An Algorithm for Finding the Largest Approximately Common Substructures of Two Trees

Jason T. L. Wang Bruce A. Shapiro

Dennis Shasha Kaizhong Zhang Kathleen M. Currey*

Abstract — Ordered, labeled trees are trees in which each node has a label and the left-to-right order of its children (if it has any) is fixed. Such trees have many applications in vision, pattern recognition, molecular biology and natural language processing. We consider a substructure of an ordered labeled tree T to be a connected subgraph of T. Given two ordered labeled trees T_1 and T_2 and an integer d, the largest approximately common substructure problem is to find a substructure U_1 of T_1 and a substructure U_2 of T_2 such that U_1 is within edit distance d of U_2 and where there does not exist any other substructure V_1 of T_1 and V_2 of T_2 such that V_1 and V_2 satisfy the distance constraint and the sum of the sizes of V_1 and V_2 is greater than the sum of the sizes of U_1 and U_2. We present a dynamic programming algorithm to solve this problem, which runs as fast as the fastest known algorithm for computing the edit distance of two trees when the distance allowed in the common substructures is a constant independent of the input trees. To demonstrate the utility of our algorithm, we discuss its application to discovering motifs in multiple RNA secondary structures (which are ordered labeled trees).

Index Terms — Computational biology, dynamic programming, pattern matching, pattern recognition, trees.

*J. T. L. Wang is with the Department of Computer and Information Science, New Jersey Institute of Technology, University Heights, Newark, NJ 07102. E-mail: jason@cis.njit.edu. B. A. Shapiro is with the Image Processing Section, Laboratory of Experimental and Computational Biology, Division of Basic Sciences, National Cancer Institute, National Institutes of Health, Frederick, MD 21702. E-mail: bshapiro@ncifcrf.gov. D. Shasha is with the Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, NY 10012. E-mail: shasha@cs.nyu.edu. K. Zhang is with the Department of Computer Science, The University of Western Ontario, London, Ontario, Canada N6A 5B7. E-mail: kzhang@csd.uwo.ca. K. M. Currey is with the Image Processing Section, Laboratory of Experimental and Computational Biology, Division of Basic Sciences, National Cancer Institute, Frederick Cancer Research and Development Center, National Institutes of Health, Frederick, MD 21702 and University of Maryland Medical Center, Department of Pediatrics, Baltimore, MD 21201.

1 Introduction

Ordered, labeled trees are trees in which each node has a label and the left-to-right order of its children (if it has any) is fixed.[1] Such trees have many applications in vision, pattern recognition, molecular biology and natural language processing, including the representation of images [12], patterns [2, 10] and secondary structures of RNA [14]. They are frequently used in other disciplines as well.

A large amount of work has been performed for comparing two trees based on various distance measures [4, 9, 11, 21, 25]. [16, 19, 27] recently generalized one of the most commonly used distance measures, namely the edit distance, for both rooted and unrooted unordered trees. These works laid out a foundation that is useful for comparing graphs [15, 24].

In this paper we extend the previous work by considering the largest approximately common substructure problem for ordered labeled trees. Various biologists [5, 14] represent RNA secondary structures as trees. Finding common patterns (also known as motifs) in these secondary structures helps both in predicting RNA folding [5] and in functional studies of RNA processing mechanisms [14].

Previous methods for detecting motifs in the RNA molecules (trees) are based on one of the following two approaches: (1) transforming the trees to sequences and then using sequence algorithms [13]; (2) representing the molecules using a highly simplified tree structure and then searching for common nodes in the trees [5]. Neither of the two approaches satisfactorily takes the full tree structure into account. By contrast, utilizing the proposed algorithm for pairs of trees enables one to locate tree-structured motifs occurring in multiple RNA secondary structures. Our experimental results concerning RNA classification show the significance of these motifs [23].

2 Preliminaries

2.1 Edit Distance and Mappings

We use the edit distance [17] to measure the dissimilarity of two trees. There are three types of edit operations, i.e., *relabeling*, *delete*, and *insert* a node. Relabeling node n means changing the label on n. Deleting a node n means making the children of n become the children of the parent of n and removing n. Insert is the inverse of delete. Inserting node n as the child of node n' makes n the parent of a consecutive subsequence of the current children of n'. Fig. 1 illustrates the edit operations. For the purpose of this work, we assume that all edit operations have a unit cost. The edit distance, or simply the *distance*, from tree T_1 to tree T_2, denoted $\delta(T_1, T_2)$, is the cost of a minimum cost sequence of edit operations transforming T_1 to T_2 [17].

The notion of edit distance is best illustrated through the concept of *mappings*. A mapping is a graphical specification of which edit operations apply to each node in the two trees. For example, the mapping in Fig. 2 shows a way to transform T_1 to T_2. The transformation includes deleting the two nodes labeled a and m in T_1 and inserting them into T_2.

[1] Throughout the paper, we shall refer to ordered labeled trees simply as trees when no ambiguity occurs.

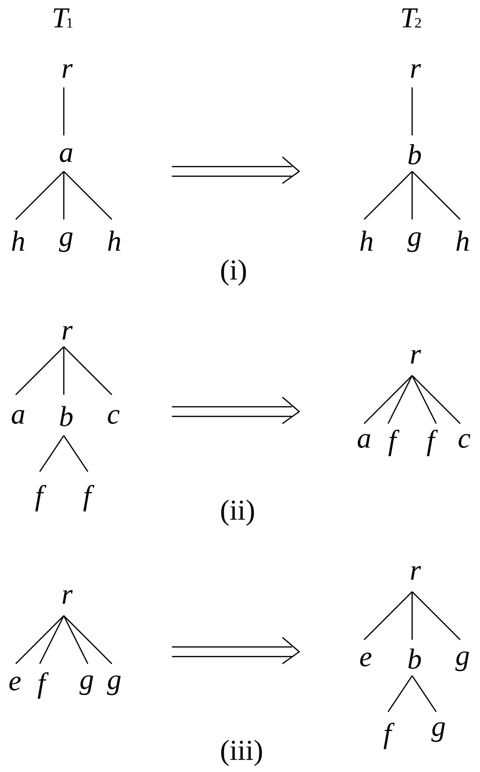


Fig. 1. (i) Relabeling: To change one node label (a) to another (b). (ii) Delete: To delete a node; all children of the deleted node (labeled b) become children of the parent (labeled r). (iii) Insert: To insert a node; a consecutive sequence of siblings among the children of the node labeled r (here, f and the left g) become the children of the newly inserted node labeled b.

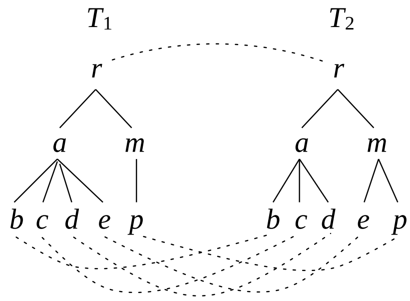


Fig. 2. A mapping from tree T_1 to tree T_2.

We use a postorder numbering of nodes in the trees. Let $t[i]$ represent the node of T whose position in the left-to-right postorder traversal of T is i. When there is no confusion, we also use $t[i]$ to represent the label of node $t[i]$. Formally, a mapping from T_1 to T_2 is a triple (M, T_1, T_2) (or simply M if the context is clear), where M is any set of ordered pairs of integers (i, j) satisfying: (i) $1 \leq i \leq |T_1|$ and $1 \leq j \leq |T_2|$; (ii) For any pair of (i_1, j_1) and (i_2, j_2) in M, (a) $i_1 = i_2$ iff $j_1 = j_2$ (one-to-one condition); (b) $t_1[i_1]$ is to the left of $t_1[i_2]$ iff $t_2[j_1]$ is to the left of $t_2[j_2]$ (sibling order preservation condition); (c) $t_1[i_1]$ is an ancestor of $t_1[i_2]$ iff $t_2[j_1]$ is an ancestor of $t_2[j_2]$ (ancestor order preservation condition). The cost of M is the cost of deleting nodes of T_1 not touched by a mapping line plus the cost of inserting nodes of T_2 not touched by a mapping line plus the cost of relabeling nodes in those pairs related by mapping lines with different labels. It can be proved [17] that $\delta(T_1, T_2)$ equals the cost of a minimum cost mapping from tree T_1 to tree T_2.

2.2 Cut Operations

We define a substructure U of tree T to be a connected subgraph of T. That is, U is rooted at a node n in T and is generated by cutting off some subtrees in the subtree rooted at n. Formally, let $T[i]$ represent the subtree rooted at $t[i]$. The operation of cutting at node $t[i]$ means removing $T[i]$. A set S of nodes of $T[k]$ is said to be a set of consistent subtree cuts in $T[k]$ if (i) $t[i] \in S$ implies that $t[i]$ is a node in $T[k]$, and (ii) $t[i], t[j] \in S$ implies that neither is an ancestor of the other in $T[k]$. Intuitively, S is the set of all roots of the removed subtrees in $T[k]$.

We use $Cut(T, S)$ to represent the tree T with subtree removals at all nodes in S. Let $Subtrees(T)$ be the set of all possible sets of consistent subtree cuts in T. Given two trees T_1 and T_2 and an integer d, the size of the largest approximately common root-containing substructures within distance k, $0 \le k \le d$, of $T_1[i]$ and $T_2[j]$, denoted $\beta(T_1[i], T_2[j], k)$ (or simply $\beta(i, j, k)$ when the context is clear), is $\max\{|Cut(T_1[i], S_1)| + |Cut(T_2[j], S_2)|\}$ subject to $\delta(Cut(T_1[i], S_1), Cut(T_2[j], S_2)) \le k$, $S_1 \in Subtrees(T_1[i])$, $S_2 \in Subtrees(T_2[j])$. Finding the largest approximately common substructure (LACS), within distance d, of $T_1[i]$ and $T_2[j]$ amounts to calculating $\max_{1 \le u \le i, 1 \le v \le j} \{\beta(T_1[u], T_2[v], d)\}$ and locating the $Cut(T_1[u], S_u)$ and $Cut(T_2[v], S_v)$, $S_u \in Subtrees(T_1[u])$, $S_v \in Subtrees(T_2[v])$ that achieve the maximum size. The size of LACS, within distance d, of T_1 and T_2 is $\max_{1 \le i \le |T_1|, 1 \le j \le |T_2|} \{\beta(T_1[i], T_2[j], d)\}$. We shall focus on computing the maximum size. By memorizing the size information during the computation and by a simple backtracking technique, one can find both the maximum size and one of the corresponding substructure pairs yielding the size in the same time and space complexity.

3 Our Algorithm

3.1 Notation

We use $desc(i)$ to represent the set of postorder numbers of the descendants of the node $t[i]$ and $l(i)$ denotes the postorder number of the leftmost leaf of the subtree $T[i]$. When $T[i]$ is a leaf, $l(i) = i$. $T[i..j]$ is an ordered forest of tree T induced by the nodes numbered i to j inclusive (see Fig. 3). If $i > j$, then $T[i..j] = \emptyset$. The definition of mappings for ordered forests is the same as for trees. Let F_1 and F_2 be two forests. The distance from F_1 to F_2, denoted $\Delta(F_1, F_2)$, equals the cost of a minimum cost mapping from F_1 to F_2 [25].

Let $F = T[i..j]$. A set S of nodes of F is said to be a set of consistent subtree cuts in F if (i) $t[p] \in S$ implies that $i \le p \le j$, and (ii) $t[p], t[q] \in S$ implies that neither is an ancestor of the other in F. We use $Cut(F, S)$ to represent the sub-forest F with subtree removals at all nodes in S. Let $Subtrees(F)$

be the set of all possible sets of consistent subtree cuts in F. Define the size of the largest approximately common root-containing substructures, within distance k, of F_1 and F_2, denoted $\Psi(F_1, F_2, k)$, to be $\max\{|Cut(F_1, S_1)| + |Cut(F_2, S_2)|\}$ subject to $\Delta(Cut(F_1, S_1), Cut(F_2, S_2)) \leq k$, $S_1 \in Subtrees(F_1)$, $S_2 \in Subtrees(F_2)$. When $F_1 = T_1[l(i)..s]$ and $F_2 = T_2[l(j)..t]$, we also represent $\Psi(F_1, F_2, k)$ by $\Psi(l(i)..s, l(j)..t, k)$ if there is no confusion.

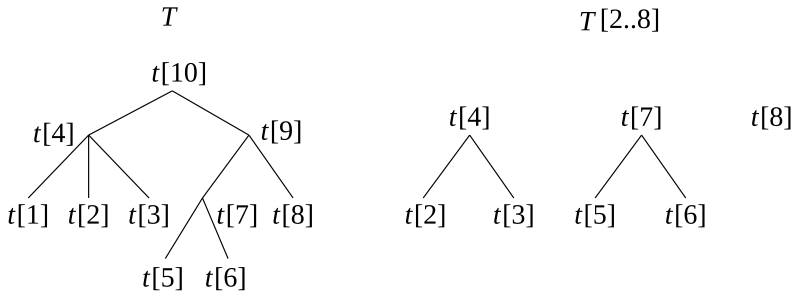


Fig. 3. An induced ordered forest.

3.2 Basic Properties

Lemma 3.1. *Suppose $s \in desc(i)$ and $t \in desc(j)$. Then*
(i) $\Psi(\emptyset, \emptyset, 0) = 0$;
(ii) $\Psi(T_1[l(i)..s], \emptyset, 0) = 0$;
(iii) $\Psi(\emptyset, T_2[l(j)..t], 0) = 0$.

Proof. Immediate from definitions. □

Lemma 3.2. *Suppose $s \in desc(i)$ and $t \in desc(j)$. Then for all k, $1 \leq k \leq d$,*
(i) $\Psi(\emptyset, \emptyset, k) = 0$;
(ii)

$$\Psi(T_1[l(i)..s], \emptyset, k) = \mathbf{max} \begin{cases} \Psi(T_1[l(i)..s-1], \emptyset, k-1) + 1, \\ \Psi(T_1[l(i)..l(s)-1], \emptyset, k); \end{cases}$$

(iii)

$$\Psi(\emptyset, T_2[l(j)..t], k) = \mathbf{max} \begin{cases} \Psi(\emptyset, T_2[l(j)..t-1], k-1) + 1, \\ \Psi(\emptyset, T_2[l(j)..l(t)-1], k). \end{cases}$$

Proof. (i) follows from the definition. For (ii), suppose $S_1 \in Subtrees(T_1[l(i)..s])$ is a smallest set of consistent subtree cuts that maximizes $|Cut(T_1[l(i)..s], S_1)|$ where $\Delta(Cut(T_1[l(i)..s], S_1), \emptyset) \leq k$. Then one of the following two cases must hold: (1) $t_1[s] \in S_1$; (2) $t_1[s] \notin S_1$. If (1) is true, then $\Psi(T_1[l(i)..s], \emptyset, k)$ $= \Psi(T_1[l(i)..l(s)-1], \emptyset, k)$; otherwise, $\Psi(T_1[l(i)..s], \emptyset, k) = \Psi(T_1[l(i)..s-1], \emptyset, k-1) + 1$. (iii) is proved similarly as for (ii). □

Lemma 3.3. *Suppose $s \in desc(i)$ and $t \in desc(j)$. If ($l(s) \neq l(i)$ or $l(t) \neq l(j)$), then*

$$\Psi(l(i)..s, l(j)..t, 0) = \mathbf{max} \begin{cases} \Psi(l(i)..l(s)-1, l(j)..t, 0), \\ \Psi(l(i)..s, l(j)..l(t)-1, 0), \\ \Psi(l(i)..l(s)-1, l(j)..l(t)-1, 0) + \beta(s,t,0). \end{cases}$$

Proof. Suppose $S_1 \in Subtrees(T_1[l(i)..s])$ and $S_2 \in Subtrees(T_2[l(j)..t])$ are two smallest sets of consistent subtree cuts that maximize $|Cut(T_1[l(i)..s], S_1)| + |Cut(T_2[l(j)..t], S_2)|$ where $\Delta(Cut(T_1[l(i)..s], S_1), Cut(T_2[l(j)..t], S_2)) = 0$. Then at least one of the following cases must hold:

Case 1. $t_1[s] \in S_1$ (i.e., the subtree $T_1[s]$ is removed). So, $\Psi(l(i)..s, l(j)..t, 0) = \Psi(l(i)..l(s)-1, l(j)..t, 0)$.

Case 2. $t_2[t] \in S_2$ (i.e., the subtree $T_2[t]$ is removed). So, $\Psi(l(i)..s, l(j)..t, 0) = \Psi(l(i)..s, l(j)..l(t)-1, 0)$.

Case 3. $t_1[s] \notin S_1$ and $t_2[t] \notin S_2$ (i.e., neither $T_1[s]$ nor $T_2[t]$ is removed) (Fig. 4). Let M be the mapping (with cost 0) from $Cut(T_1[l(i)..s], S_1)$ to $Cut(T_2[l(j)..t], S_2)$. In M, $T_1[s]$ must be mapped to $T_2[t]$ because otherwise we cannot have distance zero between $Cut(T_1[l(i)..s], S_1)$ and $Cut(T_2[l(j)..t], S_2)$. Therefore $\Psi(l(i)..s, l(j)..t, 0) = \Psi(l(i)..l(s)-1, l(j)..l(t)-1, 0) + \beta(s,t,0)$.

Since these three cases exhaust all possible mappings yielding $\Psi(l(i)..s, l(j)..t, 0)$, we take the maximum of the corresponding sizes, which gives the formula asserted by the lemma. □

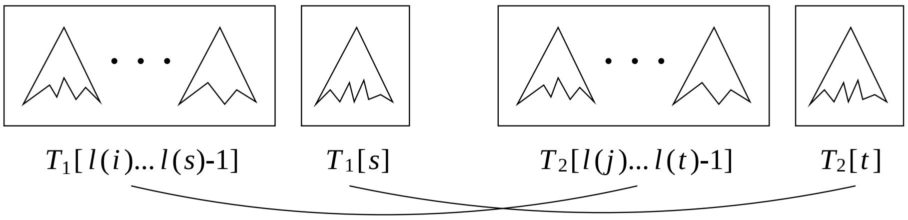


Fig. 4. Illustration of the case in which one of $T_1[l(i)..s]$ and $T_2[l(j)..t]$ is a forest and neither $T_1[s]$ nor $T_2[t]$ is removed.

Lemma 3.4. *Suppose $s \in desc(i)$ and $t \in desc(j)$. Suppose both $T_1[l(i)..s]$ and $T_2[l(j)..t]$ are trees (i.e., $l(s) = l(i)$ and $l(t) = l(j)$). Then*

$$\Psi(l(i)..s, l(j)..t, 0) = \begin{cases} \Psi(l(i)..s-1, l(j)..t-1, 0) + 2 & \textit{if } t_1[s] = t_2[t], \\ 0 & \textit{otherwise.} \end{cases}$$

Proof. Since $l(s) = l(i)$ and $l(t) = l(j)$, $T_1[l(i)..s] = T_1[s]$ and $T_2[l(j)..t] = T_2[t]$. First, consider the case where $t_1[s] = t_2[t]$. Suppose $S_1 \in Subtrees(T_1[s])$ and $S_2 \in Subtrees(T_2[t])$ are two smallest sets of consistent subtree cuts that maximize $|Cut(T_1[s], S_1)| + |Cut(T_2[t], S_2)|$ where $\delta(Cut(T_1[s], S_1), Cut(T_2[t], S_2)) = 0$. Let M be the mapping (with cost 0) from $Cut(T_1[s], S_1)$ to $Cut(T_2[t], S_2)$ (see Fig. 5). Clearly, in M, $t_1[s]$ must be mapped to $t_2[t]$. Furthermore, the largest common root-containing substructure of $T_1[l(i)..s-1]$ and $T_2[l(j)..t-1]$ plus $t_1[s]$ (or $t_2[t]$) must be the largest common root-containing substructure of $T_1[s]$ and $T_2[t]$. This means that $\Psi(l(i)..s, l(j)..t, 0) = \Psi(l(i)..s-1, l(j)..t-1, 0) + 2$, where the 2 is obtained by including the two nodes $t_1[s]$ and $t_2[t]$.

Next consider the case where $t_1[s] \neq t_2[t]$ (i.e., the roots of the two trees $T_1[s]$ and $T_2[t]$ differ). In order to get distance zero between the two trees after applying cut operations to them, we have to remove both trees entirely. Thus, $\Psi(l(i)..s, l(j)..t, 0) = 0$. $\square$

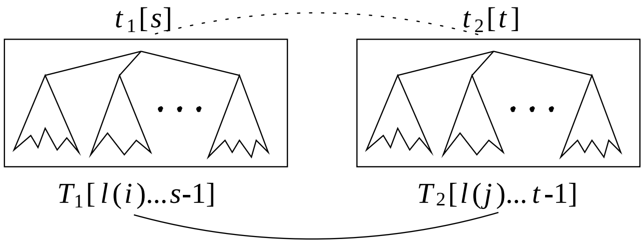


Fig. 5. Illustration of the case in which both $T_1[l(i)..s]$ and $T_2[l(j)..t]$ are trees and $t_1[s] = t_2[t]$.

Lemma 3.5. *Suppose $s \in desc(i)$ and $t \in desc(j)$. If ($l(s) \neq l(i)$ or $l(t) \neq l(j)$), then for all k, $1 \leq k \leq d$,*

$$\Psi(l(i)..s, l(j)..t, k) = \mathbf{max} \begin{cases} \Psi(l(i)..l(s)-1, l(j)..t, k), \\ \Psi(l(i)..s, l(j)..l(t)-1, k), \\ \Psi(l(i)..s-1, l(j)..t, k-1)+1, \\ \Psi(l(i)..s, l(j)..t-1, k-1)+1, \\ \max_{0 \leq h \leq k}\{\Psi(l(i)..l(s)-1, l(j)..l(t)-1, k-h) + \beta(s,t,h)\}. \end{cases}$$

Proof. Suppose $S_1 \in Subtrees(T_1[l(i)..s])$ and $S_2 \in Subtrees(T_2[l(j)..t])$ are two smallest sets of consistent subtree cuts that maximize $|Cut(T_1[l(i)..s], S_1)| + |Cut(T_2[l(j)..t], S_2)|$ where $\Delta(Cut(T_1[l(i)..s], S_1), Cut(T_2[l(j)..t], S_2)) \leq k$. Then at least one of the following cases must hold:

Case 1. $t_1[s] \in S_1$. So, $\Psi(l(i)..s, l(j)..t, k) = \Psi(l(i)..l(s)-1, l(j)..t, k)$.

Case 2. $t_2[t] \in S_2$. So, $\Psi(l(i)..s, l(j)..t, k) = \Psi(l(i)..s, l(j)..l(t)-1, k)$.

Case 3. $t_1[s] \notin S_1$ and $t_2[t] \notin S_2$. Let M be a minimum cost mapping from $Cut(T_1[l(i)..s], S_1)$ to $Cut(T_2[l(j)..t], S_2)$. There are three subcases to examine:

(a) $t_1[s]$ is not touched by a line in M. Then, $\Psi(l(i)..s, l(j)..t, k) = \Psi(l(i)..s-1, l(j)..t, k-1) + 1$.

(b) $t_2[t]$ is not touched by a line in M. Then, $\Psi(l(i)..s, l(j)..t, k) = \Psi(l(i)..s, l(j)..t-1, k-1) + 1$.

(c) $t_1[s]$ and $t_2[t]$ are both touched by lines in M. Then $(s,t) \in M$. So, there exists an h such that $\Psi(l(i)..s, l(j)..t, k) = \Psi(l(i)..l(s)-1, l(j)..l(t)-1, k-h) + \beta(s,t,h)$. The value of h ranges from 0 to k. Therefore we take the maximum of the corresponding sizes, i.e., $\Psi(l(i)..s, l(j)..t, k) = \max_{0 \leq h \leq k}\{\Psi(l(i)..l(s)-1, l(j)..l(t)-1, k-h) + \beta(s,t,h)\}$. $\square$

Lemma 3.6. *Suppose $s \in desc(i)$ and $t \in desc(j)$. Suppose both $T_1[l(i)..s]$ and $T_2[l(j)..t]$ are trees* *(i.e., $l(s) = l(i)$ and $l(t) = l(j)$). Then for all k, $1 \le k \le d$,*

$$\Psi(l(i)..s, l(j)..t, k) = \mathbf{max} \begin{cases} \Psi(l(i)..s-1, l(j)..t, k-1) + 1, \\ \Psi(l(i)..s, l(j)..t-1, k-1) + 1, \\ \Psi(l(i)..s-1, l(j)..t-1, k-c) + 2, \end{cases}$$

where

$$c = \begin{cases} 0 & \textit{if } t_1[s] = t_2[t], \\ 1 & \textit{otherwise.} \end{cases}$$

Proof. Since $T_1[l(i)..s]$ and $T_2[l(j)..t]$ are trees, $T_1[l(i)..s] = T_1[s]$ and $T_2[l(j)..t] = T_2[t]$. We first show that removing either $T_1[s]$ or $T_2[t]$ would not yield the maximum size. There are three cases to be considered:

Case 1. Both $T_1[s]$ and $T_2[t]$ are removed. Then, $\Psi(l(i)..s, l(j)..t, k) = 0$. However, since $k \ge 1$, cutting at just the children of $t_1[s]$ and $t_2[t]$ would cause $\Psi(l(i)..s, l(j)..t, k) \ge 2$. Therefore removing both $T_1[s]$ and $T_2[t]$ cannot yield the maximum size.

Case 2. Only $T_1[s]$ is removed. Then, $\Psi(l(i)..s, l(j)..t, k) = \beta(\emptyset, T_2[t], k)$. Assume without loss of generality that $|T_2[t]| \ge k$. The above equation implies that we have to remove some subtrees from $T_2[t]$ so that there are no more than k nodes left in $T_2[t]$. Thus, $\Psi(l(i)..s, l(j)..t, k) = \beta(\emptyset, T_2[t], k) = k$. On the other hand, if we just cut at the children of $t_1[s]$ and leave $t_1[s]$ in the tree, we would map $t_1[s]$ to $t_2[t]$. This would lead to $\Psi(l(i)..s, l(j)..t, k) \ge \Psi(\emptyset, T_2[l(j)..t-1], k-1) + 2 = k+1$. Thus, removing $T_1[s]$ alone cannot yield the maximum size.

Case 3. Only $T_2[t]$ is removed. The proof is similar to that in Case 2.

The above arguments lead to the conclusion that in order to obtain the maximum size, neither $T_1[s]$ nor $T_2[t]$ can be removed. Now suppose $S_1 \in Subtrees(T_1[s])$ and $S_2 \in Subtrees(T_2[t])$ are two smallest sets of consistent subtree cuts that maximize $|Cut(T_1[s], S_1)| + |Cut(T_2[t], S_2)|$ where $\delta(Cut(T_1[s], S_1), Cut(T_2[t], S_2)) \le k$. Let M be a minimum cost mapping from $Cut(T_1[s], S_1)$ to $Cut(T_2[t], S_2)$. Then at least one of the following cases must hold:

Case 1. $t_1[s]$ is not touched by a line in M. So, $\Psi(l(i)..s, l(j)..t, k) = \Psi(l(i)..s-1,\ l(j)..t,\ k-1) + 1$.

Case 2. $t_2[t]$ is not touched by a line in M. So, $\Psi(l(i)..s, l(j)..t, k) = \Psi(l(i)..s,\ l(j)..t-1,\ k-1) + 1$.

Case 3. Both $t_1[s]$ and $t_2[t]$ are touched by lines in M. By the ancestor order preservation and sibling order preservation conditions on mappings (cf. Section 2.1), (s, t) must be in M. Thus, if $t_1[s] = t_2[t]$, we have $\Psi(l(i)..s, l(j)..t, k) = \Psi(l(i)..s-1, l(j)..t-1, k) + 2$; otherwise, since mapping $t_1[s]$ to $t_2[t]$ costs 1, we have $\Psi(l(i)..s, l(j)..t, k) = \Psi(l(i)..s-1, l(j)..t-1, k-1) + 2$. □

3.3 The Algorithm

From Lemma 3.4 and Lemma 3.6, we observe that when s is on the path from $l(i)$ to i and t is on the path from $l(j)$ to j, we need not compute $\beta(s, t, k)$, $0 \le k \le d$, separately, since they can be obtained during the computation of $\beta(i, j, k)$. Thus, we will only consider nodes that are either the roots of the trees or having a left sibling. Let keynodes(T) contain all such nodes of a tree T, i.e., keynodes(T) $= \{k |$there exists no $k' > k$ such that $l(k) = l(k')\}$. For each $i \in$ keynodes(T_1) and $j \in$ keynodes(T_2), Procedure Find-Largest-1 in Fig. 6 computes $\beta(s, t, 0)$ for $l(i) \le s \le i$ and $l(j) \le t \le j$ and

Procedure Find-Largest-2 in Fig. 6 computes $\beta(s,t,k)$ for $1 \le k \le d$. The main algorithm is summarized in Fig. 6.

Now, to calculate the size of the largest approximately common substructures (LACSs), within distance d, of $T_1[i]$ and $T_2[j]$, we build, in a bottom-up fashion, another array $\gamma(i,j,d)$, $1 \le i \le |T_1|$, $1 \le j \le |T_2|$, using $\beta(i,j,d)$ as follows. Let $L = \max_{1\le u\le s}\gamma(i_u,j,d)$ where $i_1 \dots i_s$ are the postorder numbers of the children of $t_1[i]$ or $L = 0$ if $t_1[i]$ is a leaf. Let $R = \max_{1\le v\le t}\gamma(i,j_v,d)$ where $j_1 \dots j_t$ are the postorder numbers of the children of $t_2[j]$ or $R = 0$ if $t_2[j]$ is a leaf. Calculate $\gamma(i,j,d) = \max\{\beta(i,j,d), L, R\}$. The size of LACSs, within distance d, of $T_1[i]$ and $T_2[j]$ is $\gamma(i,j,d)$. The size of LACSs, within distance d, of T_1 and T_2 is $\gamma(|T_1|,|T_2|,d)$.

Consider the complexity of the algorithm. We use an array to hold Ψ, β and γ, respectively. These arrays require $O(d \times |T_1| \times |T_2|)$ space. Regarding the time complexity, given $\beta(i,j,d)$, $1 \le i \le |T_1|$, $1 \le j \le |T_2|$, calculating $\gamma(i,j,d)$ requires $O(|T_1| \times |T_2|)$ time. For a fixed i and j, Procedure Find-Largest-1 requires $O(|T_1[i]| \times |T_2[j]|)$ time and Procedure Find-Largest-2 requires $O(d^2 \times |T_1[i]| \times |T_2[j]|)$ time. So the total time is bounded by

$$
\begin{aligned}
& \sum_{i\in \text{keynodes}(T_1)} \sum_{j\in \text{keynodes}(T_2)} O(|T_1[i]| \times |T_2[j]|) + O(d^2 \times |T_1[i]| \times |T_2[j]|) \\
\le\ & O\Big(\sum_{i\in \text{keynodes}(T_1)} \sum_{j\in \text{keynodes}(T_2)} d^2 \times |T_1[i]| \times |T_2[j]|\Big) \\
\le\ & O\Big(d^2 \times \sum_{i\in \text{keynodes}(T_1)} |T_1[i]| \times \sum_{j\in \text{keynodes}(T_2)} |T_2[j]|\Big).
\end{aligned}
$$

From [25, Theorem 2], the last term above is bounded by $O(d^2 \times |T_1| \times |T_2| \times \min(H_1, L_1) \times \min(H_2, L_2))$ where H_i, $i = 1, 2$, is the height of T_i and L_i is the number of leaves in T_i. When d is a constant, this is the same as the complexity of the best current algorithm for tree matching based on the edit distance [11, 25], even though the problem at hand appears to be harder than tree matching.

Note that to calculate $\max_{1\le i\le |T_1|, 1\le j\le |T_2|}\{\beta(i,j,0)\}$, one could use a faster algorithm that runs in time $O(|T_1| \times |T_2|)$. However, the reason for considering the keynodes and the formulas as specified in Lemmas 3.3 and 3.4 is to prepare the optimal sizes from forests to forests and store these size values in the array to be used in calculating $\beta(s,t,k)$ for $k \ne 0$. Even if one could incorporate the faster algorithm into the Find-Largest algorithm, the overall time complexity would not be changed, because the calculation of $\beta(s,t,k)$ for $k \ne 0$ dominates the cost.

4 Implementation and Discussion

We have applied our algorithm to find motifs in multiple RNA secondary structures. In this experiment, we examined three phylogenetically related families of mRNA sequences chosen from GenBank [1] pertaining to the poliovirus, human rhinovirus and coxsackievirus. Each family contained two sequences, as shown in Table 1.

Algorithm Find-Largest

Input: Trees T_1, T_2 and an integer d.

Output: $\beta(i,j,k)$ where $1 \le i \le |T_1|$, $1 \le j \le |T_2|$ and $0 \le k \le d$.

```
for i' := 1 to |keynodes(T1)| do
    for j' := 1 to |keynodes(T2)| do
        begin
            i := keynodes(T1)[i'];
            j := keynodes(T2)[j'];
            run Procedure Find-Largest-1 on input (i, j, 0);
            run Procedure Find-Largest-2 on input (i, j, d);
        end;
```

Procedure Find-Largest-1

Input: $i, j, 0$.

Output: $\beta(s,t,0)$ where $l(i) \le s \le i$ and $l(j) \le t \le j$.

```
Ψ(∅, ∅, 0) := 0;
for s := l(i) to i do
    Ψ(T1[l(i)..s], ∅, 0) := 0;
for t := l(j) to j do
    Ψ(∅, T2[l(j)..t], 0) := 0;
for s := l(i) to i do
    for t := l(j) to j do
        if (l(s) ≠ l(i) or l(t) ≠ l(j)) then
            compute Ψ(l(i)..s, l(j)..t, 0) as in Lemma 3.3;
        else begin      /* l(s) = l(i) and l(t) = l(j) */
            compute Ψ(l(i)..s, l(j)..t, 0) as in Lemma 3.4;
            β(s, t, 0) := Ψ(l(i)..s, l(j)..t, 0);
        end;
```

Procedure Find-Largest-2

Input: i, j, d.

Output: $\beta(s,t,k)$ where $l(i) \le s \le i$, $l(j) \le t \le j$ and $1 \le k \le d$.

```
for k := 1 to d do
    Ψ(∅, ∅, k) := 0;
for k := 1 to d do
    for s := l(i) to i do
        compute Ψ(T1[l(i)..s], ∅, k) as in Lemma 3.2 (ii);
for k := 1 to d do
    for t := l(j) to j do
        compute Ψ(∅, T2[l(j)..t], k) as in Lemma 3.2 (iii);
for k := 1 to d do
    for s := l(i) to i do
        for t := l(j) to j do
            if (l(s) ≠ l(i) or l(t) ≠ l(j)) then
                compute Ψ(l(i)..s, l(j)..t, k) as in Lemma 3.5;
            else begin      /* l(s) = l(i) and l(t) = l(j) */
                compute Ψ(l(i)..s, l(j)..t, k) as in Lemma 3.6;
                β(s, t, k) := Ψ(l(i)..s, l(j)..t, k);
            end;
```

Fig. 6. Algorithm for computing $\beta(i,j,k)$.

Family	Sequence	# of trees	File #
poliovirus	polio3 sabin strain	3,026	file 1
	pol3mut	3,000	file 2
human rhinovirus	rhino 2	3,000	file 3
	rhino 14	3,000	file 4
coxsackievirus	cox5	3,000	file 5
	cvb305pr	2,999	file 6

Table 1. Data used in the experiment.

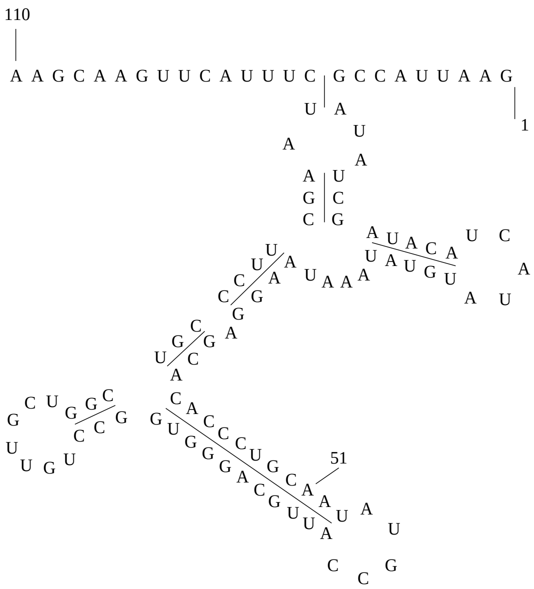
Under physiological conditions, i.e., at or above the room temperature, these RNA molecules do not take on only a single structure. They may change their conformation between structures with similar free energies or be trapped in local minima. Thus, one has to consider not only the optimal structure but all structures within a certain range of free energies. On the other hand, a loose rule of thumb is that the "real" structure of an RNA molecule appears in the top 5% - 10% of suboptimal structures of the sequence based on the ranking of their energies with the minimum energy one (i.e. the optimal one) being at the top. Therefore, we folded the 5' non-coding region of the selected mRNA sequences and collected (roughly) the top 3,000 suboptimal structures for each sequence. We then transformed these suboptimal structures into trees using the algorithms described in [13, 14]. Fig. 7 illustrates an RNA secondary structure and its tree representation.

The structure is decomposed into five terms: stem, hairpin, bulge, internal loop and multi-branch loop [14]. In the tree, H represents hairpin nodes, I represents internal loops, B represents bulge loops, M represents multi-branch loops, R represents helical stem regions (shown as connecting arcs) and N is a special node used to make sure the tree is connected. The tree is considered to be an ordered one where the ordering is imposed based upon the 5' to 3' nature of the molecule. The resulting trees for each mRNA sequence selected from GenBank were stored in a separate file, where the trees had between 70 and 180 nodes (cf. Table 1). Each tree is represented by a fully parenthesized notation where the root of every subtree precedes all the nodes contained in the subtree. Thus, for example, the tree depicted in Fig. 7(ii) is represented as (N(R(I(R(M(R(B(R(M(R(H))(R(H))))))(R(H))))))).

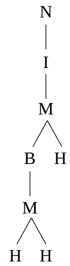
For each pair of trees T_1, T_2 in a file, we ran the algorithm Find-Largest on T_1, T_2, finding the size of the largest approximately common substructures, within distance 1, for each subtree pair $T_1[i]$ and $T_2[j]$, $1 \leq i \leq |T_1|$ and $1 \leq j \leq |T_2|$, and locating one of the corresponding substructure pairs yielding the size. These substructures constituted candidate motifs. Then we calculated the occurrence number[2] of each candidate motif M by adding variable length don't cares (VLDCs) to M as the new root and leaves to form a VLDC pattern V and then comparing V with each tree T in the file using the pattern matching technique developed in [26]. (A VLDC (conventionally denoted by "$*$") can be matched, at no cost, with a path or portion of a path in T. The technique calculates the minimum distance between V and T after implicitly computing an optimal substitution for the VLDCs in V, allowing zero or more cuttings at nodes from T (see Fig. 8).) This way we can locate the motifs approximately occurring in all (or the majority

[2]The occurrence number of a motif M with respect to distance k refers to the number of trees of the file in which M approximately occurs (i.e. these trees approximately contain M) within distance k.

of) the trees in the file.[3]



(i)



(ii)

Fig. 7. Illustration of a typical RNA secondary structure and its tree representation. (i) Normal polygonal representation of the structure. (ii) Tree representation of the structure.

Table 2 summarizes the results where the motifs occur within distance 0 in at least 350 trees in the corresponding file. The table shows the number of motifs discovered for each sequence, the number of distinct motifs found in common between both sequences of each family, and the minimum and maximum sizes of these common motifs. Table 3 shows some big motifs found in common in all the three families and the number of each sequence's secondary structures that contain the motifs. These motifs serve as a starting point to conduct further study of common motif analysis [3, 22].

[3] One can speed up this method by encoding the candidate motifs into a suffix tree and then using the statistical sampling and optimization techniques described in [23] to find the motifs.

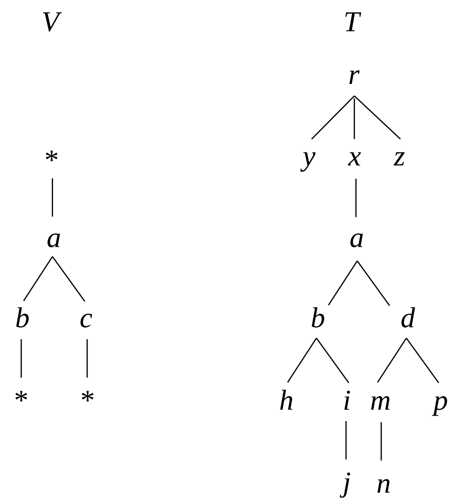


Fig. 8. Matching a VLDC pattern V and a tree T (both the pattern and tree are hypothetical ones solely used for illustration purposes). The root $*$ in V would be matched with nodes r, x in T, and the two leaves $*$ in V would be matched with nodes i, j and m, n in T, respectively. Nodes y, z, h, p in T would be cut. The distance of V and T would be 1 (representing the cost of changing c in V to d in T).

Family	Sequence	# of motifs found	# of common motifs	min size	max size
poliovirus	polio3 sabin strain	836	347	3	101
	pol3mut	793			
rhinovirus	rhino 2	287	70	3	10
	rhino 14	283			
coxsackievirus	cox5	306	136	3	20
	cvb305pr	391			

Table 2. Statistics concerning motifs discovered from the secondary structures of the mRNA sequences used in the experiment.

Motifs found	polio3	pol3mut	rhino 2	rhino 14	cox 5	cvb305pr
(R(M(R(I(R(H))))(R(B(R)))))	2,496	1,829	791	357	815	2,478
(R(M(R(H))(R(I(R)))))	3,024	3,000	3,000	801	2,997	2,999
(R(B(R(B(R(B(R)))))))	2,272	1,822	3,000	2,252	2,997	2,979
(R(M(R)(R(I(R(H))))))	2,074	1,712	3,000	702	2,997	2,999
(R(M(R(I(R)))(R(H))))	754	1,498	2,463	2,794	2,744	2,197

Table 3. Motifs found in common in the secondary structures of the poliovirus, human rhinovirus and coxsackievirus sequences. The motifs are represented in a fully parenthesized notation where the root of every subtree precedes all the nodes contained in the subtree. For each motif, the table also shows the number of each sequence's suboptimal structures that contain the motif.

The proposed algorithm and the discovered motifs have also been applied to RNA classification successfully [23]. Our experimental results showed that one can get more intersections of motifs from sequences of the same family. This indicates that closeness in motif corresponds to closeness in family. Another application of our algorithm is to apply it to a tree T and itself and calculate $\beta(i, j, 0)$ for $1 \leq i, j \leq |T|$. This allows one to find repeatedly occurring substructures (or *repeats* for short) in T. Finding repeats in secondary structures across different RNA sequences may help understand the structures of RNA. Readers interested in obtaining these programs may send a written request to any one of the authors.

Our work is based on the edit distance originated in [17]. This metric is more permissive than other worthy metrics (e.g. [18, 19, 20]) and therefore helps to locate subtle motifs existing in RNA secondary structures. The algorithm presented here assumes a unit cost for all edit operations. In practice, a more refined non-unit cost function can reflect more subtle differences in the RNA secondary structures [14]. It would then be interesting to score the measures in detecting common substructures or repeats in trees. Another interesting problem is to find a largest consensus motif T_3 in two input trees T_1 and T_2 where T_3 is a largest tree such that each of T_1 and T_2 has a substructure that is within a given distance to T_3. A comparison of the different types of common substructures (see also [6, 7, 8]), probably based on different metrics (e.g. [18, 19, 20]), as well as their applications remains to be explored.

Acknowledgments

We wish to thank the anonymous reviewers for their constructive suggestions and pointers to some relevant papers. We also thank Wojcieok Kasprzak (National Cancer Institute), Nat Goodman (Whitehead Institute of MIT) and Chia-Yo Chang for their useful comments and implementation efforts. This work was supported by the National Science Foundation under Grants IRI-9224601, IRI-9224602, IRI-9531548, IRI-9531554, and by the Natural Sciences and Engineering Research Council of Canada under Grant OGP0046373.

References

[1] C. Burks, M. Cassidy, M. J. Cinkosky, K. E. Cumella, P. Gilna, J. E.-D. Hayden, G. M. Keen, T. A. Kelley, M. Kelly, D. Kristofferson, and J. Ryals. GenBank. *Nucleic Acids Research*, 19:2221–2225, 1991.

[2] Y. C. Cheng and S. Y. Lu. Waveform correlation by tree matching. *IEEE Trans. Pattern Anal. Machine Intell.*, 7:299–305, May 1985.

[3] K. M. Currey and B. A. Shapiro. Secondary structure computer prediction of the polio virus 5' non-coding region is improved with a genetic algorithm. *Comput. Applic. Biosci.*, 13(1):1-12, 1997.

[4] T. Jiang, L. Wang, and K. Zhang. Alignment of trees – An alternative to tree edit. In M. Crochemore and D. Gusfield, editors, *Combinatorial Pattern Matching, Lecture Notes in Computer Science, 807*, pages 75–86. Springer-Verlag, 1994.

[5] S.-Y. Le, J. Owens, R. Nussinov, J.-H. Chen, B. A. Shapiro, and J. V. Maizel. RNA secondary structures: Comparison and determination of frequently recurring substructures by consensus. *Comput. Applic. Biosci.*, 5(3):205–210, 1989.

[6] S. Liu and E. Tanaka. A largest common similar substructure problem for trees embedded in a plane. Technical Report of the Institute of Electronics, Information and Communication Engineers, COMP 95–74, Jan. 1996.

[7] S. Liu and E. Tanaka. Largest common similar substructures of rooted and unordered trees. *Mem. Grad. School Sci. & Technol., Kobe Univ.*, 14-A:107–119, 1996.

[8] S. Liu and E. Tanaka. The largest common similar substructure problem. *IEICE Trans. Fundamentals*, E80-A:643–650, 1997.

[9] S. Y. Lu. A tree-matching algorithm based on node splitting and merging. *IEEE Trans. Pattern Anal. Machine Intell.*, 6(2):249–256, Mar. 1984.

[10] B. Moayer and K. S. Fu. A tree system approach for fingerprint pattern recognition. *IEEE Trans. Pattern Anal. Machine Intell.*, 8:376–387, May 1986.

[11] K. Ohmori and E. Tanaka. A unified view on tree metrics. In *Preprint of the Workshop on Syntactic and Structural Pattern Recognition* (Barcelona, 1986). Syntactic and Structural Pattern Recognition, Eds. G. Ferrate *et al.*, Springer, 1988.

[12] H. Samet. Distance transform for images represented by quadtrees. *IEEE Trans. Pattern Anal. Machine Intell.*, 4(3):298–303, May 1982.

[13] B. A. Shapiro. An algorithm for comparing multiple RNA secondary structures. *Comput. Applic. Biosci.*, 4(3):387–393, 1988.

[14] B. A. Shapiro and K. Zhang. Comparing multiple RNA secondary structures using tree comparisons. *Comput. Applic. Biosci.*, 6(4):309–318, 1990.

[15] L. G. Shapiro and R. M. Haralick. Structural descriptions and inexact matching. *IEEE Trans. Pattern Anal. Machine Intell.*, 3(5):504–519, Sep. 1981.

[16] D. Shasha, J. T. L. Wang, K. Zhang, and F. Y. Shih. Exact and approximate algorithms for unordered tree matching. *IEEE Transactions on Systems, Man and Cybernetics*, 24(4):668–678, April 1994.

[17] K.-C. Tai. The tree-to-tree correction problem. *J. ACM*, 26(3):422–433, 1979.

[18] E. Tanaka. The metric between rooted and ordered trees based on strongly structure preserving mapping and its computing method. *IECE Trans.*, J67-D(6):722–723, 1984.

[19] E. Tanaka. A metric between unrooted and unordered trees and its bottom-up computing method. *IEEE Trans. Pattern Anal. Machine Intell.*, 16(12):1233–1238, Dec. 1994.

[20] (a) E. Tanaka and K. Tanaka. A metric on trees and its computing method. *IECE Trans.*, J65-D(5): 511–518, 1982. (b) Correction to "A metric on trees and its computing method." *IEICE Trans.*, J76-D-I(11):635, 1993.

[21] E. Tanaka and K. Tanaka. The tree-to-tree editing problem. *International Journal of Pattern Recognition and Artificial Intelligence*, 2(2):221–240, 1988.

[22] Z. Tu, N. M. Chapman, G. Hufnagel, S. Tracy, B. A. Shapiro, J. R. Romero, W. H. Barry, L. Zhao, and K. M. Currey. The cardiovirulent phenotype of coxsackievirus B3 is determined at a single site in the genomic 5' non-translated region. *J. Virology*, 69:4607–4618, 1995.

[23] J. T. L. Wang, B. A. Shapiro, D. Shasha, K. Zhang, and C.-Y. Chang. Automated discovery of active motifs in multiple RNA secondary structures. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pages 70–75, Portland, Oregon, August 1996.

[24] A. K. Wong, M. You, and S. C. Chang. An algorithm for graph optimal monomorphism. *IEEE Transactions on Systems, Man and Cybernetics*, 20:628–639, 1990.

[25] K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262, Dec. 1989.

[26] K. Zhang, D. Shasha, and J. T. L. Wang. Approximate tree matching in the presence of variable length don't cares. *Journal of Algorithms*, 16(1):33–66, Jan. 1994.

[27] K. Zhang, J. T. L. Wang, and D. Shasha. On the editing distance between undirected acyclic graphs. *International Journal of Foundations of Computer Science*, 7(1):43–57, March 1996.