Algorithmica (1996) 15: 205–222

Algorithmica

A Constrained Edit Distance Between Unordered Labeled Trees[1]

Kaizhong Zhang[2]

Abstract. This paper considers the problem of computing a constrained edit distance between unordered labeled trees. The problem of approximate unordered tree matching is also considered. We present dynamic programming algorithms solving these problems in sequential time $O(|T_1| \times |T_2| \times (deg(T_1) + deg(T_2)) \times \log_2(deg(T_1) + deg(T_2)))$. Our previous result shows that computing the edit distance between unordered labeled trees is NP-complete.

Key Words. Unordered trees, Constrained edit distance, Approximate tree matching.

1. Introduction. Unordered rooted 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 many fields: genealogical studies, e.g., determining genetic diseases based on ancestry tree patterns; computer vision, e.g., object representation and recognition [7], [8]; chemistry, e.g., studying the relationship between the structure of chemical compounds and their properties [10], [4]; file system, e.g., information retrieval in a unix file system. For many such applications, it would be useful to compare unordered labeled trees by some meaningful distance metric.

The edit distance metric, used with success [6] for ordered labeled trees [17], [18], is one such natural metric. A previous result has shown that computing the edit distance for unordered labeled trees is NP-complete [19] (in fact MAX SNP-hard [16]). Since unordered trees are important in many applications, one would like to find distances that can be efficiently computed.

Motivated by these results, this paper considers a constrained edit distance metric between unordered labeled trees. The constraint we add is a natural one, namely, disjoint subtrees be mapped to disjoint subtrees. Actually some applications would require this constraint (e.g., classification tree comparison) [11]. We present a polynomial-time algorithm to compute this constrained edit distance metric.

In Section 2 we review the previous NP-completeness and MAX SNP-hard results on the edit distance metric between unordered trees. In Section 3 we introduce the new distance metric between trees, which we call the constrained edit distance. In Section 4 we investigate the properties of the constrained edit distance metric. In Section 5 we present a dynamic programming algorithm to compute the constrained edit distance

[1] This research was supported by the Natural Sciences and Engineering Research Council of Canada under Grant No. OGP0046373.

[2] Department of Computer Science, University of Western Ontario, London, Ontario, Canada, N6A 5B7. kzhang@csd.uwo.ca.

Received October 16, 1993; revised May 18, 1994. Communicated by H. N. Gabow.

metric and analyse the complexity. In Section 6 we consider the approximate unordered tree-matching problem.

2. Preliminaries. In this section we first introduce some basic definitions and then review the previous results on unordered trees.

2.1. *Trees.* A *free tree* is a connected, acyclic, undirected graph. A *rooted tree* is a free tree in which one of the vertices is distinguished from the others. The distinguished vertex is called the root of the tree. We refer to a vertex of a rooted tree as a *node* of the tree. An *unordered tree* is just a rooted tree. We use the term unordered tree to distinguish it from the rooted, ordered tree defined below. An *ordered tree* is a rooted tree in which the children of each node are ordered. That is, if a node has k children, then we can designate them as the first child, the second child, ..., and the kth child.

Unless otherwise stated, all trees we consider in the paper are unordered labeled trees.

2.2. *Editing Operations.* 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 [17]) of the current children of m.

Suppose each node label is a symbol chosen from an alphabet Σ. Let λ, a unique symbol not in Σ, denote the null symbol. Following [9], [17], and [19], 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 T_2 be the tree that results from the application of an edit operation $a \to b$ to tree T_1; this is written $T_1 \to T_2$ via $a \to b$.

Let S be a sequence $s_1, \ldots, s_k$ of edit operations. An S-derivation from tree A to tree 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 $1 \leq i \leq k$. 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) $\gamma(a \to b) \geq 0$, $\gamma(a \to a) = 0$.
(ii) $\gamma(a \to b) = \gamma(b \to a)$.
(iii) $\gamma(a \to c) \leq \gamma(a \to b) + \gamma(b \to c)$.

We extend γ to the sequence of edit operations S by letting $\gamma(S) = \sum_{i=1}^{|S|} \gamma(s_i)$.

2.3. *Editing Distance and Edit Distance Mapping.* The results in this subsection are from [19]. We omit the proofs.

2.3.1. *Editing Distance.* The *edit distance* between two trees is defined [19] by considering the minimum cost edit operations sequence that transforms one tree to the other.

Formally the edit distance between T_1 and T_2 is defined as

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

2.3.2. *Editing Distance 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.

Given a tree, it is usually convenient to use a numbering to refer to the nodes of the tree. For an ordered tree T, the left-to-right postorder numbering or left-to-right preorder numbering are often used to number the nodes of T from 1 to $|T|$, the size of tree T. Since we are concerned with unordered trees, we can fix an arbitrary order for each of the node in the tree and then use left-to-right postorder numbering or left-to-right preorder numbering. We refer to the above as a *numbering* of the tree.

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_e, T_1, T_2) to be an *edit distance mapping* from T_1 to T_2, where M_e is any set of ordered pairs of integers (i, j) satisfying:

(0) $1 \le i \le |T_1|, 1 \le j \le |T_2|$; where $|T_k|$ is the size of tree T_k, $k = 1, 2$.
(1) For any pair of (i_1, j_1) and (i_2, j_2) in M_e:
  (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 use M_e instead of (M_e, T_1, T_2) if there is no confusion. Let M_e be an edit distance mapping from T_1 to T_2. Then we can define the cost of M_e:

$$\gamma(M_e) = \sum_{(i,j)\in M_e} \gamma(t_1[i] \to t_2[j]) + \sum_{\{i \mid \nexists\, j \text{ s.t. } (i,j)\in M_e\}} \gamma(t_1[i] \to \lambda) + \sum_{\{j \mid \nexists\, i \text{ s.t. } (i,j)\in M_e\}} \gamma(\lambda \to t_2[j]).$$

The relation between an edit distance mapping and a sequence of edit operations is as follows:

LEMMA 1. *Given S, a sequence $s_1, \ldots, s_k$ of edit operations from T_1 to T_2, an edit distance mapping M_e from T_1 to T_2 exists such that $\gamma(M_e) \le \gamma(S)$. Conversely, for any edit distance mapping M_e, a sequence of edit operations exists such that $\gamma(S) = \gamma(M_e)$.*

Based on the lemma, the following theorem states the relation between the edit distance and the edit distance mappings. This is whey we call this kind of mapping edit distance mapping.

THEOREM 1.

$$D_e(T_1, T_2) = \min_{M_e}\{\gamma(M_e) \mid M_e \text{ is an edit distance mapping from } T_1 \text{ to } T_2\}.$$

One of the results in [19] is that finding $D_e(T_1, T_2)$ is NP-complete even if the trees are binary trees with a label alphabet of size two. Kilpelainen and Mannila [3] showed

that even the inclusion problem for unordered trees in NP-complete. In fact we recently proved a stronger result [16]: that the problem of finding the minimum cost edit distance mapping (edit distance) between two unordered trees and the problem of finding the largest common subtree of two unordered trees are both MAX SNP-hard. This means that there is no polynomial-time approximation scheme (PTAS) for these problems unless P = NP [1]. Since unordered trees are important in many applications, it is desired to find distances that can be efficiently computed.

3. A Constrained Edit Distance Metric Between Unordered Trees. Our new distance metric is based on a constraint of the mappings allowed between two trees. The intuitive idea is that two separate subtrees of T_1 should be mapped to two separate subtrees of T_2. This idea was proposed by Tanaka and Tanaka [11] in their definition for a structure preserving mapping between two ordered labeled trees although the definition in [11] does not capture the idea precisely. Tanaka and Tanaka [11] also showed that in some applications (e.g., classification tree comparison) this kind of mapping is more meaningful than edit distance mapping. In a separate result [15], we developed an algorithm for the constrained edit distance between ordered labeled trees with time complexity $O(|T_1||T_2|)$. This paper extends the definition from ordered trees to unordered trees.

3.1. *Constrained Edit Distance Mappings.* Suppose again that we have a numbering for each tree. Let $t[i]$ be the ith node of tree T in the given numbering. Let $T[i]$ be the subtree rooted at $t[i]$ and let $F[i]$ be the unordered forest obtained by deleting $t[i]$ from $T[i]$.

Formally we define a triple (M, T_1, T_2) to be a *constrained edit distance mapping* from T_1 to T_2, where M is any set of ordered pairs of integers (i, j) satisfying:

(1) M is an edit distance mapping.
(2) For any triple (i_1, j_1), (i_2, j_2), and (i_3, j_3) in M let $t_1[I]$ be the $\text{lca}(t_1[i_1], t_1[i_2])$ and let $t_2[J]$ be the $\text{lca}(t_2[j_1], t_2[j_2])$, where lca represents least common ancestor. $t_1[I]$ is a proper ancestor of $t_1[i_3]$ iff $t_2[J]$ is a proper ancestor of $t_2[j_3]$.

This definition adds condition (2) to the definition of the edit distance mapping. We examine the meaning of the constrained edit distance mappings. Suppose $t_1[I]$ is a descendant of $t_1[i_3]$, then $t_1[i_1]$ and $t_1[i_2]$ must be descendants of $t_1[i_3]$. By condition (1) $t_2[j_1]$ and $t_2[j_2]$ must be descendants of $t_2[j_3]$. This in turn means that $t_2[J]$ is a descendant of $T_2[j_3]$. Therefore condition (1) implies that $t_1[I]$ is a descendant of $t_1[i_3]$ if and only if $t_2[J]$ is a descendant of $t_2[j_3]$. Condition (2) adds the constraint that $t_1[I]$ is a proper ancestor of $t_1[i_3]$ if and only if $t_2[J]$ is a proper ancestor of $t_2[j_3]$. Suppose now that neither $t_1[I]$ nor $t_1[i_3]$ is an ancestor of the other, then by (1) and (2) neither $t_2[J]$ nor $t_2[j_3]$ is an ancestor of the other.

Consider two separate subtrees $T_1[i_1]$ and $T_1[i_2]$ such that neither $t_1[i_1]$ nor $t_1[i_2]$ is an ancestor of the other. Given a mapping M, let

$$M_k = \{n \mid (m, n) \in M \text{ and } t_1[m] \text{ is a node of } T_1[i_k]\}$$

where $k = 1, 2$. Let $t_2[j_k] = \text{lca}(M_k)$ for $k = 1, 2$. $T_2[j_k]$ can be considered as the

image of $T_1[i_k]$ under mapping M. With our definition of the constrained edit distance mapping, neither $t_2[j_1]$ nor $t_2[j_2]$ is an ancestor of the other. This coincides with the intuitive idea that separate subtrees of T_1 should be mapped to separate subtrees of T_2.

We use M instead of (M, T_1, T_2) if there is no confusion. Let M be a constrained edit distance mapping from T_1 to T_2; the cost of M, $\gamma(M)$, is well defined since M is also an edit distance mapping.

Constrained edit distance mappings can be composed. Let M_1 be a constrained edit distance mapping from T_1 and T_2, and let M_2 be a constrained edit distance mapping from T_2 to T_3. Define

$$M_1 \circ M_2 = \{(i, j) \mid \exists k \text{ s.t. } (i, k) \in M_1 \text{ and } (k, j) \in M_2\}.$$

LEMMA 2.

(1) *$M_1 \circ M_2$ is a constrained edit distance mapping between T_1 and T_3.*
(2) $\gamma(M_1 \circ M_2) \leq \gamma(M_1) + \gamma(M_2)$.

PROOF. (1) follows from the definition of constrained edit distance mappings. We check condition (2) only. Let (i_1, j_1), (i_2, j_2), and (i_3, j_3) be in $M_1 \circ M_2$. By the definition of $M_1 \circ M_2$, there are k_1, k_2, and k_3 such that (i_1, k_1), (i_2, k_2), and (i_3, k_3) are in M_1 and (k_1, j_1), (k_2, j_2), and (k_3, j_3) are in M_2. Let I be the lca(i_1, i_2), let K be the lca(k_1, k_2), and let J be the lca(j_1, j_2). By the definition of M_1 and M_2, I is a proper ancestor of i_3 iff K is a proper ancestor of k_3. Moreover, K is a proper ancestor of k_3 iff J is a proper ancestor of j_3. Therefore I is a proper ancestor of i_3 iff J is a proper ancestor of j_3.

(2) Let M_1 be the constrained edit distance mapping from T_1 to T_2, let M_2 be the constrained edit distance mapping from T_2 to T_3, and let $M_1 \circ M_2$ be the composed constrained edit distance mapping from T_1 to T_3. Three general situations occur: $(i, j) \in M_1 \circ M_2, i \notin M_1$, or $j \notin M_2$. In each case this corresponds to an edit operation $\gamma(x \to y)$ where x and y may be nodes or may be λ. In all such cases, the triangle inequality on the distance metric γ ensures that $\gamma(x \to y) \leq \gamma(x \to z) + \gamma(z \to y)$. □

3.2. *A Constrained Edit Distance Between Trees.* We can now define a dissimilarity measure between T_1 and T_2 as:

$$D(T_1, T_2) = \min_M \{\gamma(M) \mid M \text{ is a constrained edit distance mapping from } T_1 \text{ to } T_2\}.$$

In fact this dissimilarity measure is a distance metric.

THEOREM 2.

(1) $D(T, T) = 0$.
(2) $D(T_1, T_2) = D(T_2, T_1)$.
(3) $D(T_1, T_3) \leq D(T_1, T_2) + D(T_2, T_3)$.

PROOF. (1) and (2) follow directly from the definition of the constrained edit distance mapping. For (3), consider the minimum cost constrained edit distance mappings M_1

between T_1 and T_2 and M_2 between T_2 and T_3. It is easy to see the following:

$$D(T_1, T_3) \leq \gamma(M_1 \circ M_2) \leq \gamma(M_1) + \gamma(M_2) = D(T_1, T_2) + D(T_2, T_3). \qquad \square$$

We call this distance metric the constrained edit distance. The relation between the constrained edit distance metric D and the edit distance metric D_e is $D_e(T_1, T_2) \leq D(T_1, T_2)$. The reason is that any constrained edit distance mapping is always an edit distance mapping.

4. Properties of the Constrained Edit Distance. In this section we present several lemmas which are the basis for the algorithm in the next section. We use θ to represent the empty tree.

LEMMA 3. *Let $t_1[i_1], t_1[i_2], \ldots, t_1[i_{n_i}]$ be the children of $t_1[i]$ and let $t_2[j_1], t_2[j_2], \ldots, t_2[j_{n_j}]$ be the children of $t_2[j]$. Then*

$$D(\theta, \theta) = 0,$$

$$D(F_1[i], \theta) = \sum_{k=1}^{n_i} D(T_1[i_k], \theta), \qquad D(T_1[i], \theta) = D(F_1[i], \theta) + \gamma(t_1[i] \to \lambda),$$

$$D(\theta, F_2[j]) = \sum_{k=1}^{n_j} D(\theta, T_2[j_k]), \qquad D(\theta, T_2[j]) = D(\theta, F_2[j]) + \gamma(\lambda \to t_2[j]).$$

PROOF. Trivial. $\square$

LEMMA 4. *Let $t_1[i_1], t_1[i_2], \ldots, t_1[i_{n_i}]$ be the children of $t_1[i]$ and let $t_2[j_1], t_2[j_2], \ldots, t_2[j_{n_j}]$ be the children of $t_2[j]$. Then*

$$D(T_1[i], T_2[j]) = \min \begin{cases} D(\theta, T_2[j]) + \min\limits_{1 \leq t \leq n_j} \{D(T_1[i], T_2[j_t]) - D(\theta, T_2[j_t])\}, \\ D(T_1[i], \theta) + \min\limits_{1 \leq s \leq n_i} \{D(T_1[i_s], T_2[j]) - D(T_1[i_s], \theta)\}, \\ D(F_1[i], F_2[j]) + \gamma(t_1[i] \to t_2[j]). \end{cases}$$

PROOF. Consider the minimum cost constrained edit distance mapping M between $T_1[i]$ and $T_2[j]$. There are four cases:

(1) $i \in M$ and $j \notin M$.
(2) $i \notin M$ and $j \in M$.
(3) $i \in M$ and $j \in M$.
(4) $i \notin M$ and $j \notin M$.

Case 1. Let (i, t) in M. Since $j \notin M$, t must be a node in $F_2[j]$. Let $t_2[j_t]$ be the child of $t_2[j]$ on the path from $t_2[t]$ to $t_2[j]$. Thus $D(T_1[i], T_2[j]) = D(T_1[i], T_2[j_t]) +$

$D(\theta, T_2[j_1]) + \cdots + D(\theta, T_2[j_{t-1}]) + D(\theta, T_2[j_{t+1}]) + \cdots + D(\theta, T_2[j_{n_j}]) + \gamma(\lambda, t_2[j])$. Since $D(\theta, T_2[j]) = \gamma(\lambda, t_2[j]) + \sum_{k=1}^{n_j} D(\theta, T_2[j_k])$, we can rewrite the right-hand side of the formula as $D(\theta, T_2[j]) + D(T_1[i], T_2[j_t]) - D(\theta, T_2[j_t])$. Since the range of k is from 1 to n_j, we take the minimum of these corresponding costs.

Case 2. Similar to case 1.

Case 3. Since $i \in M$ and $j \in M$, by the condition of constrained edit distance mapping, (i, j) must be in M. Since $M - \{(i, j)\}$ is a constrained edit distance mapping between $F_1[i]$ and $F_2[j]$, and for any constrained edit distance mapping M' between $F_1[i]$ and $F_2[j]$, $(i, j) \cup M'$ is a constrained edit distance mapping between $T_1[i]$ and $T_2[j]$, we know $D(T_1[i], T_2[j]) = D(F_1[i], F_2[j]) + \gamma(t_1[i], t_2[j])$.

Case 4. Similar to Case 3. The formula would be $D(T_1[i], T_2[j]) = D(F_1[i], F_2[j]) + \gamma(t_1[i], \lambda) + \gamma(\lambda, t_2[j])$. Since $\gamma(t_1[i], t_2[j]) \leq \gamma(t_1[i], \lambda) + \gamma(\lambda, t_2[j])$, we do not have to include this case in our final formula. □

Before we proceed to the next lemma we need the following definition.

A *restricted mapping* $RM(i, j)$ between $F_1[i]$ and $F_2[j]$ is defined as follows:

(1) $RM(i, j)$ is a constrained edit distance mapping between $F_1[i]$ and $F_2[j]$.
(2) If (l, k) is in $RM(i, j)$, $t_1[l]$ is in $T_1[i_s]$, and $t_2[k]$ is in $T_2[j_t]$, then, for any (l_1, k_1) in $RM(i, j)$, $t_1[l_1]$ is in $T_1[i_s]$ if and only if $t_2[k_1]$ is in $T_2[j_t]$.

Since a restricted mapping is a constrained edit distance mapping, the cost of a restricted mapping is well defined. The meaning of the restriction is that nodes from a subtree $T_1[i_s]$ can only map to nodes of one subtree $T_2[j_t]$ and vice versa. It turns out that, although there are exceptions, this is the general case when we consider the constrained edit distance mapping between $F_1[i]$ and $F_2[j]$.

LEMMA 5. *Let* $t_1[i_1], t_1[i_2], \ldots, t_1[i_{n_i}]$ *be the children of* $t_1[i]$ *and let* $t_2[j_1], t_2[j_2], \ldots, t_2[j_{n_j}]$ *be the children of* $t_2[j]$. *Then*

$$D(F_1[i], F_2[j]) = \min \begin{cases} D(\theta, F_2[j]) + \min\limits_{1 \leq t \leq n_j} \{D(F_1[i], F_2[j_t]) - D(\theta, F_2[j_t])\}, \\ D(F_1[i], \theta) + \min\limits_{1 \leq s \leq n_i} \{D(F_1[i_s], F_2[j]) - D(F_1[i_s], \theta)\}, \\ \min_{RM(i,j)} \gamma(RM(i, j)). \end{cases}$$

PROOF. We consider the minimum-cost constrained edit distance mapping M between $F_1[i]$ and $F_2[j]$. There are four cases.

Case 1. There is a $1 \leq t \leq n_j$ such that if $(k, l) \in M$, then $t_2[l]$ is a node in subtree $T_2[j_t]$; and there are (k_1, l_1) and (k_2, l_2) in M such that $t_1[k_1]$ is a node in $T_1[i_{s_1}]$ and $t_1[k_2]$ is a node in $T_1[i_{s_2}]$, where $1 \leq s_1 \neq s_2 \leq n_i$. Note that in this case j_t cannot be in M. This is similar to case 1 in Lemma 4, and hence we have the following formula: $D(F_1[i], F_2[j]) = D(\theta, F_2[j]) + \min_{1 \leq t \leq n_j} \{D(F_1[i], F_2[j_t]) - D(\theta, F_2[j_t])\}$.

Case 2. There is a $1 \leq s \leq n_i$ such that if $(k, l) \in M$, then $t_1[k]$ is a node in subtree $T_1[i_s]$; and there are (k_1, l_1) and (k_2, l_2) in M such that $t_2[l_1]$ is a node in $T_2[j_{t_1}]$ and $t_2[l_2]$ is a node in $T_2[j_{t_2}]$, where $1 \leq t_1 \neq t_2 \leq n_j$. This is similar to case 1.

Case 3. There are s and t such that if $(k, l) \in M$, then $t_1[k]$ is node in $T_1[i_s]$ and $t_2[l]$ is node in $T_2[i_t]$. In this case M is a restricted mapping and therefore $D(F_1[i], F_2[j]) = \min_{RM(i,j)} \gamma(RM(i, j))$.

Case 4. There are (k_1, l_1), (k_2, l_2), (x_1, y_1), (x_2, y_2) in M such that $t_1[k_1]$ is a node in $T_1[i_{s_1}]$, $t_1[k_2]$ is a node in $T_1[i_{s_2}]$, $t_2[y_1]$ is a node in $T_2[i_{t_1}]$, and $t_2[y_2]$ is a node in $T_2[i_{t_2}]$, where $1 \leq s_1 \neq s_2 \leq n_i$ and $1 \leq t_1 \neq t_2 \leq n_j$. We show that in this case the constrained edit distance mapping M is a restricted mapping between $F_1[i]$ and $F_2[j]$.

Suppose this is not true. Then, without loss of generality, we assume that there are (a_1, b_1) and (a_2, b_2) in M such that $t_1[a_1]$ and $t_1[a_2]$ belong to the same subtree and $t_2[b_1]$ and $t_2[b_2]$ belong to different subtrees. Let $(a_3, b_3) \in M$ such that $t_1[a_3]$ and $t_1[a_1]$ belong to different subtrees of $F_1[i]$. Now consider $\text{lca}(t_1[a_1], t_1[a_2])$ and $t_1[a_3]$. Since $t_1[a_1]$ and $t_1[a_2]$ belong to the same subtree which is different from the subtree $t_1[a_3]$ belongs to, we know that $\text{lca}(t_1[a_1], t_1[a_2])$ and $t_1[a_3]$ do not have the ancestor or descendant relationship. However, if we consider $\text{lca}(t_2[b_1], t_2[b_2])$ and $t_2[b_3]$, it is easy to see that $\text{lca}(t_2[b_1], t_2[b_2]) = t_2[j]$, which is an ancestor of $t_2[b_3]$. This means that M is not a valid constrained edit distance mapping. Contradiction. □

From Lemma 5, in order to compute $D(F_1[i], F_2[j])$ we have to compute $\min_{RM(i,j)} \gamma(RM(i, j))$. From the definition of the restricted mapping, we know that nodes of one subtree of $F_1[i]$ can only map to nodes of one subtree of $F_2[j]$. This means that if $T_1[i_s]$ is involved in the mapping, then it will map to $T_2[j_t]$ for some $1 \leq t \leq n_j$; if it is not involved in the mapping, then it will map to an empty tree. This gives a kind of matching between subtrees of $F_1[i]$ and subtrees of $F_2[j]$. The next two lemmas establish the relationship between $\min_{RM(i,j)} \gamma(RM(i, j))$ and $D(T_1[i_s], T_2[j_t])$, where $1 \leq s \leq n_i$ and $1 \leq t \leq n_j$. We need the following definitions in the next two lemmas.

Let $I = \{i_1, i_2, \ldots, i_{n_i}\}$, $J = \{j_1, j_2, \ldots, j_{n_j}\}$, $A = \{a_1, a_2, \ldots, a_{n_j}\}$, and $B = \{b_1, b_2, \ldots, b_{n_i}\}$. We use A and B to represent empty trees. Let $L = I \cup A$ and $R = J \cup B$, we define a weighted bipartite graph $G(i, j) = (V, E)$ as follows:

(1) $V = L \cup R$.
(2) $E = L \times R$, where $\times$ stands for the Cartesian product.
(3) $w(u, v) = D(T_1[u], T_2[v])$ if $u \in I$ and $v \in J$,
$w(u, v) = D(T_1[u], \theta)$ if $u \in I$ and $v \in B$,
$w(u, v) = D(\theta, T_2[v])$ if $u \in A$ and $v \in J$,
$w(u, v) = 0$ if $u \in A$ and $v \in B$.

A matching on $G(i, j)$ is a subset of edges $M \subset E$ such that, for all vertices $v \in V$, at most one edge of M is incident on v. The size of M, $|M|$, is the number of edges in M. A maximum matching is a matching with maximum cardinality, that is, a matching M such that, for any matching M', we have $|M| \geq |M'|$. Let $MM(i, j)$ be a maximum matching on $G(i, j)$, then $|MM(i, j)| = n_i + n_j$.

Let $MM(i, j)$ be a maximum matching on $G(i, j)$. The cost of $MM(i, j)$, the summation of the weights of the edges in $MM(i, j)$, is defined as follows:

$$\begin{aligned}\gamma(MM(i,j)) &= \sum_{(u,v)\in MM(i,j)} w(u,v) \\ &= \sum_{(u,v)\in MM \text{ and } u\in I \text{ and } v\in J} D(T_1[u], T_2[v]) \\ &\quad + \sum_{(u,v)\in MM \text{ and } u\in I \text{ and } v\in B} D(T_1[u], \theta) \\ &\quad + \sum_{(u,v)\in MM \text{ and } u\in A \text{ and } v\in J} D(\theta, T_2[v]).\end{aligned}$$

LEMMA 6. $\min_{RM(i,j)} \gamma(RM(i, j)) = \min_{MM(i,j)} \gamma(MM(i, j))$.

PROOF. Given a restricted mapping $RM(i, j)$, we define a maximum matching $MM(i, j)$ on $G(i, j)$ as follows:

$$MM(i, j) = H_{MM} \cup I_{MM} \cup J_{MM} \cup K_{MM},$$

where

$$\begin{aligned}H_{MM} &= \{(i_s, j_t) \mid \text{there is } (k, l) \in RM(i, j) \\ &\qquad \text{s.t. } t_1[k] \ (t_2[l]) \text{ is a node in } T_1[i_s] \ (T_2[j_t])\}, \\ I_{MM} &= \{(i_s, b_s) \mid \text{there is no } j_t \text{ s.t. } (i_s, j_t) \in H_{MM}\}, \\ J_{MM} &= \{(a_t, j_t) \mid \text{there is no } i_s \text{ s.t. } (i_s, j_t) \in H_{MM}\}, \\ K_{MM} &= \{(a_s, b_t) \mid (i_t, j_s) \in H_{MM}\}.\end{aligned}$$

By the definition of a restricted mapping this is indeed a maximum matching. Furthermore, it is easy to see that $\gamma(MM(i, j)) \leq \gamma(RM(i, j))$.

On the other hand, given a maximum matching $MM(i, j)$, we can construct a restricted mapping $RM(i, j)$ as follows:

$$RM(i, j) = \left\{ (k, l) \;\middle|\; \begin{array}{l}\text{there exists } (i_s, j_t) \in MM(i, j) \text{ s.t. } (k, l) \text{ is in the} \\ \text{minimum cost constrained edit distance mapping} \\ \text{between } T_1[i_s] \text{ and } T_2[j_t]\end{array} \right\}.$$

It is clear that this is indeed a restricted mapping. Therefore, $\gamma(RM(i, j)) \leq \gamma(MM(i, j))$.

Hence $\min_{RM(i,j)} \gamma(RM(i, j)) = \min_{MM(i,j)} \gamma(MM(i, j))$. □

Combining Lemmas 5 and 6, we have proved the following lemma.

LEMMA 7. *Let* $t_1[i_1], t_1[i_2], \ldots, t_1[i_{n_i}]$ *be the children of* $t_1[i]$ *and let* $t_2[j_1], t_2[j_2], \ldots, t_2[j_{n_j}]$ *be the children of* $t_2[j]$. *Then*

$$D(F_1[i], F_2[j]) = \min \begin{cases} D(\theta, F_2[j]) + \min\limits_{1 \le t \le n_j} D(F_1[i], F_2[j_t]) - D(\theta, F_2[j_t]), \\ D(F_1[i], \theta) + \min\limits_{1 \le s \le n_i} D(F_1[i_s], F_2[j]) - D(F_1[i_s], \theta), \\ \min\limits_{MM(i,j)} \gamma(MM(i, j)). \end{cases}$$

5. Algorithm and Complexity. We first consider how to compute

$$\min_{MM(i,j)} \gamma(MM(i, j))$$

and then present our simple algorithm.

5.1. *Algorithm.* From the definition of $MM(i, j)$ and $\gamma(MM(i, j))$, it is clear that this problem is the minimum cost maximum bipartite matching problem. Therefore we can use the weighted maximum matching algorithm to solve this problem. However, since we have many empty trees in graph $G(i, j)$, this will result in redundant computation. Instead, we reduce this problem directly to the minimum cost maximum flow problem by adding only two empty trees, one to $F_1[i]$ and the other to $F_2[j]$.

Given $F_1[i]$ and $F_2[j]$, let $I = \{i_1, i_2, \ldots, i_{n_i}\}$ and $J = \{j_1, j_2, \ldots, j_{n_j}\}$, where i_k, $1 \le k \le n_i$, represents tree $T_1[i_k]$, and j_l, $1 \le l \le n_j$, represents tree $T_2[j_l]$.

We construct a graph $G = (V, E)$ as follows:

vertex set: $V = \{s, t, e_i, e_j\} \cup I \cup J$, where s is the source, t is the sink, and e_i and e_j represent two empty trees;
edge set: $[s, i_k]$, $[s, e_i]$, $[j_l, t]$, $[e_j, t]$ with cost zero, $[i_k, j_l]$ with cost $D(T_1[i_k], T_2[j_l])$, $[e_i, j_l]$ with cost $D(\theta, T_2[j_l])$, $[i_k, e_j]$ with cost $D(T_1[i_k], \theta)$, and $[e_i, e_j]$ with cost zero. All the edges have capacity one except $[s, e_i]$, $[e_i, e_j]$, and $[e_j, t]$ whose capacities are n_j, $\max\{n_i, n_j\} - \min\{n_i, n_j\}$, and n_i, respectively.

G is a network with integer capacities, nonnegative costs, and the maximum flow $f^* = n_i + n_j$ (see Figure 1). Let $cost(G)$ be the cost of the minimum cost on maximum flows of G, where only the costs on positive flows are considered. The following lemma states the relation between $cost(G)$ and $\min_{MM(i,j)} \gamma(MM(i, j))$, which enables us to use the minimum cost flow algorithm to compute $\min_{MM(i,j)} \gamma(MM(i, j))$.

LEMMA 8. $cost(G) = \min_{MM(i,j)} \gamma(MM(i, j))$.

PROOF. Given an $MM(i, j)$, $\gamma(MM(i, j))$ represents the cost of the following maximum flow on G: for any $(i_s, j_t) \in MM(i, j)$ the flow on edge $[i_s, j_t]$ is one; for any $(i_s, b_t) \in$

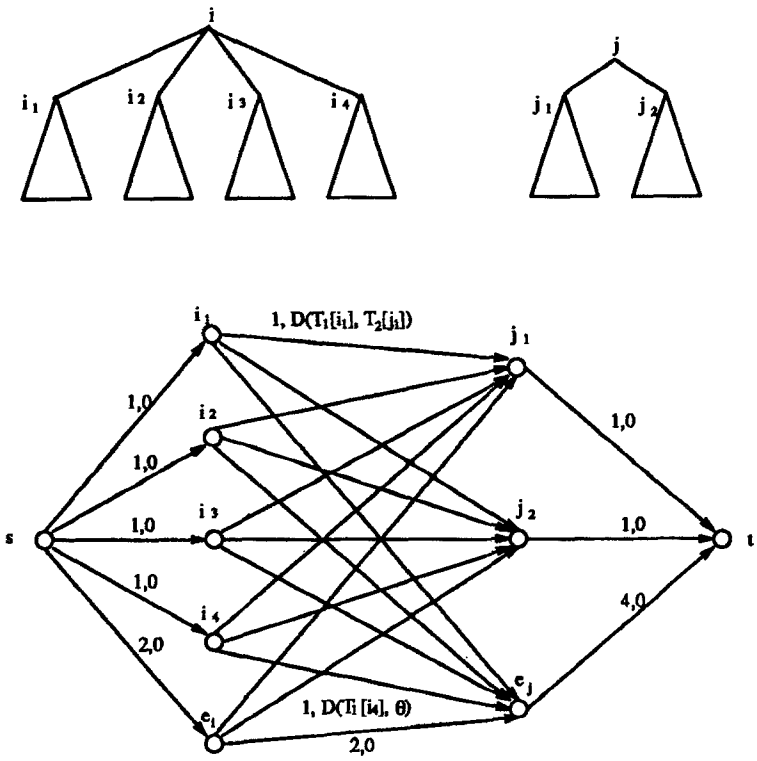

Fig. 1. Reduction to the minimum cost flow problem.

$MM(i, j)$ the flow on edge $[i_s, e_j]$ is one; for any $(a_s, j_t) \in MM(i, j)$ the flow on edge $[e_i, j_t]$ is one; the flow on edge

$$[e_i, e_j] = |\{(u, v) \mid (u, v) \in MM(i, j) \text{ and } u \in A \text{ and } v \in B\}|;$$

the flows on edge $[s, i_k]$ and edge $[j_l, t]$ are one; the flow on edge $[s, e_i]$ is n_j; the flow on edge $[e_j, t]$ is n_i; and all the flows on the other edges are zero.

On the other hand, given a maximum flow on G, $cost(G)$ represents the cost of the following maximum matching on $G(i, j)$:

$$\begin{aligned} MM(i, j) = & \{(i_s, j_t), (a_t, b_s) \mid \text{the flow on edge } [i_s, j_t] \text{ is one}\} \\ & \cup \{(i_s, b_s) \mid \text{the flow on edge } [i_s, e_j] \text{ is one}\} \\ & \cup \{(a_t, j_t) \mid \text{the flow on edge } [e_i, j_t] \text{ is one}\}. \end{aligned}$$

Therefore the cost of the minimum cost maximum flow is exactly $\min_{MM(i,j)} \gamma(MM(i, j))$. □

We are now ready to give our algorithm.

Input: T_1 and T_2.
Output: $D(T_1[i], T_2[j])$, where $1 \le i \le |T_1|$ and $1 \le j \le |T_2|$.

$D(\theta, \theta) = 0$;
for $i = 1$ to $|T_1|$

$$D(F_1[i], \theta) = \sum_{k=1}^{n_i} D(T_1[i_k], \theta)$$

$$D(T_1[i], \theta) = D(F_1[i], \theta) + \gamma(t_1[i] \to \lambda)$$

for $j = 1$ to $|T_2|$

$$D(\theta, F_2[j]) = \sum_{k=1}^{n_j} D(\theta, T_2[j_k])$$

$$D(\theta, T_2[j]) = D(\theta, F_2[j]) + \gamma(\lambda \to t_2[j])$$

for $i = 1$ to $|T_1|$
for $j = 1$ to $|T_2|$

$$D(F_1[i], F_2[j]) = \min \begin{cases} D(\theta, F_2[j]) + \min\limits_{1 \le t \le n_j} \{D(F_1[i], F_2[j_t]) - D(\theta, F_2[j_t])\}, \\ D(F_1[i], \theta) + \min\limits_{1 \le s \le n_i} \{D(F_1[i_s], F_2[j]) - D(F_1[i_s], \theta)\}, \\ \min\limits_{MM(i,j)} \gamma(MM(i, j)). \end{cases}$$

$$D(T_1[i], T_2[j]) = \min \begin{cases} D(\theta, T_2[j]) + \min\limits_{1 \le t \le n_j} \{D(T_1[i], T_2[j_t]) - D(\theta, T_2[j_t])\}, \\ D(T_1[i], \theta) + \min\limits_{1 \le s \le n_i} \{D(T_1[i_s], T_2[j]) - D(T_1[i_s], \theta)\}, \\ D(F_1[i], F_2[j]) + \gamma(t_1[i] \to t_2[j]). \end{cases}$$

5.2. *Complexity.* The complexity of computing $D(T_1[i], T_2[j])$ is, by Lemma 4, bounded by $O(n_i + n_j)$. The complexity of computing $D(F_1[i], F_2[j])$ is bounded by $O(n_i + n_j)$ plus the complexity of the minimum cost maximum flow computation.

Our graph G is a graph with integer capacities, nonnegative edge costs, and maximum flow $f^* = n_i + n_j$. The complexity of finding minimum cost maximum flow for such a graph with n vertices and m edges is $O(m|f^*| \log_{(2+m/n)} n) \le O(m|f^*| \log_2 n)$ [12]. For our graph, $n = n_i + n_j + 4$ and $m = n_i \times n_j + 2n_i + 2n_j + 3$; therefore the complexity is bounded by $O(n_i \times n_j \times (n_i + n_j) \times \log_2(n_i + n_j))$.

Hence for any pair i and j, the complexity of computing $D(T_1[i], T_2[j])$ and $D(F_1[i], F_2[j])$ is bounded by $O(n_i \times n_j \times (n_i + n_j) \times \log_2(n_i + n_j))$. Therefore the complexity of our algorithm is

$$\sum_{i=1}^{|T_1|} \sum_{j=1}^{|T_2|} O(n_i \times n_j \times (n_i + n_j) \times \log_2(n_i + n_j))$$

$$\le \sum_{i=1}^{|T_1|} \sum_{j=1}^{|T_2|} O(n_i \times n_j \times (deg(T_1) + deg(T_2)) \times \log_2(deg(T_1) + deg(T_2)))$$

$$\leq O\left((deg(T_1) + deg(T_2)) \times \log_2(deg(T_1) + deg(T_2)) \times \sum_{i=1}^{|T_1|} n_i \times \sum_{j=1}^{|T_2|} n_j\right)$$

$$\leq O(|T_1| \times |T_2| \times (deg(T_1) + deg(T_2)) \times \log_2(deg(T_1) + deg(T_2))).$$

6. Approximate Unordered Tree Matching. Approximate unordered tree matching is a natural extension of approximate string matching [5], [13], [2] and approximate ordered tree matching [17].

The approximate-string-matching problem is as follows. Given two strings *TEXT* and *PAT*, the problem is to compute, for each i, $SD[i, PAT] = \min_j\{D(TEXT[j..i], PAT)\}$, where $1 \leq j \leq i + 1$ and SD is the string distance metric. With the edit distance metric, the problem is to compute, for each i, the minimum number of edit operations between "pattern" string $PAT[1..|PAT|]$ and the "text" substring $TEXT[1..i]$, where any prefix can be removed from $TEXT[1..i]$. Intuitively the problem is the find the "occurrence" in *TEXT* that most closely matches *PAT*.

Given pattern tree P and data tree T, in order to allow P to match only a part of T, we must generalize the notion of a substring and the notion of removing a prefix. For us, a substring means a subtree and a prefix means a collection of subtrees.

Assume a numbering of tree T, we use $t[i]$ to represent the ith node of tree T, $T[i]$ to represent the subtree rooted at $t[i]$, and $T_f[i]$ to represent the forest obtained by deleting $t[i]$ from $T[i]$. Similarly, for pattern tree P we can define $p[i]$, $P[i]$, and $P_f[i]$.

We first define the operation of removing at a node [17].

Removing at node $t[i]$ means removing the subtree rooted at $t[i]$. Given tree T, we define $S(T)$ as a set of nodes of T satisfying

$$t[i], t[j] \in S(T) \text{ implies that neither is an ancestor of the other.}$$

$S(T)$ represents a set of nonoverlapping subtrees of T.

Define $R(T, S(T))$ to be the tree obtained by *removing at* all the nodes in $S(T)$ from tree T.

Now we can give the definition of approximate unordered tree matching. Given pattern tree P and data tree T, for each subtree $T[i]$, we want to compute

$$D_r(P, T[i]) = \min_S\{D(P, R(T[i], S(T[i])))\}.$$

The minimum here is over all possible subtree sets $S(T[i])$.

The intuitive meaning of the above definition is as follows. Given a pattern tree P and a data tree T, with approximate tree matching we want to find part of T, say T_p, that most closely matches P. We assume that T_p is connected, therefore T_p must be a tree. Since we consider rooted trees, T_p has a root. Let $t[i]$ be the root of T_p; it is clear that we can obtain T_p by removing some subtrees, say $S(T[i])$, from $T[i]$ since T_p is connected. Hence, $\min_i D_r(P, T[i])$ will give us the distance between P and T_p.

In the following we use n_i to represent the number of children of the node $p[i]$ and n_j to represent the number of children of the node $t[j]$.

LEMMA 9. *Let* $p[i_1], p[i_2], \ldots, p[i_{n_i}]$ *be the children of* $p[i]$ *and let* $t[j_1], t[j_2], \ldots, t[j_{n_j}]$ *be the children of* $t[j]$. *Then*

$$D_r(\theta, \theta) = 0,$$

$$D_r(\theta, T_f[j]) = 0, \qquad D_r(\theta, T[j]) = 0,$$

$$D_r(P_f[i], \theta) = \sum_{k=1}^{n_i} D_r(P[i_k], \theta), \qquad D_r(P[i], \theta) = D_r(P_f[i], \theta) + \gamma(p[i] \to \lambda).$$

PROOF. Trivial. □

LEMMA 10. *Let* $p[i_1], p[i_2], \ldots, p[i_{n_i}]$ *be the children of* $p[i]$ *and let* $t[j_1], t[j_2], \ldots, t[j_{n_j}]$ *be the children of* $t[j]$. *Then*

$$D_r(P[i], T[j]) = \min \begin{cases} D_r(P[i], \theta), \\ \gamma(\lambda \to t[j]) + \min\limits_{1 \le t \le n_j} D_r(P[i], T[j_t]), \\ D_r(P[i], \theta) + \min\limits_{1 \le s \le n_i} D_r(P[i_s], T[j]) - D_r(P[i_s], \theta), \\ D_r(P_f[i], T_f[j]) + \gamma(p[i] \to t[j]). \end{cases}$$

PROOF. We consider the best mapping between $P[i]$ and $T[j]$ after performing an optimal removal of subtrees of $T[j]$. If the whole tree $T[j]$ is removed, then the distance should be $D_r(P[i], \theta)$. Otherwise we have the same four cases as in Lemma 4. In case 1 $p[i]$ is in the best mapping but $t[j]$ is not. Suppose that $p[i]$ is mapped to $t[k]$ which is a node in $T[j_t]$. Then subtrees $T[j_1], \ldots, T[j_{t-1}], T[j_{t+1}], \ldots, T[j_{n_j}]$ should have been removed in the optimal removal of subtrees of $T[j]$. Therefore the distance is $\gamma(\lambda \to t[j]) + D_r(P[i], T[j_t])$. The other three cases are the same as in Lemma 4. □

In the following lemma we again use the concept of a maximum matching on a weighted bipartite graph. However, we have to modify the definition of the bipartite graph: Let $I = \{i_1, i_2, \ldots, i_{n_i}\}$, $J = \{j_1, j_2, \ldots, j_{n_j}\}$, and $B = \{b_1, b_2, \ldots, b_{n_i}\}$. We, again, use B to represent empty trees. Let $L \doteq I$ and $R = J \cup B$; we define a weighted bipartite graph $G_r(i, j) = (V, E)$ as follows:

(1) $V = L \cup R$.
(2) $E = L \times R$, where $\times$ stands for Cartesian product.
(3) $w(u, v) = D(T_1[u], T_2[v])$ if $u \in I$ and $v \in J$,
$w(u, v) = D(T_1[u], \theta)$ if $u \in I$ and $v \in B$.

We use $MM_r(i, j)$ to represent a maximum matching on graph $G_r(i, j)$.

LEMMA 11. *Let* $p[i_1], p[i_2], \ldots, p[i_{n_i}]$ *be the children of* $p[i]$ *and let* $t[j_1], t[j_2], \ldots, t[j_{n_j}]$ *be the children of* $t[j]$. *Then*

$$D_r(P_f[i], T_f[j]) = \min \begin{cases} \min\limits_{1 \le t \le n_j} D_r(P_f[i], T_f[j_t]) + \gamma(\lambda \to t[j_t]), \\ D_r(P_f[i], \theta) + \min\limits_{1 \le s \le n_i} D_r(P_f[i_s], T_f[j]) - D_r(P_f[i_s], \theta), \\ \min\limits_{MM_r(i,j)} \gamma(MM_r(i, j)). \end{cases}$$

PROOF. Again we consider the best mapping after performing the optimal removal. If $T_f[j]$ is removed, then the distance should be $D_r(P_f[i], \theta)$. However, this is handled by $\min_{MM_r(i,j)} \gamma(MM_r(i, j))$ since $\gamma(MM_r(i, j)) = D_r(P_f[i], \theta)$ when $MM_r(i, j) = \{(i_s, b_s) \mid 1 \le s \le n_i\}$.

Suppose that $T_f[j]$ is not removed, then we have four cases as in Lemma 5. Case 1 is similar to the case 1 in Lemma 5. The only difference is that we do not need to consider the costs of inserting subtrees of $T_f[j]$ which are not in the mapping since they should have been removed. Case 2 is the same as case 2 in Lemma 5. Case 3 is a special case for $\min_{MM_r(i,j)} \gamma(MM_r(i, j))$, where $MM_r(i, j)$ contains only one element from $I \times J$. Case 4 is similar to case 4 of Lemma 5. $G_r(i, j)$ and $MM_r(i, j))$ are defined to deal with this case where we do not need to consider those subtrees of $T_f[j]$ that are not in the best mapping. □

We again reduce the computation of $\min_{MM_r(i,j)} \gamma(MM_r(i, j))$ to the minimum cost flow problem.

Given $P[i]$ and $T[j]$, let $I = \{i_1, i_2, \ldots, i_{n_i}\}$ and $J = \{j_1, j_2, \ldots, j_{n_j}\}$, where i_k, $1 \le k \le n_i$, represents tree $P[i_k]$, and j_l, $1 \le l \le n_j$, represents tree $T[j_l]$.

We construct a graph $G_r = (V, E)$ as follows:

vertex set: $V = \{s, t, e, f\} \cup I \cup J$, where s is the source, t is the sink, and e represents an empty tree;
edge set: $[s, i_k]$, $[j_l, f]$, $[e, f]$, $[f, t]$ with cost zero, $[i_k, j_l]$ with cost $D_r(P[i_k], T[j_l])$, and $[i_k, e]$ with cost $D_r(P[i_k], \theta)$. All the edges have capacity one except $[e, f]$ and $[f, t]$, whose capacities are n_i.

G_r is a network with integer capacities, nonnegative costs, and the maximum flow $f^* = n_i$ (see Figure 2).

We can prove that $cost(G_r) = \min_{MM_r(i,j)} \gamma(MM_r(i, j))$. The proof is similar to the proof of Lemma 8.

We are now ready to give an algorithm for approximate unordered tree matching. The time complexity of the algorithm is $O(|P| \times |T| \times deg(P) \times \log_2(deg(P) + deg(T)))$ since f^* for G_r is n_i instead of $n_i + n_j$.

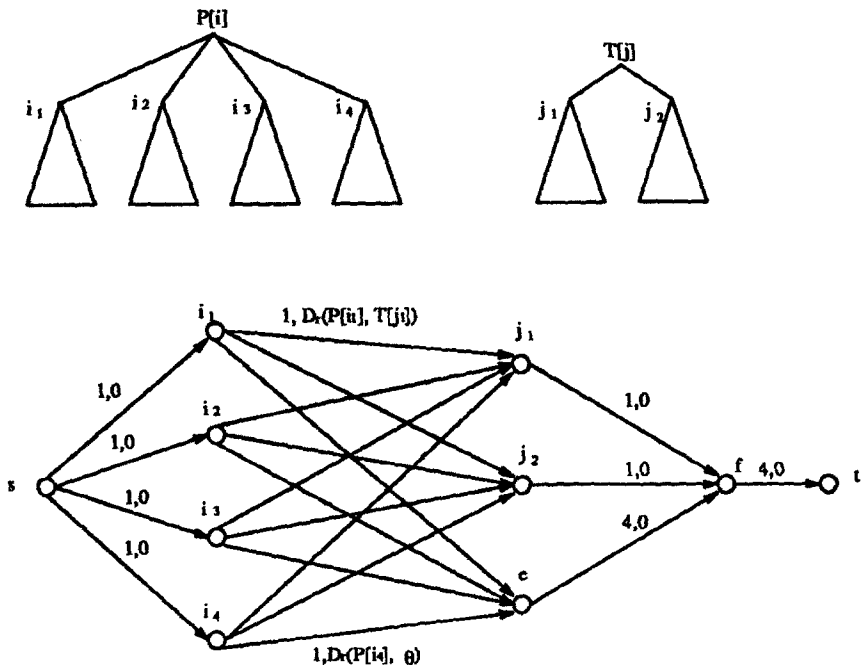

Fig. 2. Reduction to the minimum cost flow problem.

Input: P and T.
Output: $D_r(P, T[i])$, where $1 \le i \le |T|$.

$D_r(\theta, \theta) = 0$;
for $j = 1$ to $|T|$
  $D_r(\theta, T_f[j]) = 0$
  $D_r(\theta, T[j]) = 0$

for $i = 1$ to $|P|$

$$D_r(P_f[i], \theta) = \sum_{k=1}^{n_i} D_r(P[i_k], \theta)$$

$$D_r(P[i], \theta) = D_r(P_f[i], \theta) + \gamma(p[i] \to \lambda)$$

for $i = 1$ to $|P|$
  for $j = 1$ to $|T|$

$$D_r(P_f[i], T_f[j]) = \min \begin{cases} \min\limits_{1 \le t \le n_j} D_r(P_f[i], T_f[j_t]) + \gamma(\lambda \to t[j_t]), \\ D_r(P_f[i], \theta) + \min\limits_{1 \le s \le n_i} D_r(P_f[i_s], T_f[j]) - D_r(P_f[i_s], \theta), \\ \min\limits_{MM_r(i,j)} \gamma(MM_r(i, j)). \end{cases}$$

$$D_r(P[i], T[j]) = \min \begin{cases} D_r(P[i], \theta), \\ \gamma(\lambda \to t[j]) + \min\limits_{1 \le t \le n_j} D_r(P[i], T[j_t]), \\ D_r(P[i], \theta) + \min\limits_{1 \le s \le n_i} D_r(P[i_s], T[j]) - D_r(P[i_s], \theta,), \\ D_r(P_f[i], T_f[j]) + \gamma(p[i] \to t[j]). \end{cases}$$

7. Conclusion. Motivated by the NP-complete [3], [19] and MAX SNP-hard results [16], we have defined a constrained edit distance metric between unordered labeled trees. We present an algorithm for computing this distance metric based on a reduction to the minimum cost maximum flow problem. Our algorithm is generalizable with the same complexity to the approximate unordered-tree-matching problem.

The work presented is part of a project to develop a comprehensive tool for approximate tree pattern matching [14]. The proposed algorithms and their implementation are currently integrated into this tool.

Acknowledgments. The author would like to thank the anonymous referees for their many helpful suggestions.

References

[1] A. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy, Proof verification and hardness of approximation problems, *Proc. 33rd IEEE Symp. on the Foundation of Computer Science*, 1992, pp. 14–23.

[2] G. M. Landau and U. Vishkin, Fast parallel and serial approximate string matching, *J. Algorithms*, **10** (1989), 157–169.

[3] P. Kilpelainen and H. Mannila, The tree inclusion problem, *Proc. Internat. Joint Conf. on the Theory and Practice of Software Development (CAAP '91)*, 1991, Vol. 1, pp. 202–214.

[4] S. Masuyama, Y. Takahashi, T. Okuyama, and S. Sasaki, On the largest common subgraph problem, *Algorithms and Computing Theory*, RIMS, Kokyuroku (Kyoto University), 1990, pp. 195–201.

[5] P. H. Sellers, The theory and computation of evolutionary distances, *J. Algorithms*, **1** (1980), 359–373.

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

[7] F. Y. Shih, Object representation and recognition using mathematical morphology model, *J. System Integration*, **1** (1991), 235–256.

[8] F. Y. Shih and O. R. Mitchell, Threshold decomposition of grayscale morphology into binary morphology, *IEEE Trans. Pattern Anal. Mach. Intell.*, **11** (1989), 31–42.

[9] K. C. Tai, The tree-to-tree correction problem, *J. Assoc. Comput. Mach.*, **26** (1979), 422–433.

[10] Y. Takahashi, Y. Satoh, H. Suzuki, and S. Sasaki, Recognition of largest common structural fragment among a variety of chemical structures, *Anal. Sci.*, **3** (1987), 23–28.

[11] E. Tanaka and K. Tanaka, The tree-to-tree editing problem, *Internat. J. Pattern Recog. Artificial Intell.*, **2**(2) (1988), 221–240.

[12] R. E. Tarjan, *Data Structures and Network Algorithms*, CBMS–NSF Regional Conference Series in Applied Mathematics, CBMS, Washington, DC, 1983.

[13] E. Ukkonen, Finding approximate patterns in strings, *J. Algorithms*, **6** (1985), 132–137.

[14] J. T. L. Wang, Kaizhong Zhang, Karpjoo Jeong, and D. Shasha, ATBE: a system for approximate tree matching, *IEEE Trans. Knowledge Data Engrg*, **6**(4) (1994), 559–571.

[15] Kaizhong Zhang, Algorithms for the Constrained Editing Distance Between Ordered Labeled Trees and Related Problems, Technical Report No. 361, Department of Computer Science, University of Western Ontario, 1993.

[16] Kaizhong Zhang and Tao Jiang, Some MAX SNP-hard results concerning unordered labeled trees, *Inform. Process. Lett.*, **49** (1994), 249–254.

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

[18] Kaizhong Zhang, D. Shasha, and J. Wang, Approximate tree matching in the presence of variable length don't cares, *J. Algorithms*, **16** (1994), 33–66.

[19] Kaizhong Zhang, R. Statman and D. Shasha, On the editing distance between unordered labeled trees, *Inform. Process. Lett.*, **42** (1992), 133–139.