The Tree-to-Tree Correction Problem

KUO-CHUNG TAI

North Carolina State University, Raleigh, North Carolina

ABSTRACT The tree-to-tree correction problem is to determine, for two labeled ordered trees T and T', the distance from T to T' as measured by the minimum cost sequence of *edit* operations needed to transform T into T' The edit operations investigated allow changing one node of a tree into another node, deleting one node from a tree, or inserting a node into a tree An algorithm is presented which solves this problem in time $O(V*V'*L^2*L'^2)$, where V and V' are the numbers of nodes respectively of T and T', and L and L' are the maximum depths respectively of T and T' Possible applications are to the problems of measuring the similarity between trees, automatic error recovery and correction for programming languages, and determining the largest common substructure of two trees

KEY WORDS AND PHRASES tree correction, tree modification, tree similarity

CR CATEGORIES 3 79, 4.12, 4 22, 5 23, 5 25

1. *Introduction*

The string-to-string correction problem, which is to determine the *distance* between two strings as measured by the minimum cost sequence of *edit* operations needed to transform one string into the other, was investigated in [7, 8, 10, 12, 13, 14]. In [13] Wagner and Fischer considered the following three edit operations: changing a character into another character, deleting a character, and inserting a character; and they presented an algorithm that computes the distance between two strings in time $O(m*n)$, where m and n are the lengths of the two given strings.

The various attempts to achieve high-dimensional generalizations of strings include trees, graphs, webs, and plex structures. Trees are considered the most important nonlinear structures arising in computer algorithms [6]. Tree automata, tree grammars, and their applications for syntactic pattern recognition have been studied (e.g. [2]).

In this paper we define the notion of distance between two labeled ordered trees and present an algorithm that computes the distance in time $O(V*V'*L^2*L'^2)$, where V and V' are the numbers of nodes of the two trees and L and L' are the maximum depths of the two trees. The set of allowable edit operations includes: (1) changing one node into another node (changing the label of the node); (2) deleting one node from a tree; and (3) inserting a node into a tree.

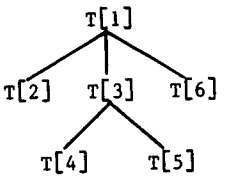
Since each string can be considered a tree of depth two (with a virtual root added), the string-to-string correction problem is just a special case of the tree-to-tree correction problem with $L = L' = 2$. Possible applications of the notion of distance between two trees are discussed at the end of this paper.

2. *Edit Operations on Trees*

In this paper all trees we discuss are rooted, ordered, and labeled. Let T be a tree. $|T|$

Author's address Department of Computer Science, North Carolina State University, P O Box 5972, Raleigh, NC 27607

denotes the number of nodes of T. $T[i]$ denotes the node of T whose position in the preorder for nodes of T is i. The preorder traversal on T is to first visit the root of T and then traverse the subtrees of T from left to right, each subtree traversed in preorder. The following diagram illustrates how nodes of a tree T are denoted:



LEMMA 2.1. *Assume that $T[i_1]$ and $T[i_2]$ are nodes of T with $T[i_1]$ being an ancestor of $T[i_2]$. Then*

(1) $i_1 < i_2$;

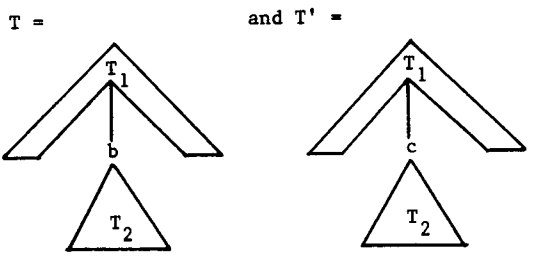
(2) *For any i such that $i_1 < i \leq i_2$, $T[i]$ is a descendant of $T[i_1]$;*

(3) *For the father $T[i_3]$ of $T[i_2]$, $T[i_3]$ is $T[i_2 - 1]$ or an ancestor of $T[i_2 - 1]$, and $T[i_3]$ is on the path from $T[i_1]$ to $T[i_2 - 1]$.*

PROOF. The proof follows from the definition of preorder traversal. Q.E.D.

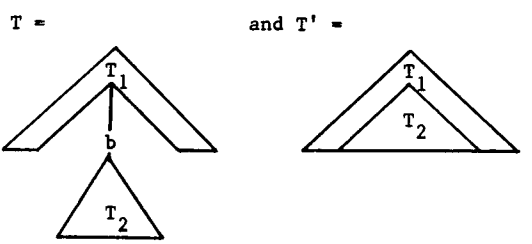
Let Λ denote the null node. An edit operation is written $b \rightarrow c$, where each of b and c is either a node or Λ. $b \rightarrow c$ is a *change* operation if $b \neq \Lambda$ and $c \neq \Lambda$, a *delete* operation if $b \neq \Lambda = c$, and an *insert* operation if $b = \Lambda \neq c$. Let T' be the tree that results from the application of an edit operation $b \rightarrow c$ to tree T; this is written $T \Rightarrow T'$ via $b \rightarrow c$. The relations between T and T' are described as follows:

(1) If $b \rightarrow c$ is a change operation, then node b is replaced by node c, i.e.,



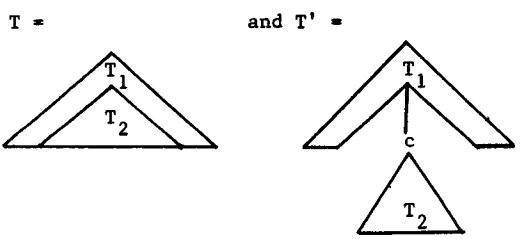
for some trees T_1 and T_2.

(2) If $b \rightarrow c$ is a delete operation, then node b is deleted, i.e.,



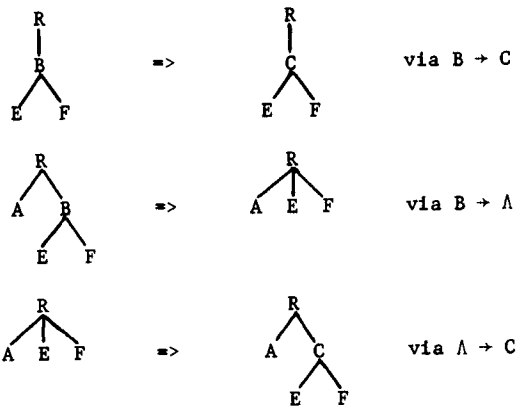
for some trees T_1 and T_2.

(3) If $b \rightarrow c$ is an insert operation, then node c is inserted, i.e.,



for some trees T_1 and T_2.

Each node of a tree except the root is associated with a unique edge connecting the node and its father. For the deletion or insertion of a node, both the node and its associated edge are included. Without loss of generality it may be assumed that the roots of all trees have the same unique label and that the root of each tree remains unchanged during editing. Examples are given below to illustrate the applications of the three edit operations.



Let S be a sequence $s_1, s_2, \ldots, s_m$ of edit operations. S transforms tree T to tree T' if there is a sequence of trees $T_0, T_1, \ldots, T_m$ such that $T = T_0$, $T' = T_m$ and $T_{i-1} => T_i$ via s_i for $1 \leq i \leq m$.

Let r be an arbitrary cost function which assigns to each edit operation $b \to c$ a nonnegative real number $r(b \to c)$. Extend r to a sequence of edit operations $S = s_1, s_2, \ldots, s_m$ by letting $r(S) = \sum_{i=1}^{m} r(s_i)$. Without loss of generality it may be assumed that $r(b \to b) = 0$ and that $r(b \to a) + r(a \to c) \geq r(b \to c)$.

The *distance* $d(T, T')$ from tree T to tree T' is defined to be the minimum cost of all sequences of edit operations which transform T into T', i.e.,

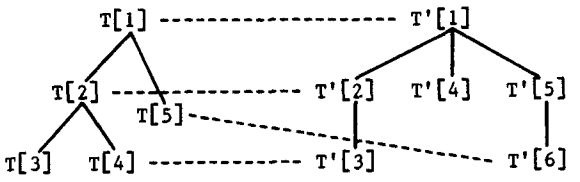
$$d(T, T') = \min\{r(S) \mid S \text{ is a sequence of edit operations which transforms } T \text{ into } T'\}.$$

The number of different sequences of edit operations which transform T into T' is infinite. Therefore, it is impossible to enumerate all valid sequences and find the minimum cost In the next section, structures called mappings are defined so that $d(T, T')$ can be computed in polynomial time.

3 *Mappings*

Intuitively, a *mapping* is a description of how a sequence of edit operations transforms T into T', ignoring the order in which edit operations are applied.

Consider the following diagram:

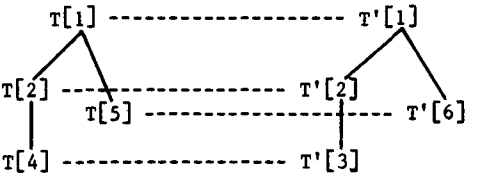


A dotted line from $T[i]$ to $T'[j]$ indicates that $T[i]$ should be changed to $T'[j]$ if $T[i] \neq T'[j]$, or that $T[i]$ remains unchanged, but becomes $T'[j]$, if $T[i] = T'[j]$. Nodes of T not touched by dotted lines are to be deleted and nodes of T' not touched are to be inserted. Such a diagram is called a mapping, which shows a way of transforming T to T'. Formally, we define a triple (M, T, T') to be a *mapping* from T to T', where M is any set of pairs of integers (i, j) satisfying:

(1) $1 \leq i \leq |T|$, $1 \leq j \leq |T'|$;
(2) For any two pairs (i_1, j_1) and (i_2, j_2) in M,
  (a) $i_1 = i_2$ iff $j_1 = j_2$;
  (b) $i_1 < i_2$ iff $j_1 < j_2$;
  (c) $T[i_1]$ is an ancestor (descendant) of $T[i_2]$ iff $T'[j_1]$ is an ancestor (descendant) of $T'[j_2]$.

Where there is no resulting confusion, we do not distinguish between the triple (M, T, T') and the set M.

Each pair (i, j) in M is interpreted to be a line segment joining $T[i]$ and $T'[j]$. Condition (2a) ensures that each node of both T and T' is touched by at most one line. Conditions (2b) and (2c) ensure that after untouched nodes of both T and T' are deleted, T and T' become similar (i.e., they have the same structure), and the one-to-one correspondence between nodes of T and T' which preserves the structure is indicated exactly by the lines of M. By deleting untouched nodes, the preceding diagram becomes:



Let M be a mapping from T to T' and let I and J be the sets of untouched nodes of T and T' respectively. We define the *cost* of M:

$$cost(M) = \sum_{(i,j)\in M} r(T[i] \to T'[j]) + \sum_{i\in I} r(T[i] \to \Lambda) + \sum_{j\in J} r(\Lambda \to T'[j]).$$

Thus, the cost of M is just the cost of the sequence of edit operations which consists of a change operation $T[i] \to T'[j]$ for each line (i, j) of M, a delete operation $T[i] \to \Lambda$ for each node $T[i]$ not touched by a line of M, and an insert operation $\Lambda \to T'[j]$ for each node $T'[j]$ not touched by a line of M.

In order to show that $d(T, T')$ can be determined by a minimum cost mapping from T to T', the following two lemmas need to be verified:

LEMMA 3.1. *Let M_1 be a mapping from T_1 to T_2 and let M_2 be a mapping from T_2 to T_3. Then:*

(1) *$M_1 \circ M_2 = \{(i, k) | (i, j) \in M_1$ and $(j, k) \in M_2$ for some $j\}$ is a mapping from T_1 to T_3;*
(2) *$cost(M_1 \circ M_2) \leq cost(M_1) + cost(M_2)$.*

PROOF.

(1) Let (i_1, k_1) and (i_2, k_2) be two lines of $M_1 \circ M_2$. Then there exist j_1 and j_2 such that (i_1, j_1), $(i_2, j_2) \in M_1$ and (j_1, k_1), $(j_2, k_2) \in M_2$. By the definition of a mapping:
  (a) $(i_1 = i_2$ iff $j_1 = j_2)$ and $(j_1 = j_2$ iff $k_1 = k_2)$;
  (b) $(i_1 < i_2$ iff $j_1 < j_2)$ and $(j_1 < j_2$ iff $k_1 < k_2)$;
  (c) $(T_1[i_1]$ is an ancestor (descendant) of $T_1[i_2]$ iff $T_2[j_1]$ is an ancestor (descendant) of $T_2[j_2])$, and $(T_2[j_1]$ is an ancestor (descendant) of $T_2[j_2]$ iff $T_3[k_1]$ is an ancestor (descendant) of $T_3[k_2])$.

Therefore, $M_1 \circ M_2$ is a mapping from T_1 to T_3.

(2) The proof that $cost(M_1 \circ M_2) \leq cost(M_1) + cost(M_2)$ follows closely the proof of Lemma 1 in [13] and is omitted. This proof relies on the assumption that $r(b \to c) \leq r(b \to a) + r(a \to c)$. Q.E.D.

LEMMA 3.2. *For any sequence S of edit operations which transforms T into T', there exists a mapping M from T to T' such that $cost(M) \leq r(S)$.*

PROOF. It can be shown by induction on m that if $S = s_1, s_2, \ldots, s_m$ is a sequence of edit operations and $T_0, T_1, \ldots, T_m$ is the sequence of trees from T_0 to T_m via S, then there

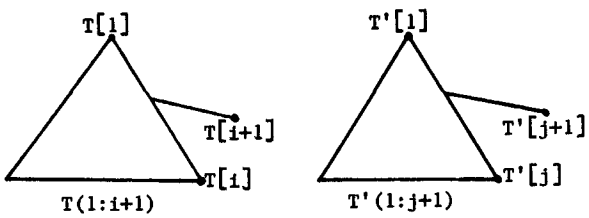
exists a mapping M from T_0 to T_m such that $cost(M) \leq r(S)$. This proof is similar to the proof of Theorem 1 in [13] and is omitted. Q.E.D.

Since $d(T, T') = \min\{cost(S) \mid S$ is a sequence of edit operations which transforms T into $T'\}$, we have:

THEOREM 3.1 $d(T, T') = min\{cost(M) \mid M$ *is a mapping from* T *to* $T'\}$.

Hence the search for a minimal cost sequence of edit operations has been reduced to a search for a minimal cost mapping.

Let $T(i_1: i_2)$ denote the portion of T which consists of nodes $T[i_1]$, $T[i_1 + 1]$, ... , $T[i_2]$, and their associated edges, where i_1 is an ancestor of i_2; and similarly define $T'[j_1: j_2]$. Let $D(i + 1, j + 1)$ be the distance from tree $T(1: i + 1)$ to tree $T'(1: j + 1)$, i.e., $D(i + 1, j + 1) = d(T(1: i + 1), T'(1: j + 1))$. Consider the following diagram:



Suppose M is a minimum cost mapping from $T(1: i + 1)$ to $T'(1: j + 1)$, i.e., $cost(M) = D(i + 1, j + 1)$. Then at least one of the following three cases must hold:

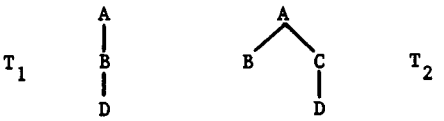
Case 1: $T[i + 1]$ is not touched by a line of M. Then $cost(M) = D(i, j + 1) + r(T[i + 1] \rightarrow \Lambda)$, corresponding to the cost of transforming $T(1: i)$ to $T'(1: j + 1)$ plus the cost of deleting $T[i + 1]$.

Case 2: $T'[j + 1]$ is not touched by a line of M. Then $cost(M) = D(i + 1, j) + r(\Lambda \rightarrow T'[j + 1])$, corresponding to the cost of transforming $T(1: i + 1)$ to $T'(1: j)$ plus the cost of inserting $T'[j + 1]$.

Case 3: $T[i + 1]$ and $T'[j + 1]$ are touched by lines $(i + 1, q)$ and $(p, j + 1)$ of M. By the definition of a mapping, $(i + 1 = p$ iff $q = j + 1)$ and $(i + 1 > p$ iff $q > j + 1)$. Since $i + 1 \geq p$ and $j + 1 \geq q$, $i + 1 = p$ and $j + 1 = q$. Thus, $T[i + 1]$ and $T'[j + 1]$ are both touched by the same line $(i + 1, j + 1)$. Note that if $M' = M - \{(i + 1, j + 1)\}$ is not a minimum cost mapping from $T(1: i)$ to $T'(1: j)$, then

$$cost(M) = cost(M') + r(T[i + 1] \rightarrow T'[j + 1]) > D(i, j) + r(T[i + 1] \rightarrow T'[j + 1]).$$

Thus, it may be true that $D(i + 1, j + 1) > D(i, j) + r(T[i + 1] \rightarrow T'[j + 1])$, for a minimum cost mapping from $T(1: i)$ to $T'(1: j)$ may not be extendable to a mapping by adding $(i + 1, j + 1)$. For example, consider the following trees T_1 and T_2:



Assume that the cost of each change, delete, or insert operation is 1 for all nodes. $\{(1, 1), (2, 2)\}$ is a minimum cost mapping from $T_1(1: 2)$ to $T_2(1: 3)$. However, $\{(1, 1), (2, 2), (3, 4)\}$ is not a mapping from $T_1(1: 3)$ to $T_2(1: 4)$ because $T_1[2]$ is an ancestor of $T_1[3]$ but $T_2[2]$ is not an ancestor of $T_2[4]$.

Define

$$MIN\_M(i + 1, j + 1) = \min\{cost(M) \mid M \text{ is a mapping from } T(1: i + 1) \text{ to } T'(1: j + 1) \text{ such that } (i + 1, j + 1) \in M\}.$$

From the above argument we have the following theorem:

THEOREM 3.2.

$$D(i+1, j+1) = \min\{D(i, j+1) + r(T[i+1] \to \Lambda),$$
$$D(i+1, j) + r(\Lambda \to T'[j+1]),$$
$$MIN\_M(i+1, j+1)\}$$

for all i, j, $1 \le i < |T|$, $1 \le j < |T'|$.

For the special case that $MIN\_M(i+1, j+1) = D(i, j) + r(T[i+1] \to T'[j+1])$ for all i, j, $1 \le i \le |T|$, and $1 \le j \le |T'|$, $D(|T|, |T'|)$ can be computed in time $O(|T| * |T'|)$ (see [13]).

The computation of $MIN\_M(i+1, j+1)$ is not trivial. In the next section we explore some properties of mappings and then show that $MIN\_M(i+1, j+1)$ can be computed in polynomial time. With the assumption that the root of any tree remains unchanged during editing, every mapping from $T(1: i+1)$ to $T'(1: j+1)$ contains (1, 1) and $D(1, 1) = 0$.

LEMMA 3.3.

$$D(1, 1) = 0,$$

$$D(i, 1) = \sum_{k=2}^{i} r(T[k] \to \Lambda), \quad 1 < i \le |T|, \quad \textit{and}$$

$$D(1, j) = \sum_{k=2}^{j} r(\Lambda \to T'[k]), \quad 1 < j \le |T'|.$$

4. *Computation of* $MIN\_M(i+1, j+1)$

We first show that a mapping from $T(1: i+1)$ to $T'(1: j+1)$ may be decomposed into submappings such that each submapping is a mapping from one portion of $T(1: i+1)$ to one portion of $T'(1: j+1)$. Consequently, for a minimum cost mapping, each of its submappings is also a minimum cost mapping.

LEMMA 4.1. *Assume that* $T[p]$ *and* $T'[q]$ *are ancestors of* $T[i+1]$ *and* $T'[j+1]$, *respectively, and that* M *is a mapping from* $T(1: i+1)$ *to* $T'(1: j+1)$ *such that* (p, q) *and* $(i+1, j+1)$ *are in* M. *Let* M_1 *and* M_2 *be the subsets of* M *defined by*

$$M_1 = \{(m, n) | (m, n) \in M,\ 1 \le m \le p \text{ and } 1 \le n \le q\}$$

and

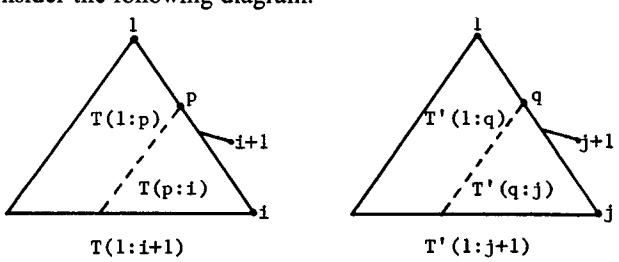
$$M_2 = \{(m, n) | (m, n) \in M,\ p \le m \le i \text{ and } q \le n \le j\}.$$

Then

(1) M_1 *is a mapping from* $T(1: p)$ *to* $T'(1: q)$;
M_2 *is a mapping from* $T(p: i)$ *to* $T'(q: j)$;
$M = M_1 \cup M_2 \cup \{(i+1, j+1)\}$; *and*
$cost(M) = cost(M_1) + cost(M_2) - r(T[p] \to T'[q]) + r(T[i+1] \to T'[j+1])$.

(2) $cost(M) = \min\{cost(M') | M'$ *is a mapping from* $T(1: i+1)$ *to* $T'(1: j+1)$ *such that* (p, q) *and* $(i+1, j+1)$ *are in* $M'\}$ *iff*
$cost(M_1) = \min\{cost(M_1') | M_1'$ *is a mapping from* $T(1: p)$ *to* $T'(1: q)$ *such that* $(p, q) \in M_1'\}$ *and*
$cost(M_2) = \min\{cost(M_2') | M_2'$ *is a mapping from* $T(p: i)$ *to* $T'(q: j)$ *such that* $(p, q) \in M_2'$ *and* $M_2' \cup \{(i+1, j+1)\}$ *is a mapping*$\}$.

PROOF. Consider the following diagram:



(1) For each (m, n) in M, $m \geq p$ if and only if $n \geq q$. Therefore, $M = M_1 \cup M_2 \cup \{(i+1, j+1)\}$ and M_1 and M_2 have exactly one common line, (p, q).

(2) *Only if*: Assume that

$$cost(M_1) > \min\{cost(M_1') \mid M_1' \text{ is a mapping from } T(1{:}\,p) \text{ to } T'(1{:}\,q) \text{ with } (p, q) \text{ in } M_1'\} = cost(M_1''),$$

where M_1'' is a mapping from $T(1{:}\,p)$ to $T'(1{:}\,q)$ with (p, q) in M_1''. Then $M'' = M_1'' \cup M_2 \cup \{(i+1, j+1)\}$ is a mapping from $T(1{:}\,i+1)$ to $T'(1{:}\,j+1)$ with (p, q) and $(i+1, j+1)$ in M'', and $cost(M'') < cost(M)$. Thus

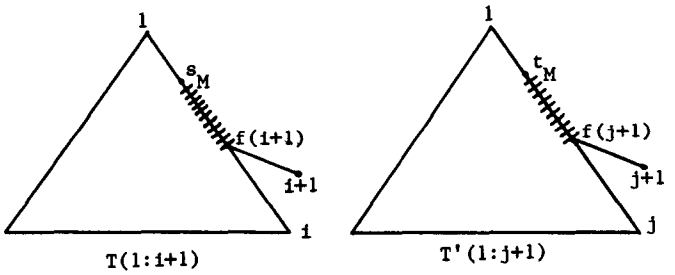
$$cost(M) > \min\{cost(M') \mid M' \text{ is a mapping from } T(1{:}\,i+1) \text{ to } T'(1{:}\,j+1) \text{ with } (p, q) \text{ and } (i+1, j+1) \text{ in } M'\}.$$

The same conclusion holds if

$$cost(M_2) > \min\{cost(M_2') \mid M_2' \text{ is a mapping from } T(p{:}\,i) \text{ to } T'(q{:}\,j) \text{ with } (p, q) \text{ in } M_2' \text{ and } M_2' \cup \{(i+1, j+1)\} \text{ is a mapping}\}.$$

If: The proof is similar to the "only if" proof. Q.E.D.

Assume that M is a mapping from $T(1{:}\,i+1)$ to $T'(1{:}\,j+1)$ with $(i+1, j+1)$ in M. Let $T[s_M]$ and $T'[t_M]$ be the latest ancestors of $T[i+1]$ and $T'[j+1]$, respectively, touched by lines of M. (It has been assumed that every mapping from $T(1{:}\,i+1)$ to $T'(1{:}\,j+1)$ contains the line (1, 1), and therefore $T[s_M]$ and $T'[t_M]$ must exist.) Since $(i+1, j+1)$ is in M, (s_M, t_M) is in M. The following diagram illustrates the meaning of nodes $T[s_M]$ and $T'[t_M]$:



In the above diagram, $f(x)$ denotes the father of node x. By Lemma 2.1, $T[f(i+1)]$ ($T'[f(j+1)]$) is on the path from $T[s_M]$ ($T'[t_M]$) to $T[i]$ ($T'[j]$). Slashes crossing the line from s_M to $f(i+1)$ (not including s_M) indicate that any descendant of $T[s_M]$ on the path from $T[s_M]$ to $T[f(i+1)]$ is not touched by any line of M. Slashes in $T'(1{:}\,j+1)$ are defined similarly.

LEMMA 4.2.

$$MIN\_M(i+1, j+1) = r(T[i+1] \to T'[j+1]) + \min_{s,t}\{\min\{cost(M_1) \mid M_1 \text{ is a mapping from } T(1{:}\,s) \text{ to } T'(1{:}\,t) \text{ with } (s, t) \text{ in } M_1\} + \min\{cost(M_2) \mid M_2 \text{ is a mapping from } T(s{:}\,i) \text{ to } T'(t{:}\,j) \text{ such that } (s, t) \text{ is in } M \text{ and any descendant of } T[s] \ (T'[t]) \text{ on the path from } T[s] \ (T'[t]) \text{ to } T[f(i+1)] \ (T'[f(j+1)]) \text{ is not touched by any line of } M_2\} - r(T[s] \to T'[t])\}$$

where $T[s]$ and $T'[t]$ are ancestors of $T[i+1]$ and $T'[j+1]$, respectively.

PROOF. Let R denote the right side of the above formula and let L denote $MIN\_M(i+1, j+1)$, which is $\min\{cost(M) \mid M$ is a mapping from $T(1{:}\,i+1)$ to $T'(1{:}\,j+1)$ with $(i+1, j+1)$ in $M\}$.

Assume that M is a mapping from $T(1{:}\,i+1)$ to $T'(1{:}\,j+1)$ with $(i+1, j+1)$ in M. Then M contains (s_M, t_M), where $T[s_M]$ and $T'[t_M]$ are the latest ancestors of $T[i+1]$ and $T'[j+1]$, respectively, touched by lines of M. By Lemma 4.1, M can be decomposed into submappings M_1 and M_2 such that M_1 is a mapping from $T(1{:}\,s_M)$ to $T'(1{:}\,t_M)$, M_2 is a

mapping from $T(s_M: i)$ to $T'(t_M: j)$, and $M = M_1 \cup M_2 \cup \{(i+1, j+1)\}$. It follows that $L \geq R$.

Assume that $T[s]$ and $T'[t]$ are ancestors of $T[i+1]$ and $T'[j+1]$, respectively. Let M_1 be a mapping from $T(1: s)$ to $T'(1: t)$ with (s, t) in M_1 and let M_2 be a mapping from $T(s: i)$ to $T'(t. j)$ such that (s, t) is in M_2 and any descendant of $T[s]$ $(T'[t])$ on the path from $T[s]$ $(T'[t])$ to $T[f(i+1)]$ $(T'[f(j+1)])$ is not touched by any line of M_2. Then $M = M_1 \cup M_2 \cup \{(i+1, j+1)\}$ is a mapping from $T(1: i+1)$ to $T'(j+1)$. Therefore $L \leq R$. From $L \geq R$ and $L \leq R$, it follows that $L = R$. Q.E.D.

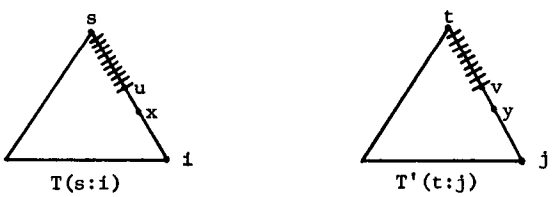
Assume that $s \leq u \leq i$, $t \leq v \leq j$, $T[u]$ is on the path from $T[s]$ to $T[i]$, and $T'[v]$ is on the path from $T'[t]$ to $T'[j]$. Define $E[s: u: i, t: v: j]$ to be $\min\{cost(M) | M$ is a mapping from $T(s: i)$ to $T'(t: j)$ such that (s, t) is in M and no descendant of $T[s]$ $(T'[t])$ on the path from $T[s]$ $(T'[t])$ to $T[u]$ $(T'[v])$ is touched by a line of $M\}$. Then we have the following theorem from Lemma 4.2:

THEOREM 4.1.

$$MIN\_M(i+1, j+1) = r(T[i+1] \to T'[j+1]) + \min_{s,t}\{MIN\_M(s, t) + E[s: f(i+1): i, t: f(j+1): j] - r(T[s] \to T'[t])\}$$

where $T[s]$ and $T'[t]$ are ancestors of $T[i+1]$ and $T'[j+1]$, respectively.

Now the remaining problem is how to compute $E[s: u: i, t: v: j]$, where $s \leq u \leq i$, $t \leq v \leq j$, and $T[u]$ $(T'[v])$ is on the path from $T[s]$ $(T'[t])$ to $T[i]$ $(T'[j])$. First, assume that $s \leq u < i$ and $t \leq v < j$. Then $T[u]$ $(T'[v])$ has a son on the path from $T[u]$ $(T'[v])$ to $T[i]$ $(T'[j])$. Consider the following diagram:



where $T[x]$ $(T'[y])$ is the son of $T[u]$ $(T'[v])$ on the path from $T[u]$ $(T'[v])$ to $T[i]$ $(T'[j])$.

LEMMA 4.3. *Assume that $s \leq u < i$ and $t \leq v < j$. Then*

$$E[s: u: i, t: v: j] = \min\{E[s: x: i, t: v: j], E[s: u: i, t: y: j], E[s: u: x-1, t: v: y-1] + E[x: x: i, y: y: j]\}.$$

PROOF. Let M be a mapping from $T(s: i)$ to $T'(t: j)$ such that $cost(M) = E[s: u: i, t: v: j]$, (s, t) is in M, and no descendant of $T[s]$ $(T'[t])$ on the path from $T[s]$ $(T'[t])$ to $T[u]$ $(T'[v])$ is touched by a line of M. Then at least one of the following three cases must hold:

Case 1: $T[x]$ is not touched by a line of M. Then $E[s: u: i, t: v: j] = E[s: x: i, t: v: j]$.

Case 2: $T'[y]$ is not touched by a line of M. Then $E[s: u: i, t: v: j] = E[s: u: i, t: y: j]$.

Case 3: $T[x]$ and $T'[y]$ are touched by lines (x, q) and (p, y) of M. Assume that $p > x$. Since $T[i]$ is a descendant of $T[x]$ and $i \geq p > x$, by Lemma 2.1 $T[p]$ is a descendant of $T[x]$. By the definition of a mapping, $y > q$ and $T'[y]$ is a descendant of $T'[q]$. Thus, $T'[q]$ is a descendant of $T'[t]$ on the path from $T'[t]$ to $T'[v]$. However, this contradicts the assumption that no descendant of $T'[t]$ on the path from $T'[t]$ to $T'[v]$ is touched by a line of M. Also, $p < x$ will cause a similar contradiction. Therefore, $p = x$ and $q = y$. Let M_1 and M_2 be defined by

$$M_1 = \{(m, n) | (m, n) \text{ is in } M,\ m < x \text{ and } n < y\},$$

and

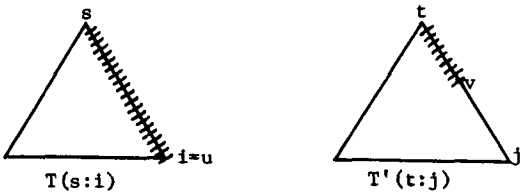
$$M_2 = \{(m, n) | (m, n) \text{ is in } M,\ m \geq x \text{ and } n \geq y\}.$$

Then M_1 is a mapping from $T(s: x-1)$ to $T'(t: y-1)$, M_2 is a mapping from $T(x: i)$ to $T'(y: j)$, and $cost(M) = cost(M_1) + cost(M_2)$. Since $cost(M) = E[s: u: i, t: v: j]$, it follows that $cost(M_1) = E[s: u: x-1, t: v: y-1]$ and $cost(M_2) = E[x: x: i, y: y: j]$. (Since $T[u]$ is the father of $T[x]$, by Lemma 2.1 $T[u]$ is on the path from $T[s]$ to $T[x-1]$ and likewise for $T'[v]$.) Therefore,

$$E[s: u: i, t: v: j] = E[s: u: x-1, t: v\ y-1] + E[x: x: i, y: y: j]. \qquad \text{Q.E.D.}$$

To compute $E[s: u: i, t: v: j]$, we now consider the cases in which one or both of $T[x]$ and $T[y]$ do not exist, i.e., $u = i$ or $v = j$. Let M be a mapping from $T(s: i)$ to $T'(t: j)$ such that $cost(M) = E[s: u: i, t: v: j]$, (s, t) is in M, and no descendant of $T[s]$ ($T'[t]$) on the path from $T[s]$ ($T'[t]$) to $T[u]$ ($T'[v]$) is touched by a line of M.

Case 1: $u = i$ and $v < j$. There are two subcases to be considered:



(a) $s = u = i$. Then $T(s: i)$ contains exactly one node, $T[s]$, and no descendant of $T'[t]$ is touched by a line of M. By Lemma 2.1, $T'[f(j)]$ is on the path from $T'[t]$ to $T'[j-1]$. Therefore,

$$E[s: u: i, t: v: j] = E[s: u: i, t: f(j): j-1] + r(\Lambda \rightarrow T'[j]).$$

(b) $s < u = i$. By Lemma 2.1, $T[f(i)]$ is on the path from $T[s]$ to $T[i-1]$. Since no descendant of $T[s]$ on the path from $T[s]$ to $T[f(i)]$ is touched by a line of M, it follows that

$$E[s: u: i, t: v: j] = E[s: f(i): i-1, t: v: j] + r(T[i] \rightarrow \Lambda)$$

Case 2: $u < i$ and $v = j$. This case is similar to case 1.

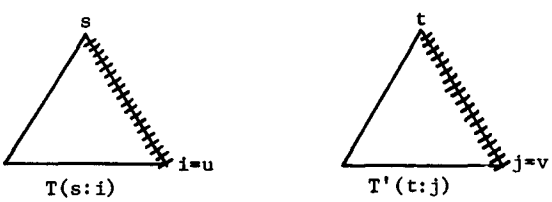
(a) $t = v = j$. Then

$$E[s: u: i, t: v: j] = E[s: f(i): i-1, t: v: j] + r(T[i] \rightarrow \Lambda)$$

(b) $t < v = j$. Then

$$E[s: u: i, t: v: j] = E[s: u: i, t: f(j): j-1] + r(\Lambda \rightarrow T'[j]).$$

Case 3. There are four subcases to be considered:



(a) $s = u = i$ and $t = v = j$. Then

$$E[s: u: i, t: v: j] = r(T[i] \rightarrow T'[j]).$$

(b) $s = u = i$ and $t < v = j$. Then

$$E[s: u: i, t: v: j] = E[s: u: i, t: f(j): j-1] + r(\Lambda \rightarrow T'[j]).$$

(c) $s < u = i$ and $t = v = j$. Then

$$E[s: u: i, t: v: j] = E[s: f(i): i-1, t: v: j] + r(T[i] \rightarrow \Lambda)$$

(d) $s < u = i$ and $t < v = j$. Then neither $T[i]$ nor $T'[j]$ is touched by a line of M. Therefore,

$$E[s\colon u\colon i, t\colon v\colon j] = E[s\colon f(i)\colon i-1, t\colon v\colon j] + r(T[i] \to \Lambda), \quad \text{or}$$
$$E[s\colon u\colon i, t\colon f(j)\colon j-1] + r(\Lambda \to T'[j]), \quad \text{or}$$
$$E[s\colon f(i)\colon i-1, t\colon f(j)\colon j-1] + r(T[i] \to \Lambda) + r(\Lambda \to T'[j]).$$

Although eight subcases are considered in Cases 1, 2, and 3, only four different formulas are used.

5. *An Algorithm for the Tree-to-Tree Correction Problem*

Based on the results shown in Sections 3 and 4, an algorithm is presented which computes the distance from tree T to tree T' in polynomial time. This algorithm consists of the following three steps:

(1) Compute $E[s\colon u\colon i, t\colon v\colon j]$ for all s, u, i, t, v, j, where

$$1 \le i \le |T|, \quad 1 \le j \le |T'|,$$

$T[u]$ ($T'[v]$) is on the path from $T[1]$ ($T'[1]$) to $T[i]$ ($T'[j]$),

$T[s]$ ($T'[t]$) is on the path from $T[1]$ ($T'[1]$) to $T[u]$ ($T'[v]$);

(2) Compute $MIN\_M(i, j)$ for all i, j, where $1 \le i \le |T|$ and $1 \le j \le |T'|$;

(3) Compute $D(i, j)$ for all i, j, where $1 \le i \le |T|$ and $1 \le j \le |T'|$.

Define $f^n(x) = f(f^{n-1}(x))$ for $n \ge 1$ and $x > 1$, where $f(x)$ is the father of node x, and $f^0(x) = x$ The following is an algorithm for step (1):

```
for i = 1, 2, , |T| do
for j = 1, 2, , |T'| do
for u = i, f(i), f^2(i), . , 1 do
for s = u, f(u), f^2(u), , 1 do
for v = j, f(j), f^2(j), , 1 do
for t = v, f(v), f^2(v), , 1 do
if s = u = i ∧ t = v = j then E[s u i, t v j] = r(T[i] → T'[j])
else if s = u = i ∨ t < v = j then E[s u i, t v j] = E[s: u i, t f(j) j − 1] + r(Λ → T'[j])
else if s < u = i ∨ t = v = j then E[s. u i, t v j] = E[s f(i) i − 1, t v j] + r(T[i] → Λ)
else E[s u i, t v. j] = min{E[s x i, t v. j], E[s u i, t y j], E[s u x − 1, t v y − 1] + E[x x i, y. y. j]}
```

($T[x]$ is the son of $T[u]$ on the path from $T[u]$ to $T[i]$, and $T'[y]$ is the son of $T'[v]$ on the path from $T'[v]$ to $T'[j]$.)

The following is an algorithm for step (2):

```
MIN_M(1, 1) = 0,
for i = 2, 3, , |T| do
  for j = 2, 3, , |T'| do
  begin
    MIN_M(i, j) ← INFINITE,
    for s = f(i), f^2(i), , 1 do
      for t = f(j), f^2(j), , 1 do
        begin
        temp ← MIN_M(s, t) + E[s f(i) i − 1, t f(j) j − 1] − r(T[s] → T'[t]),
        MIN_M(i, j) ← min(temp, MIN_M(i, j))
        end,
    MIN_M(i, j) ← MIN_M(i, j) + r(T[i] → T'[j])
  end,
```

Finally, an algorithm for step (3) is given below:

```
D(1, 1) ← 0,
D(i, 1) ← D(i − 1, 1) + r(T[i] → Λ)  for  i = 2, 3, . , |T|,
D(1, j) ← D(1, j − 1) + r(Λ → T'[j])  for  j = 2, 3, , |T'|,
for i = 2, 3, , |T| do
  for j = 2, 3, , |T'| do
    D(i, j) ← min(D(i, j − 1) + r(Λ → T'[j]), D(i − 1, j) + r(T[i] → Λ), MIN_M(i, j));
```

Now the tree-to-tree correction problem has been completely solved. The following theorem shows the time complexity of the proposed algorithm for computing the distance from one tree to another.

THEOREM 5.1 *Given two trees T and T', the proposed algorithm computes the distance from T to T' in time $O(V * V' * L^2 * L'^2)$, where V and V' are the numbers of nodes respectively of T and T', and L and L' are the maximum depths respectively of T and T'*

PROOF. Step (1) computes $E[s: u: i, t: v: j]$ for all s, u, i, t, v, j, where

$$1 \leq i \leq V,\ 1 \leq j \leq V',$$

$T[u]$ $(T'[v])$ is on the path from $T[1]$ $(T'[1])$ to $T[i]$ $(T'[j])$,

$T[s]$ $(T'[t])$ is on the path from $T[1]$ $(T'[1])$ to $T[u]$ $(T'[v])$.

Thus, step (1) takes $O(V * V' * L^2 * L'^2)$ time.

Step (2) computes $MIN\_M(i, j)$ for all i, j, where $1 \leq i \leq V$ and $1 \leq j \leq V'$. Thus, step (2) takes $O(V * V' * L * L')$ time.

Step (3) computes $D(i, j)$ for all i, j, where $1 \leq i \leq V$ and $1 \leq j \leq V'$. Thus, step (3) takes $O(V * V')$ time.

From steps (1), (2), and (3), the distance $D(V, V')$ from T to T' can be computed in time $O(V * V' * L^2 * L'^2)$. Q.E.D.

6. *Conclusion*

The notion of *distance* between two trees can be applied to measuring the *similarity* between two trees. Since trees have been used for different applications, the similarity between trees can have different interpretations. One possible application is to the problem of syntactic error recovery and correction for programming languages. In [11] it was suggested that for the selection of an error recovery or correction, the similarity between a corrected string and its replacement should be based on the two strings as well as their associated parse trees.

The *longest common subsequence* problem, which is a special case of the string-to-string correction problem, has received much attention [1, 3, 4, 5, 13]. The notion of longest common subsequence between two strings can be extended for trees as well. Tree T'' is a *substructure* of tree T if there exists a mapping M from T'' to T such that every node of T'' is touched by a line of M and for every $(p, q) \in M$, $T[p] = T'[q]$ Tree T'' is a *common substructure* of trees T and T' if T'' is a substructure of both T and T'. Tree T'' is a *largest common substructure* of T and T' if there is no common substructure of T and T' that has more nodes than T''. With constant cost W_C, W_D, W_I for changing, deleting, and inserting any node, and with $W_C = W_I + W_D$, the tree resulting from T by deleting nodes of T that are changed or deleted during a minimum cost transformation from T to T' is a largest common substructure of T and T'.

In [7, 12] the string-to-string correction problem was extended by allowing the operation of interchanging two adjacent characters. Wagner [12] showed that under certain restrictions the extended string-to-string correction problem can be solved in deterministic polynomial time, but the general problem is NP-complete. How to extend the tree-to-tree correction problem by allowing the operation of interchanging two adjacent nodes is currently being investigated.

The tree-to-tree correction problem may also be extendable by modifying the definitions of change, delete, and insert operations. One example is to add the restriction that the delete and insert operations can only be applied to the leaves of trees. For this restricted problem, one more condition should be added to the definition of a mapping M:

$$\text{For every } (i, j) \in M, \quad \text{if} \quad i \neq 1 \neq j, \quad \text{then} \quad (f(i), f(j)) \in M.$$

This condition implies that if $T[i]$ and $T'[j]$ are touched by the same line, then they have the same level number. The algorithm presented in Section 5 can be simplified to solve the

restricted problem, but the simplification process is not trivial. In [9] Selkow proposed a simple algorithm which solves this restricted problem in time $O(|T| * |T'|)$

ACKNOWLEDGMENTS. I would like to thank the referees for their helpful suggestions for improving the readability of this paper. The notion of $E[s: u: i, t: v: j]$ was motivated by a suggestion given by one referee. Using this notation, the computation of MIN_$M(i, j)$ has been substantially simplified.

REFERENCES

1 AHO, A V, HIRSCHBERG, D S, AND ULLMAN, J D Bounds on the complexity of the longest common subsequence problem *J ACM 23*, 1 (Jan 1976), 1–12
2 FU, K S, AND BHARGAVA, B K Tree systems for syntactic pattern recognition *IEEE Trans Comptrs C-22*, 12 (Dec 1973), 1087–1099
3 HIRSCHBERG, D S A linear space algorithm for computing maximal common subsequences *Comm ACM 18*, 6 (June 1975), 341–343
4 HIRSCHBERG, D S Algorithms for the longest common subsequence problem *J ACM 24*, 4 (Oct 1977), 664–675
5 HUNT, J W, AND SZYMANSKI, T G A fast algorithm for computing longest common subsequences *Comm ACM 20*, 5 (May 1977), 350–353
6 KNUTH, D E *The Art of Computer Programming, Vol 1 Fundamental Algorithms* Addison-Wesley, Reading, Mass, sec ed, 1973
7 LOWRANCE, R, AND WAGNER, R A An extension of the string-to-string correction problem *J ACM 22*, 2 (April 1975), 177–183
8 SANKOFF, D Matching sequences under deletion/insertion constraints *Proc Nat Acad Sci USA 69*, 1 (Jan 1974), 4–6
9 SELKOW, S M The tree-to-tree editing problem *Inform Processing Letters 6*, 6 (Dec 1977), 184–186
10 SELLERS, P H An algorithm for the distance between two finite sequences *J Combin Theory, Ser A*, 16 (1974), 253–258
11 TAI, K C Syntactic error correction in programming languages Ph D Th, Dept Comptr. Sci, Cornell U, Ithaca, N Y, 1977
12 WAGNER, R A On the complexity of the extended string-to-string correction problem Proc Seventh Annual ACM Symp on Theory of Compting, Albuquerque, New Mex, 1975, pp 218–223
13 WAGNER, R A, AND FISCHER, M J The string-to-string correction problem. *J ACM 21*, 1 (Jan 1974), 168–173
14 WONG, C K, AND CHANDRA, A K Bounds for the string editing problem *J. ACM 23*, 1 (Jan 1976), 13–16

RECEIVED MARCH 1977, REVISED JANUARY 1979