THE TREE-TO-TREE EDITING PROBLEM

Stanley M. SELKOW
Department of Computer Science, University of Tennessee, Knoxville, TN, USA

Received 30 March 1977, revised version received 27 June 1977

1. Introduction

Many processes, including evolution, derivation of a sentence in a grammar, hierarchical clustering and game playing, may be represented as a labeled ordered tree. It is often desirable to compare two trees of the minimum number of operations required to convert one to the other. We present here an algorithm to compute such a minimum sequence of operations.

A special case of this problem involves the comparison of two trees of depth two (each tree has a root and an ordered sequence of leaves which are the children of the root) in order to derive a minimum cost sequence of edit operations to transform one sequence of leaves to the other. Sankoff [2] and Wagner and Fisher [3] presented an algorithm to compute a minimum cost sequence of edit operations in $O(mn)$ operations, where the trees have $m$ and $n$ leaves. Wong and Chandra [4] and Aho, Hirschberg and Ullman [1] have proved that for a wide class of computation models, the Sankoff algorithm is optimal.

We will show that a straightforward generalization of the Sankoff algorithm will provide a solution to the tree-to-tree editing problem. Since the time required by our algorithm is of the same order of magnitude as the time required by the Sankoff algorithm, it follows that our algorithm must be optimal over a wide class of computation models.

2. Edit distance

A (labeled ordered) tree is a finite nonempty set of vertices $T$ with a labeling function $\lambda$ such that

1. $T$ has distinguished vertex called the root of the tree.
2. the remaining vertices (excluding the root) are partitioned into $m \geq 0$ disjoint sets $T_1, ..., T_m$ and each of these sets is a tree (they are called the subtrees of $T$),
3. associated with each vertex $v \in T$ is a label $\lambda(v)$. We let $\lambda(T)$ denote the label of the root of $T$. For $0 \leq i \leq m$, let $T(i)$ denote the tree obtained from $T$ by removing subtrees $T_{i+1}, ..., T_m$. If $A$ is a tree with subtrees $A_1, ..., A_m$ and $B$ is a tree with subtrees $B_1, ..., B_m$, then $A$ and $B$ are equal, $A = B$, if $\lambda(A) = \lambda(B)$ and $m = n$ and $A_i = B_i$ for $1 \leq i \leq m$. Note that $A = A(m)$.

Given a tree $T$ with $\lambda(T) = s_j$ and subtrees $T_1, ..., T_m$, $1$-label change operation $L(s_i, s_k)$ applied to $T$ yields the tree $T^*$ with $\lambda(T^*) = s_k$ and subtrees $T_1, ..., T_m$.
2. for $0 \leq i \leq m$ and tree $A$, an insert operation $I(A)$ applied to $T$ at $i$ yields the tree $T^*$ with $\lambda(T^*) = s_j$ and subtrees $T_1, ..., T_i, A, T_{i+1}, ..., T_m$.
3. for $1 \leq i \leq m$, a delete operation $D(T_i)$ applied to $T$ at $i$ yields the tree $T^*$ with $\lambda(T^*) = s_j$ and subtrees $T_1, ..., T_{i-1}, T_{i+1}, ..., T_m$.

An edit operation is any of the above three operations.

We associate a nonnegative cost with each edit operation in the following manner. Associated with each pair of labels $(s_i, s_j)$ is a cost $c_L(s_i, s_j)$ of applying the operation $L(s_i, s_j)$. For each label $s_i$, we let $c_F(s_i)$ and $c_D(s_i)$ denote the costs of applying $I(T)$ and $D(T)$ respectively, where $T$ is a tree with one vertex and $\lambda(T) = s_i$. For an arbitrary tree $T$, we let...
For any three labels $s_i$, $s_j$, and $s_k$, we assume $c_L(s_i, s_j) = 0$, and

$$c_L(s_i, s_k) \leq c_L(s_i, s_j) + c_L(s_k, s_j).$$

Given any trees $A$ and $B$ and the set of sequences of edit operations which when applied to $A$ yield a tree equal to $B$, we let $\delta(A, B)$ denote the minimum of the sums of the costs of each sequence. If tree $A$ has subtrees $A_1, \ldots, A_m$ and tree $B$ has subtrees $B_1, \ldots, B_n$, then

$$\delta(A, B) \leq c_L(\lambda(A), \lambda(B)) + \sum_{i=1}^{m} c_D(A_i) + \sum_{j=1}^{n} c_I(B_j).$$

**Theorem.** For any tree $A$ with subtrees $A_1, \ldots, A_m$ ($m \geq 0$) and tree $B$ with subtrees $B_1, \ldots, B_n$ ($n \geq 0$),

$$\delta(A(0), B(0)) = c_L(\lambda(A), \lambda(B)) + \sum_{k=1}^{j} c_I(B_k),$$

$$\delta(A(i), B(0)) = c_L(\lambda(A), \lambda(B)) + \sum_{k=1}^{i} c_D(A_k),$$

for $0 \leq j \leq n$ and $0 \leq i \leq m$, and

$$\delta(A(i), B(j)) = \min \{ \delta(A(i - 1), B(j - 1)) + \delta(A_k, B_j), \delta(A(i - 1), B(j)) + \delta(A_i, B(j)), \delta(A(i), B(j - 1)) + \delta(A_i, B(j)) \},$$

for $1 \leq i \leq m$ and $1 \leq j \leq n$.

**Proof.** The first two equalities follow directly from the nonnegativity of the costs. To prove the third equality, we let $S_{ij}$ denote a minimum cost sequence of edit operations which, when applied to $A(i)$, yields a tree equal to $B(j)$. For each vertex which appears in $A(i)$ and $B(j)$, draw a line joining these two occurrences. There is a line joining the roots of $A(i)$ and $B(j)$. Since the edit operations do not affect the order of the subtrees, the lines do not cross. That is, if $A_i$ and $B_j$ are joined by a line and $A_k$ and $B_l$ are joined by a line then either $i > k$ and $j > l$ or $i < k$ and $j < l$ or $i = k$ and $j = l$.

There are three cases to consider.

1. The roots of $A_i$ and $B_j$ are each touched by a line. Since they must be touched by the same line, $S_{ij}$ may be decomposed into a sequence to convert $A(i - 1)$ to $B(j - 1)$ and a sequence to convert $A_i$ to $B_j$. These two subsequences have costs $\delta(A(i - 1), B(j - 1))$ and $\delta(A_i, B_j)$ respectively.

2. Then root of $B_j$ is not touched by a line. Since the edit operation $I(B_j)$ must have been used, $S_{ij}$ may be decomposed into a sequence to convert $A(i)$ to $B(j - 1)$ and a sequence consisting of $I(B_j)$, and the costs of these subsequences are $\delta(A(i), B(j - 1))$ and $c_I(B_j)$ respectively.

3. The root of $A_i$ is not touched by a line. Since $D(A_i)$ must have been used, $S_{ij}$ may be decomposed into a sequence to convert $A(i - 1)$ to $B(j)$ and the sequence consisting of $D(A_i)$, and the costs of these two subsequences are $\delta(A(i - 1), B(j))$ and $c_D(A_i)$ respectively.

3. The algorithm

A recursive algorithm to compute $\delta(A, B)$ follows directly from the theorem. The algorithm assumes as input the following three arrays:

- $\lambda(a_i, b_j)$ which for each pair of labels $a_i$ and $b_j$ contains $c_L(a_i, b_j)$,
- $\lambda(B_k)$ which is a precomputed array which contains $c_I(B_k)$ for each subtree $B_k$ of $B$,
- $\lambda(A_k)$ which is a precomputed array which contains $c_D(A_k)$ for each subtree $A_k$ of $A$.

Following is a recursive algorithm to compute $\delta(A, B)$.

```
edit: procedure (A, B);
(assume A has subtrees A_1, \ldots, A_m and B has subtrees B_1, \ldots, B_n)
DECLARE \delta(0: m, 0: n);
\delta(0, 0) = \lambda(a(A), b(B));
do k = 1 to n;
    \delta(0, k) = \delta(0, k - 1) + \lambda(\lambda(B_k));
end;
do k = 1 to m;
    \delta(k, 0) = \delta(k - 1, 0) + \lambda(\lambda(A_k));
end;
do i = 1 to m;
do j = 1 to n;
    \delta(i, j) = \min \{ \delta(i - 1, j - 1) + \delta(\lambda(A_i), \lambda(B_j)), \delta(i - 1, j) + \lambda(\lambda(A_k)), \delta(i, j - 1) + \lambda(\lambda(B_j)) \};
end;
end;
```

185
end;
end;
return(\delta(m-n));
end edit;

Let the signature of a tree $T$ be a vector $(t_0, \ldots, t_d)$ such that $t_i$ is the number of vertices of $T$ at depth $i$. In computing $\delta(A, B)$, the procedure edit is called once for each pair of vertices at the same depth. Therefore, if $A$ and $B$ have signatures $(a_0, \ldots, a_m)$ and $(b_0, \ldots, b_n)$, then edit$(A, B)$ requires $O(\Sigma_{i=0}^{\min(m,n)} a_i b_i)$ time. Precomputation of the arrays ins and del requires $O(\Sigma_{i=0}^{m} b_i)$ and $O(\Sigma_{i=0}^{n} a_i)$ time.

References