Information Processing Letters 42 (1992) 133–139
North-Holland

25 May 1992

On the editing distance between unordered labeled trees *

Kaizhong Zhang
Department of Computer Science, University of Western Ontario, London, Ontario, Canada N6A 5B7

Rick Statman
Department of Applied Mathematics, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Dennis Shasha **
Courant Institute, New York University, New York, NY 10012, USA

Communicated by D. Gries
Received 11 June 1991
Revised 24 February 1992

Abstract

Zhang, K., R. Statman and D. Shasha, On the editing distance between unordered labeled trees, Information Processing Letters 42 (1992) 133–139.

This paper considers the problem of computing the editing distance between unordered, labeled trees. We give efficient polynomial-time algorithms for the case when one tree is a string or has a bounded number of leaves. By contrast, we show that the problem is NP-complete even for binary trees having a label alphabet of size two.

Keywords: Computational complexity, unordered trees

1. Introduction

Unordered labeled trees are trees whose nodes are labeled and in which only ancestor relationships are significant (the left-to-right order among siblings is not significant). Such trees arise naturally in genealogical studies, for example, the genetic study of the tracking of diseases. For many such applications, it would be useful to compare unordered labeled trees by some meaningful distance metric. The editing distance metric, used with some success for ordered labeled trees [2], is a natural such metric. This paper presents algorithms and complexity results for a wide spectrum of assumptions concerning this problem.

2. Definitions

The definitions below are similar in style to those for ordered labeled trees in [6]. We will

Correspondence to: K. Zhang, Department of Computer Science, University of Western Ontario, London, Ontario, Canada N6A 5B7. Email: kzhang@csd.uwo.ca.

* Research supported by the National Sciences and Engineering Research Council of Canada grants OGP0046373 and the U.S. National Science Foundation under grant IRI-89-01699 and CCR-9103953 and by the U.S. Office of Naval Research under grants N00014-90-J-1110 and N00014-91-J-1472.

** Email: shasha@ca.nyu.edu.

omit the proofs for Lemma 1 and Theorem 1 since they are similar to those in [6] and can be found in [7].

2.1. *Editing operations and editing distance between unordered labeled trees*

We consider three kinds of operations. Changing a 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 then removing n. Inserting is the complement of deleting. This means that inserting n as the child of m will make n the parent of a *subset* (as opposed to a consecutive subsequence [6]) of the current children of m.

We represent an edit operation as $a \to b$, where a is either Λ or a label of a node in tree T_1 and b is either Λ or a label of a node in tree T_2. We call $a \to b$ a change operation if $a \neq \Lambda$ and $b \neq \Lambda$; a delete operation if $b = \Lambda$; and an insert operation if $a = \Lambda$.

Let S be a sequence $s_1, \ldots, s_k$ of edit operations. An S-derivation from A to B is a sequence of trees $A_0, \ldots, A_k$ such that $A = A_0$, $B = A_k$, and $A_{i-1} \to A_i$ via s_i for $1 \leqslant i \leqslant k$. Let γ be a cost function which assigns to each edit operation $a \to b$ a nonnegative real number $\gamma(a \to b)$.

We constrain γ to be a distance metric. That is, (i) $\gamma(a \to b) \geqslant 0$, $\gamma(a \to a) = 0$; (ii) $\gamma(a \to b) = \gamma(b \to a)$; and (iii) $\gamma(a \to c) \leqslant \gamma(a \to b) + \gamma(b \to c)$.

We extend γ to the editing operations sequence S by letting $\gamma(S) = \sum_{i=1}^{|S|} \gamma(s_i)$. Formally the distance between T_1 and T_2 is defined as:

$\delta(T_1, T_2) = \min_S\{\gamma(S) \mid S$ is an edit operation sequence taking T_1 to $T_2\}$.

δ is also a distance metric according to the definition of γ.

2.2. *Mappings*

The edit operations give rise to a mapping which is a graphical specification of what edit operations apply to each node in the two unordered labeled trees.

Suppose that we have a numbering for each tree. Let $T[i]$ be the ith node of tree T in the given numbering. Formally we define a triple (M, T_1, T_2) to be a mapping from T_1 to T_2, where M is any set of pair of integers (i, j) satisfying:

(1) $1 \leqslant i \leqslant |T_1|$, $1 \leqslant j \leqslant |T_2|$;
(2) 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),
  (b) $T_1[i_1]$ is an ancestor of $T_1[i_2]$ iff $T_2[j_1]$ is an ancestor of $T_2[j_2]$ (ancestor order preserved).

We will use M instead of (M, T_1, T_2) if there is no confusion. Let M be a mapping from T_1 to T_2. Let I and J be the sets of nodes, in T_1 and T_2, not in M. Then we can define the cost of M:

$$\gamma(M) = \sum_{(i,j) \in M} \gamma(T_1[i] \to T_2[j]) + \sum_{i \in I} \gamma(T_1[i] \to \Lambda) + \sum_{j \in J} \gamma(\Lambda \to T_2[j]).$$

The relation between a mapping and a sequence of editing operations is as follows:

Lemma 1. *Given S, a sequence $s_1, \ldots, s_k$ of edit operations from T_1 to T_2, there exists a mapping M from T_1 to T_2 such that $\gamma(M) \leqslant \gamma(S)$. Conversely, for any mapping M, there exists a sequence of editing operations such that $\gamma(S) = \gamma(M)$.*

Theorem 2. $\delta(T_1, T_2) = \min_M\{\gamma(M) \mid M$ *is a mapping from T_1 to T_2*$\}$.

3. Algorithmic results

In this section, we first study the problem of finding the editing distance (a minimum cost mapping) between a string and an unordered tree; and then between an unordered tree with k leaves and a general unordered tree.

For the purpose of numbering the nodes in an unordered tree, we can take any ordering of the tree. That is, we first fix an arbitrary order among the children of each interior node of the unordered tree, yielding an ordered tree T. We will then use a postorder numbering of the nodes in T. In the intermediate steps of our algorithm, we may have to consider unordered forests. We note that the definition of mappings for unordered forests is the same as for unordered trees.

Define the out-degree of a node in a given tree to be the number of children of the node. We then define the degree of a tree to be the maximum out-degree of any of its nodes. A string can be considered to be a degree one tree.

3.1. Notation

Let $T[i]$ be the ith node in the tree according to the left-to-right postorder numbering of tree T. Let $l(i)$ be the number of the leftmost leaf descendant of the subtree rooted at $T[i]$. When $T[i]$ is a leaf, $l(i) = i$. $T[l(i)..i]$ is the unordered forest resulting from removing $T[i]$ from the subtree rooted at $T[i]$. The distance between $T_1[l(i)..i]$ and $T_2[l(j)..j]$ is denoted by *dist*(i, j). If i or j is zero it represents the empty tree. We use $d(i, 0)$ to represent $\gamma(S[i] \to \Lambda)$, $d(0, j)$ to represent $\gamma(\Lambda \to T[j])$, and $d(i, j)$ to represent $\gamma(S[i] \to T[j])$.

3.2. Algorithm for the editing distance between a string and a tree

The tree will be denoted by T. We will refer to the string as S and assume that it represents the sequence $a_1 a_2 \ldots a_n$. In the postorder representation, $S[1]$ represents a_n, $S[n]$ represents a_1, and in general $S[i]$ represents a_{n-i+1}. That is, the postorder numbering reverses the order of the string. This implies that $S[l(i)..i] = S[1..i]$ represents $a_n a_{n-1} \ldots a_{n-i+1}$.

With these preliminaries out of the way, we can now present a few lemmas and then the algorithm.

Lemma 3. *The relative order among siblings of T does not influence the editing distance between S and T.*

Proof. By the ancestor-descendant constraint on mappings, we know that, in the best mapping between S and T, S can be mapped only to a path of T from the root to a leaf. All other nodes of the tree must be inserted. Since leaf-to-root paths are preserved by different orderings of T, the order of T does not matter. □

This lemma implies that we can solve this problem by the algorithm of Zhang and Shasha [6] for ordered trees by fixing an arbitrary order for T. The time complexity will be $O(|S| \cdot |T| \cdot depth(T))$. In the following we show that this can be improved to $O(|S| \cdot |T|)$.

Assume that $T[j]$ has $k \geqslant 0$ children, $T[j_1], T[j_2], \ldots, T[j_k]$.

Lemma 4. (1) $dist(0, 0) = 0$.
(2) $dist(i, 0) = d(i, 0) + dist(i-1, 0)$.
(3) $dist(0, j) = d(0, j) + \Sigma_{i=1}^{k} dist(0, t_i)$.

Proof. Statements (1) and (2) are obvious. As for (3), insertion of *subtree*(j) is equivalent to the insertions of *subtree*$(j_1), \ldots,$ *subtree*(j_k), and then the insertion of root $T[j]$. □

Lemma 5. *If $T[j]$ has no children, then*

$$dist(i, j) = \min \begin{cases} dist(i-1, j) + d(i, 0) \\ dist(i, 0) + d(0, j) \\ dist(i-1, 0) + d(i, j). \end{cases}$$

Proof. Consider the best mapping between $S[1..i]$ and $T[j]$. We have three cases.

(1) $S[i]$ is not in the best mapping. Then $dist(i, j) = dist(i-1, j) + d(i, 0)$.

(2) $T[j]$ is not in the best mapping. Since $T[j]$ has no child, $dist(i, j) = dist(i, 0) + d(0, j)$.

(3) Both $S[i]$ and $T[j]$ are in the best mapping. Then by the definition of mapping $S[i]$ and $T[j]$ must mapped to each other. Therefore, $dist(i, j) = dist(i-1, 0) + d(i, j)$.

The lemma follows. □

Lemma 6. *If $T[j]$ has $k \geqslant 1$ children, then*

$$dist(i, j) = \min \begin{cases} dist(i-1, j) + d(i, 0) \\ dist(0, j) \\ \quad + \min_{t=1}^{k} \{dist(i, j_t) \\ \qquad - dist(0, j_t)\} \\ d(i, j) + dist(0, j) - d(0, j) \\ \quad + \min_{t=1}^{k} \{dist(i-1, j_t) \\ \qquad - dist(0, j_t)\}. \end{cases}$$

Proof. Consider the best mapping between $S[1..i]$ and $T[l(j)..j]$. Again we have three cases.

(1) $S[i]$ is not in the mapping. Then $dist(i, j) = dist(i-1, j) + d(i, 0)$.

(2) $T[j]$ is not in the mapping. Then assume that $j_1, j_2, \ldots, j_k$ are the children of $T[j]$. The best mapping must map $S[1..i]$ to one of the subtrees rooted at $T[j]$'s children. Suppose that $S[1..i]$ is mapped to the subtree rooted at $T[j_t]$. Then the distance is the sum of the cost of mapping $S[1..i]$ to $T[l(j_t)..j_t]$, the cost of mapping null to $T[j]$ and the costs of inserting all the other subtrees of $T[j]$. Therefore,

$$dist(i, j) = dist(i, j_t) + d(0, j) + \sum_{i=1}^{t-1} dist(0, j_i) + \sum_{i=t+1}^{k} dist(0, j_i).$$

By Lemma 4, $dist(0, j) = d(0, j) + \sum_{i=1}^{k} dist(0, j_i)$. So $dist(i, j) = dist(i, j_t) + dist(0, j) - dist(0, j_t)$. Hence we have the following:

$$dist(i, j) = dist(0, j) + \min_{t=1}^{k} \{dist(i, j_t) - dist(0, j_t)\}.$$

(3) (Figure 1.) $S[i]$ and $T[j]$ are both in the best mapping. Then they must map to each other, by the ancestor-descendant constraints on mappings. In this case $S[1..i-1]$ must mapped to a subtree rooted at a child of $T[j]$. Suppose that it is mapped to subtree $T[j_t]$, then

$$\begin{aligned} dist(i, j) &= d(i, j) + dist(i-1, j_t) + \sum_{i=1}^{t-1} dist(0, j_i) + \sum_{i=t+1}^{k} dist(0, j_i) \\ &= d(i, j) + dist(i-1, j_t) + dist(0, j) - dist(0, j_t) - d(0, j). \end{aligned}$$

Therefore we have the following:

$$dist(i, j) = d(i, j) + dist(0, j) - d(0, j) + \min_{t=1}^{k} \{dist(i-1, j_t) - dist(0, j_t)\}$$

The lemma follows by combining the above three formulas. □

We are now ready to give our algorithm.

```
$dist(0, 0) = 0$;
For $i = 1$ to $n$
    $dist(i, 0) = d(i-1, 0) + d(i, 0)$;
For $j = 1$ to $m$
    (* $T[j]$ has $k \geq 0$ children $j_1, \ldots, j_k$ *)
    $dist(0, j) = d(0, j) + \sum_{t=1}^{k} dist(0, j_t)$
For $i = 1$ to $n$
    For $j = 1$ to $m$
        if $l(j) = j$ then (* $T[j]$ has no child *)
```

$$dist(i, j) = \min \begin{cases} dist(i-1, j) + d(i, 0) \\ dist(i, 0) + d(0, j) \\ dist(i-1, 0) + d(i, j) \end{cases}$$

```
        else (* $T[j]$ has $k \geq 1$ children $j_1, \ldots, j_k$ *)
```

$$dist(i, j) = \min \begin{cases} dist(i-1, j) + d(i, 0), \\ dist(0, j) + \min_{t=1}^{k} \{dist(i, j_t) - dist(0, j_t)\}, \\ d(i, j) + dist(0, j) - d(0, j) + \min_{t=1}^{k} \{dist(i-1, j_t) - dist(0, j_t)\} \end{cases}$$

Theorem 7. *The algorithm correctly calculates the distance between S and T in $O(|S| \cdot |T|)$ time.*

Proof. Correctness is immediate from Lemmas 4–6.

Consider the time complexity. To compute $dist(i, j)$, the time is bounded by $O(degree(T[j]))$. Therefore, the time of the algorithm is bounded by

$$\sum_{i=1}^{|S|} \sum_{j=1}^{|T|} O(degree(T[j]) + 1) = \left(\sum_{i=1}^{|S|} 1 \right) \times \left(\sum_{j=1}^{|T|} O(degree(T[j])) \right) = O(|S| \cdot |T|). \quad □$$

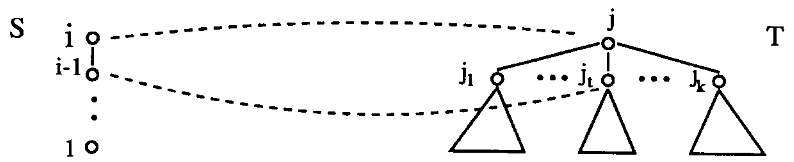

Fig. 1. If $S[i]$ maps to $T[j]$, and $S[1..i-1]$ maps to subtree rooted at $T[j_t]$. The cost of such a mapping includes the cost of inserting all subtrees of $T[j]$ aside from $T[j_t]$.

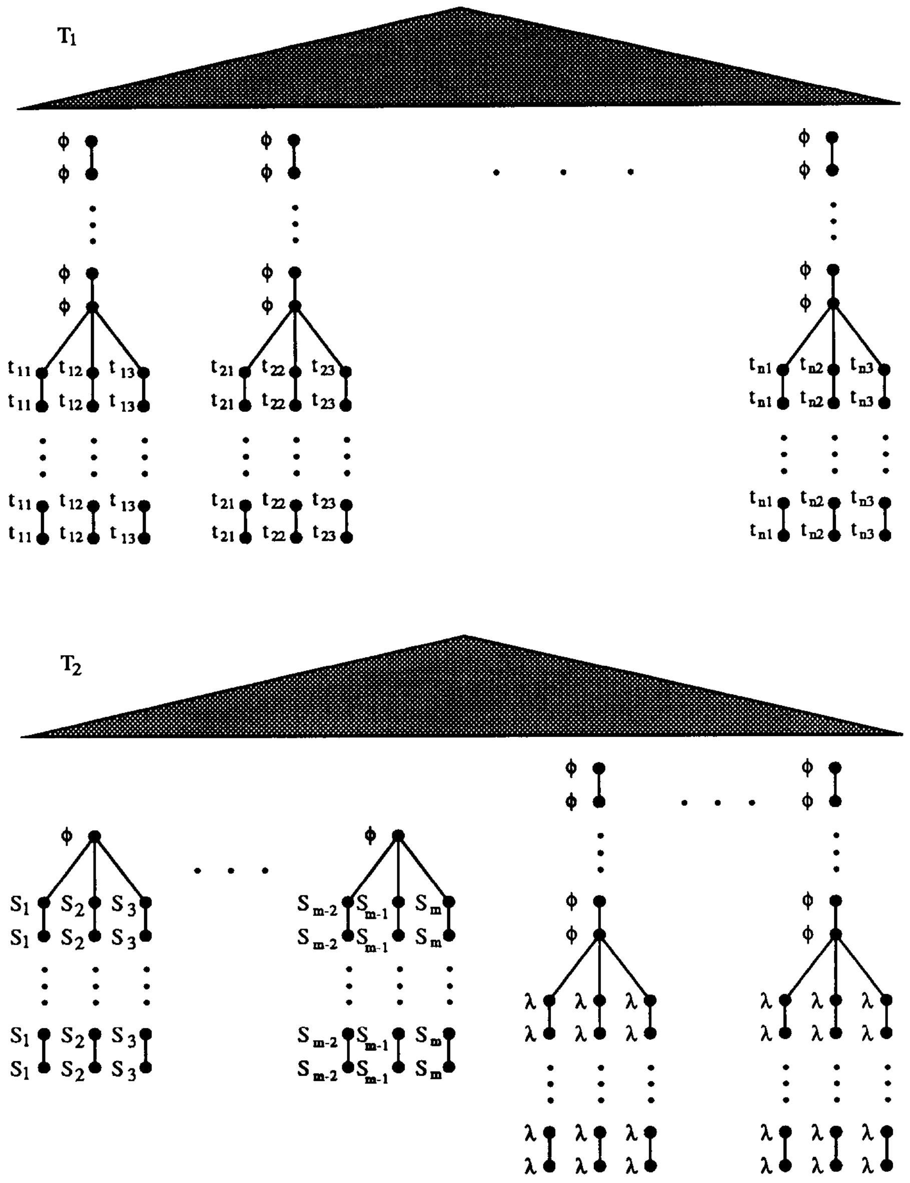

Fig. 2. Trees constructed from instance of exact 3-cover.

Strings are trees with a single leaf. Consider the problem of comparing an unordered tree with k leaves to a general unordered tree. Let us first consider the case where $k=2$. That is, there are two leaves in T_1 and an arbitrary tree T_2. Let i_0 be the node in T_1 with two children. Let its two children be i_1 and i_2. We can use the string-to-tree algorithm to compute the distance of the subtrees rooted at those children to tree T_2. To compute the distance from $T_1[l(i_0)..i_0]$ to $T_2[l(j)..j]$, one must compute the forest to forest distance $dist(l(i_0)..i_j, l(j)..j-1)$.

There are two subcases pertaining to the calculation of that term.

(1) Only one string is in the best mapping.

$$\begin{aligned} &dist(l(i_0)..i_2, l(j)..j-1)\\ &\quad = \min\{dist(i_1, l(j)..j-1) + dist(i_2, 0),\\ &\qquad\qquad dist(i_2, l(j)..j-1) + dist(i_1, 0)\}. \end{aligned}$$

(2) Suppose both strings are in the best mapping.

- If the two strings map to one subtree, then

$$\begin{aligned} &dist(l(i_0)..i_2, l(j)..j-1)\\ &\quad = dist(0, j) - d(0, j)\\ &\qquad + \min_{t=1}^{k}\{dist(l(i_0)..i_2, j_t) - dist(0, j_t)\}. \end{aligned}$$

- If each string maps to a different tree, then we use bipartite matching to find the best mapping.

From the above discussion, if we know all $dist(l(i_0)..i_2, l(j)..j-1)$, where $1 \leqslant j \leqslant |T_2|$, then we can proceed as in the string-to-tree algorithm. For fixed j, to compute $dist(l(i_0)..i_2, l(j)..j-1)$, we need

$$\begin{aligned} &2 + degree(T_2[j]) + 2! \cdot 3 \cdot (degree(T_2[j]) + 1)\\ &\quad = O(2! \cdot 3 \cdot degree(T_2[j])) \end{aligned}$$

time. The most expensive step is for bipartite matching, namely $2! \cdot 3 \cdot (degree(T_2[j]) + 1)$. Therefore, the total time for $dist(l(i_0)..i_2, l(j)..j)$, where $1 \leqslant j \leqslant |T_2|$, is $\sum_{j=1}^{T_2} 2! \cdot 3 \cdot degree(T_2[j]) = 2! \cdot 3 \cdot |T_2|$. Hence the time complexity of a two-leaf tree to a general tree is $O(|T_1| \cdot |T_2| + 2! \cdot 3 \cdot |T_2|)$.

If T_1 has k leaves, we have to consider the case where we have a subforest in that tree. The subforest consists of trees each of which has less than k leaves. There are at most 3^k such forests. Extending the matching techniques to groups of forests, we use bipartite matching to determine which group in T_1 maps to which different tree in T_2. The resulting complexity is

$$\begin{aligned} &O\big(|T_1| \cdot |T_2| + k! \cdot 3^k\\ &\qquad \cdot \left(k^3 + degree(T_2)^2\right) \cdot |T_2|\big). \end{aligned}$$

4. NP-completeness

In this section we show that the problem of computing the editing distance between unordered labeled fixed-degree $k \geqslant 2$ tree is NP-complete. For details of this section, we refer to [5,7].

We will reduce Exact **Cover by 3-Sets** [1] to computing unordered labeled tree editing distance. Given an instance of the exact 3-cover problem, let the set $S = \{s_1, s_2, \ldots, s_m\}$, where $m = 3q$. Let $T = \{C_1, C_2, \ldots, C_n\}$. Here each $C_i = \{t_{i_1}, t_{i_2}, t_{i_3}\}$, where $t_{i_j} \in S$. Without loss of generality, we assume that $n > q$.

We construct two trees as in Fig. 2. The top portion of each tree is represented by a triangle. Below the triangle are n subtrees for each tree. We use the triangle at the top to connect the n subtrees into a tree. It has ceiling of $\log_3(n)$ levels. It plays no further role in the construction. The total number of nodes in the triangle is less than n. We assume that all the nodes in the triangle have the same label, which may be arbitrary. We call a sequence of nodes with label ϕ, λ, s_i or t_{i_j} a segment. The length of each segment is $f_3(n) = 4n$.

Inspection of the figure shows that T_1 and T_2 can be constructed from an instance of exact cover by 3-sets in polynomial time.

In the following we will assume that each edit operation has unit cost, i.e. $\gamma(a \to b) = 1$ for all a,b such that $a \neq b$, and that T_1 and T_2 are trees as in Fig. 2.

Lemma 8. *Let M be a mapping between T_1 and T_2, if there are $d \geqslant 0$ nodes of T_2 not in mapping M, then $\gamma(M)(T_1, T_2) \geqslant 3(n-k)f_3(n) + k(f_3(n)-1) + d$.*

Lemma 9. *If there is an exact cover by 3-set, then $\delta(T_1, T_2) < 3(n-k)f_3(n) + k(f_3(n)-1) + f_3(n)$.*

Lemma 10. *If $\delta(T_1, T_2) < 3(n-k)f_3(n) + k(f_3(n)-1) + f_3(n)$, then there is an exact cover by 3-set.*

Theorem 11. *Computing the editing distance between unordered labeled degree three trees is* NP-*complete.*

For the editing distance between degree $k > 3$ trees, we can reduce the Exact Cover by k-sets to it. However, for degree two trees we cannot use the Exact Cover by 2-sets problem since it is in P. We can still use exact cover by 3-sets with a minor modification of the trees in Fig. 2. For any node with out-degree 3, let the node be a and its children be b, c, and d, that is (a(b)(c)(d)). We change it to two out-degree 2 nodes, namely (a(b)(a(c)(d))). Now if we replace $f_3(n) = 4n$ in Lemmas 8–10 by $f_2(n) = 6n$, we can prove the same results as in Lemmas 8–10 and Theorem 11. For the problem of computing the editing distance between a degree $d_1 \geqslant 2$ tree and a degree $d_2 \geqslant 2$ tree, we can similarly prove that it is NP-complete [7].

The previous results are based on the assumption that the size of the alphabet of the labels are not bounded. However, in applications the size of the alphabet is always a constant. When we fix the size of the alphabet, we can use different strings to encode different symbols. Therefore, even if the size of the alphabet is two all the previous results are still true [7].

References

[1] M.R. Garey and D.S. Johnson, *Computers and Intractability* (Freeman, New York, 1979).

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

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

[4] R.A. Wagner and M.J. Fisher, The string to string correction problem, *J. ACM* **21** (1974) 168–173.

[5] K. Zhang, The editing distance between trees: algorithms and applications, Ph.D. Thesis, Dept. of Computer Science, Courant Institute, 1989.

[6] K. Zhang and D. Shasha, Simple fast algorithms for the editing distance between trees and related problems, *SIAM J. Comput.* **18** (1989) 1245–1262.

[7] K. Zhang, R. Statman and D. Shasha, On the editing distance between unordered labeled trees, Tech. Rept. No. 289, Dept. of Computer Science, The University of Western Ontario, 1991.