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

Tree, tree correction, editing, algorithm

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 λ such that

(1) T has distinguished vertex called the *root* of the tree.

(2) the remaining vertices(excluding the root) are partitioned into $m \geqslant 0$ disjoint sets $T_1, \ldots, T_m$ 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 $0 \leqslant i \leqslant m$, let $T\langle i\rangle$ denote the tree obtained from T by removing subtrees $T_{i+1}, \ldots, T_m$. If A is a tree with subtrees $A_1, \ldots, A_m$ and B is a tree with subtrees $B_1, \ldots, B_n$, then A and B are *equal*, $A = B$, if $\lambda(A) = \lambda(B)$ and $m = n$ and $A_i = B_i$ for $1 \leqslant i \leqslant m$. Note that $A = A\langle m\rangle$.

Given a tree T with $\lambda(T) = s_j$ and subtrees $T_1, \ldots, T_m$;

(1) *a label change operation* $L(s_j, s_k)$ applied to T yields the tree T^* with $\lambda(T^*) = s_k$ and subtrees $T_1, \ldots, T_m$,

(2) for $0 \leqslant i \leqslant m$ and tree A, an *insert operation* $I(A)$ applied to T at i yields the tree T^* with $\lambda(T^*) = s_j$ and subtrees $T_1, \ldots, T_i, A, T_{i+1}, \ldots, T_m$,

(3) for $1 \leqslant i \leqslant m$, a *delete operation* $D(T_i)$ applied to T at i yields the tree T^* with $\lambda(T^*) = s_j$ and subtrees $T_1, \ldots, T_{i-1}, T_{i+1}, \ldots, T_m$.

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 $c_I(s_i)$ 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

$c_I(T) = \sum_{v \in T} c_I(\lambda(v))$ and $c_D(T) = \sum_{v \in T} c_D(\lambda(v))$.

For any three labels s_i, s_j and s_k, we assume $c_L(s_i, s_i) = 0$, and

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

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

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

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

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

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

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

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

$$\delta(A\langle i\rangle, B\langle j-1\rangle) + c_I(B_j),$$

$$\delta(A\langle i-1\rangle, B\langle j\rangle) + c_D(A_i)\},$$

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

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 $A\langle i\rangle$, yields a tree equal to $B\langle j\rangle$. For each vertex which appears in $A\langle i\rangle$ and $B\langle j\rangle$, draw a line joining these two occurrences. There is a line joining the roots of $A\langle i\rangle$ and $B\langle j\rangle$. 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 $A\langle i-1\rangle$ to $B\langle j-1\rangle$ and a sequence to convert A_i to B_j. These two subsequences have costs $\delta(A\langle i-1\rangle, B\langle j-1\rangle)$ 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 $A\langle i\rangle$ to $B\langle j-1\rangle$ and a sequence consisting of $I(B_j)$, and the costs of these subsequences are $\delta(A\langle i\rangle, B\langle j-1\rangle)$ 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 $A\langle i-1\rangle$ to $B\langle j\rangle$ and the sequence consisting of $D(A_i)$, and the costs of these two subsequences are $\delta(A\langle i-1\rangle, B\langle j\rangle)$ 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:

lab(s_i, s_j) which for each pair of labels s_i and s_j contains $c_L(s_i, s_j)$,

ins(B_k) which is a precomputed array which contains $c_I(B_k)$ for each subtree B_k of B,

del(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 A1, ..., Am and B subtrees B1, ..., Bn)
   DECLARE δ(0: m, 0: n);
   δ(0, 0) = lab(λ(A), λ(B));
   do k = 1 to n;
      δ(0, k) = δ(0, k − 1) + ins(Bk);
   end;
   do k = 1 to m;
      δ(k, 0) = δ(k − 1, 0) + del(Ak);
   end;
   do i = 1 to m;
      do j = 1 to n;
         δ(i, j) = min (δ(i − 1, j − 1) + edit(Ai, Bj),
                        δ(i, j − 1) + ins(Bj),
                        δ(i − 1, j) + del(Ai));
```

```
        end;
    end;
    return(δ(m,n));
end edit;
```

Let the *signature* of a tree T be a vector $(t_0, ..., t_d)$ 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 $(a_0, ..., a_m)$ and $(b_0, ..., b_n)$, 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 $O(\Sigma_{i=0}^{n} b_i)$ and $O(\Sigma_{i=0}^{m} a_i)$ time.

References

[1] A.V. Aho, D.S. Hirschberg and J.D. Ullman, Bounds on the complexity of the longest common subsequence problem. J. Assoc. Comput. Mach 23 (1) (1976) 1–12.

[2] D. Sankoff, Matching sequences under deletion/insertion constraints, Proc. Nat. Acad. Sci. 69 (1) (1972) 4–6.

[3] R.A. Wagner and M.J. Fischer, The string-to-string correction problem, J. Assoc. Comput. Mach 21 (1974) 168–173

[4] C.K. Wong and A.K. Chandra, Bounds for the string editing problem, J. Assoc. Comput. Mach 23 (1976) 13–16.