" in the simulation. p is the probability of character mutation between duplications. GREEDY-TRHIST-RESTRICTED surpasses MT^* by about 30% relative to the generating cost and is very close to the lower bound.

subset of columns of size r and then greedily choosing subsets whose joint column costs most exceed their independent costs. We have used this method, referred to as **column pairs** with $r = 2$. The algorithm for a single column can be generalized to sets of r columns in time $O(5^r n^3)$.

5 Simulation results

In our simulation tests, we show that the GREEDY algorithms perform very close to our best lower bounds and that GREEDY-TRHIST-RESTRICTED is much better than the algorithms based on duplication trees. Note that for the results presented here we did not explore the solution tree as described in section 3.

Each simulation sequence started with a single randomly generated string of length k ($k = 60$ for the results presented here, $k = 12, 25$ not shown, but similar). For the first duplication, the entire string was duplicated. In all subsequent duplications, a substring of length k was chosen and duplicated. Every duplication, except the first was preceded by "mutation" in which every character in the sequence could change to another character with probability p. Three values of p were used, .02, .03, and .04. We chose these mutation rates to bracket 1) the contraction cost for a real 60bp pattern found in the human β T cell locus sequence (not shown) and 2) the estimated average mutation rate between contractions observed in several real examples we analyzed, including the 60bp pattern and the 135bp patterns shown in Fig. 7. The number of character changes during mutation for one sequence is the *generating cost*. A history for each simulated sequence was constructed and the cost calculated. In the figures, costs are normalized by subtracting the generating cost.

TRHIST fixed boundary, fixed duplication size. For this problem, the left boundary of the duplicated substring was always at the first position of a copy (corresponding to column 1 of the multiple alignment). The best lower bound for this problem is column pairs. Of the three solution methods (Section 4), GREEDY-TRHIST-RESTRICTED is superior to the other two, surpassing MT^*, the next best method, by about 30% relative to the generating cost and giving solutions which are very close to the lower bound (Table 1, Fig. 6, left).

Unrestricted TRHIST problem. For this problem, the left boundary of the duplicated substring was unrestricted, *i.e.* it could have occurred in any column of the multiple alignment. GREEDY-TRHIST follows the same pattern of performance as GREEDY-TRHIST-RESTRICTED. The only lower bound that applies here is character differences which is not as accurate as column pairs is for the the restricted problem (Fig. 6, right).

6 Data visualization

Recall from the discussion in the introduction that there can be uncertainty in the boundaries of the duplicated pattern. Fig. 7, top, is a graphical display of this uncertainty. The circles represent contractions produced by GREEDY-TRHIST on two distinct tandem repeats of a 135bp pattern containing 18 copies (left) and 14 copies (right) from chromosome 1 in yeast. Each ring represents one contraction. The shaded arc shows the possible left boundaries (columns in the multiple alignment M) of the contraction copies. The *rows* of the contraction copies are indicated in the bottom of the figure (see below). Reading clockwise from column 1 to column 135, *any left boundary* chosen from the shaded arc *will give the same sequence after contraction*. For example, in the left circle, second ring from the outside, the left boundary can be any of columns 88 to 103. Gray level signifies contraction cost.

Fig. 7, bottom, is a graphical display of the location of contractions in the histories. Numbers on the right indicate size of the contraction copy ($1 = 1k$, $2 = 2k$). The rightmost possible boundaries for each contraction are shown. Notice that the history on the right is nearly identical to the history on the left except for the four contractions marked with an asterisk (*) and some minor reordering of the contractions. Indeed, the rightmost 12 copies in the sequence on the left are nearly identical to the leftmost 12 copies in the sequence on

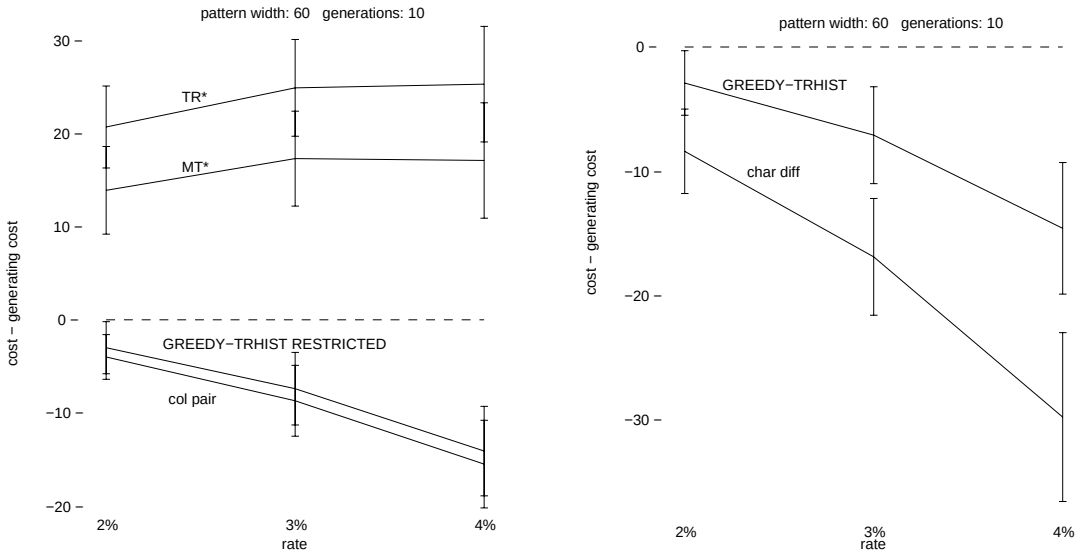

Figure 6: Comparison of lower bound and solutions for TRHIST problems. Results are from 250 simulated sequences at each of three mutation rates. Scores were normalized by subtracting the generating cost. Left, fixed boundary, fixed duplication size problem. Col pair is best lower bound on solution. GREEDY-TRHIST-RESTRICTED solutions are very close to the lower bound and are much better than TR^* or MT^*. See Table 1. Right, unrestricted TRHIST problem. Note that char diff lower bound is less accurate than that used on left. Generating costs for both graphs are similar.

the right. The sequence on the right has somehow lost an older part of the tandem repeat.

Both histories show a bias in the position of the duplication boundaries. We are currently analyzing simulations and other histories to determine if these biases are significant.

References

Bell, G., and Torney, D. 1993. Repetitive DNA sequences: some considerations for simple sequence repeats. *Computers Chem.* 17:185–190.

Charlesworth, B.; Sniegowski, P.; and Stephan, W. 1994. The evolutionary dynamics of repetitive DNA in eukaryotes. *Nature* 371:215–220.

Di Rienzo, A.; Peterson, A.; Garza, J.; Valdes, A.; Slatkin, M.; and Freimer, N. 1994. Mutational processes of simple-sequence repeat loci in human populations. *Proc. Natl. Acad. Sci. USA* 91:3166–3170.

Eichler, E.; Hammond, H.; Macpherson, J.; Ward, P.; and Nelson, D. 1995. Population survey of the human FMR1 CGG repeat substructure suggests biased polarity for the loss of AGG interruptions. *Human Molecular Genetics* 4:2199–2208.

Fitch, W. 1971. Toward defining the course of evolution: minimum change for a specific tree topology. *Syst. Zool.* 20:406–416.

Jeffreys, A.; Tamaki, K.; MacLeod, A.; Monckton, D.; Neil, D.; and Armour, J. 1994. Complex gene conversion events in germline mutation at human minisatellites. *Nature Genetics* 6:136–145.

Kang, S.; Jaworski, A.; Ohshima, K.; and Wells, R. 1995. Expansion and deletion of CTG repeats from human disease genes are determined by the direction of replication in e. coli. *Nature Genetics* 10:213–218.

Kou, L.; Markowsky, G.; and Berman, L. 1981. A fast algorithm for the Steiner problem in graphs. *Acta Inform.* 15:141–145.

Kunst, C., and Warren, S. 1994. Cryptic and polar variation of the fragile x repeat could result in predisposing normal alleles. *Cell* 77:853–861.

Levinson, G., and Gutman, G. 1987. Slipped-strand mispairing: a major mechanism for DNA sequence evolution. *Molec. Biol. Evol.* 4:203–221.

Okumura, K.; Kiyama, R.; and Oishi, M. 1987. Sequence analyses of extrachromosomal *sau3a* and related family DNA: analysis of recombination in the excision event. *Nucleic Acids Res.* 15:7477–7489.

Sankoff, D., and Rousseau, P. 1975. Locating the vertices of a Steiner tree in an arbitrary metric space. *Mathematical Programming.* 9:240–246.

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

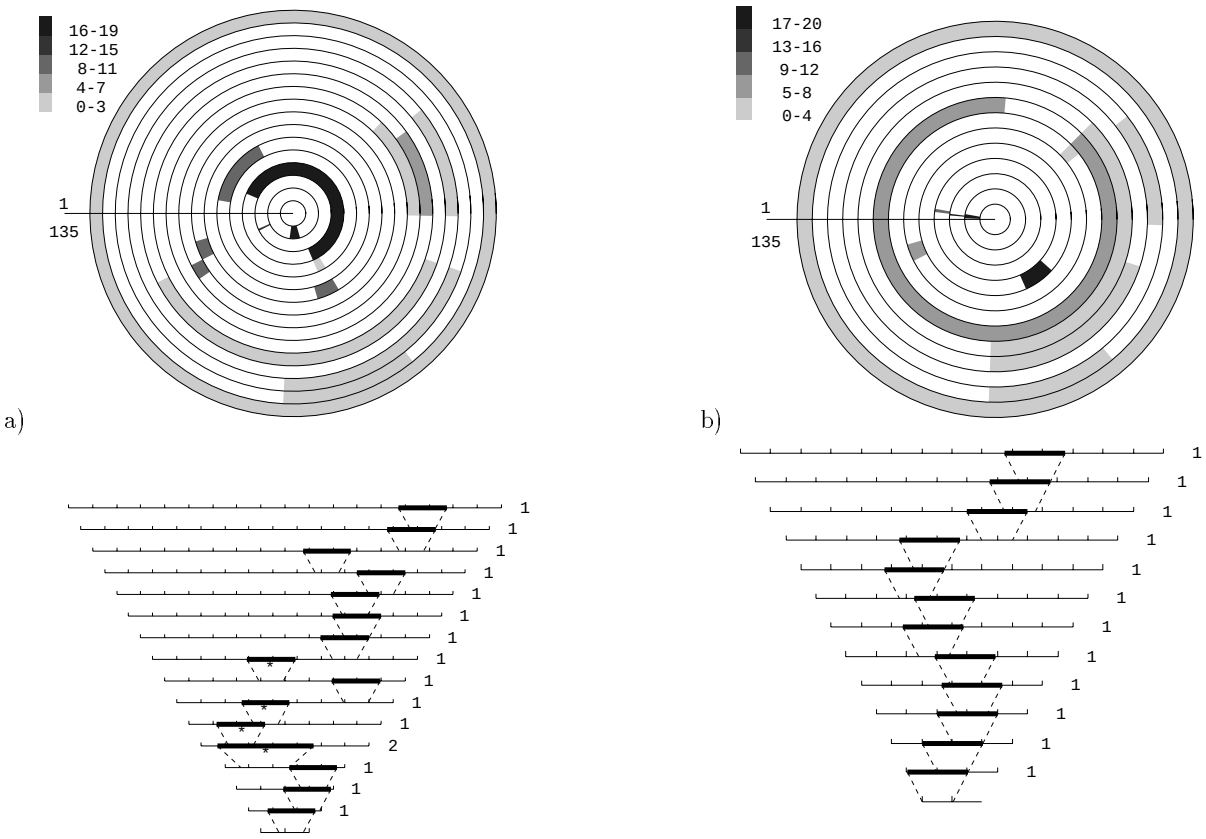

Figure 7: Graphical representation of duplication histories, a) 18 copies of a 135bp pattern and, b) 14 copies of a 135bp pattern, both from yeast chromosome 1. Each ring represents one contraction. Horizontal line in circle marks the first and last columns in multiple alignment. Shaded arcs indicate possible left boundary of duplication unit. Gray level signifies contraction cost. Triangular display shows position of contractions. Rightmost possible contraction boundaries are shown. Numbers on right indicate size of contraction copy ($1 = 1k$, $2 = 2k$).

Schlotterer, C., and Tautz, D. 1992. Slippage synthesis of simple sequence DNA. *Nucleic Acids Res.* 20:211–215.

Smith, G. 1976. Evolution of repeated DNA sequences by unequal crossover. *Science* 191:528–535.

Strand, M.; Prolla, T.; Liskay, R.; and Petes, T. 1993. Destabilization of tracts of simple repetitive DNA in yeast by mutations affecting DNA mismatch repair. *Nature* 365:274–276.

Weitzmann, M.; Woodford, K.; and Usdin, K. 1997. Dna secondary structures and the evolution of hypervariable tandem arrays. *J. of Biological Chemistry* 272:9517–9523.

Wells, R. 1996. Molecular basis of genetic instability of triplet repeats. *J. of Biological Chemistry* 271:2875–2878.

Zelikovsky, A. 1993. An 11/6-approximation algorithm for the network steiner problem. *Algorithmica* 9:463–470.

7 Appendix

Single column minimum algorithm. We are given a column of characters from the alphabet $\Sigma = \{A, C, G, T, -\}$ on which to construct a duplication history tree. *If* a history tree were given, and we wanted to compute its cost, we would use a method first described by Fitch (Fitch 1971) and later proven correct by Sankoff (Sankoff 1975; Sankoff & Rousseau 1975). The method uses a cost vector C_i of size $|\sigma|$ assigned to each node x_i of the tree. Each entry $C_i[\sigma], \sigma \in \Sigma$, represents the minimum cost of the edges in the subtree rooted at x_i including the edge to the parent of x_i when the parent is labeled with the character σ. (The root can be assumed to have a hypothetical parent.) The recurrence for C_i is as follows:

$$C_i[\sigma] = \begin{cases} \begin{cases} 0 & \sigma = \rho_i \\ 1 & \sigma \neq \rho_i \end{cases} & \text{when leaf } x_i \text{ is labeled } \rho_i. \\ \min\limits_{\rho \in \Sigma}(C_{j_1}[\rho] + \ldots + C_{j_k}[\rho] + d(\sigma, \rho)) & \\ \qquad \text{when } x_i \text{ has children } x_{j_1}, \ldots, x_{j_k}. \end{cases}$$

Single column algorithm

Data structures:
$V[substring\_size, substring\_start, letter]$ =cost of subtree when root is letter (cost vectors).
$T[partition, letter]$ =cost of subtree defined by partition when root is letter.

input$(Column[1 \ldots n])$

for $l = 1, \ldots, n$ % initialize leaf cost vectors.
  for $\sigma \in \{A, C, G, T, -\}$
    if $(\sigma == Column[l])$ $V[1, l, \sigma] = 0$;
    else $V[1, l, \sigma] = 1$;

% compute minimum cost vectors.
for $k = 2, \ldots, n$ % size of substring
  for $l = 1, \ldots, n - k$ % start of substring
    for $d = 1, \ldots, k - 1$ % partition
      % get cost vector for each partition position
      for $\sigma \in \{A, C, G, T, -\}$ %characters
        $T[d, \sigma] = V[d, l, \sigma] + V[k - d, l + d, \sigma]$;

      $m = \min_{\sigma \in \{A,C,G,T,-\}} \{T[d, \sigma]\}$;

      for $\sigma \in \{A, C, G, T, -\}$
        if $(T[d, \sigma] \neq m)$ $T[d, \sigma] = m + 1$;

    % get overall cost vector.
    for $\sigma \in \{A, C, G, T, -\}$
      $V[k, l, \sigma] = \min_{d=1,\ldots,k-1} \{T[d, \sigma]\}$;

% get minimum cost for optimal tree.
$m = \min_{\sigma \in \{A,C,G,T,-\}} \{V[n, 1, \sigma]\}$;

return(m); % return minimum from root

Figure 8: Algorithm for optimal solution for TRHIST, single column, fixed duplication size, binary contractions.

Although the Fitch and Sankoff papers assumed that the tree is given, the method can also be used to determine the minimum cost at each node of the tree as the tree is being constructed from the bottom up.

Fig. 8 is pseudocode for a $O(n^3)$ algorithm for finding the cost of the optimal tree with *binary* contractions. It builds the tree from the bottom up, working with substrings of the column of size $k = 2 \ldots n$. For each substring, an optimal binary tree connecting the characters in the substring is determined. As above, each substring is assigned a cost vector with the costs of the leaves fixed. Cost vectors are stored in an array V indexed by the substring length and starting position.

When $k = 2$ the cost vector for adjacent characters is determined by the method described above. When $k \geq 3$, the optimal tree for the substring will have a left and a right subtree. There are $k-1$ possible locations in the substring for a partition between the subtrees and each is tried in turn. One cost vector per partition position is determined and a final cost vector for the substring is obtained by finding for each character its minimum over all the fixed partition costs. Note that this implies that each character may have its own minimizing tree.

Extension to r columns. The preceding algorithm can be generalized for r columns as follows. Instead of finding a single column vector for each substring, we instead find an r-dimensional array, one dimension for each column. The size of this array is 5^r. The values in the array represent the combined costs of the same tree on each of the r substrings for every letter combination at the r roots. Time complexity is $O(5^r \cdot n^3)$ which is exponential in the number of columns.