Vol.2010-AL-129 No.2
2010/3/5

無順序木の編集距離計算のための厳密アルゴリズム

阿久津 達也[†1] 深川 大路[†2]
高須 淳宏[†2] 田村 武幸[†1]

本稿では無順序木の編集距離を計算するためのいくつかのアルゴリズムを示す．一つは，各編集操作に単位コストが割り当てられる場合の固定パラメータ・アルゴリズムである．そして最大の編集距離 k を固定パラメータとし，大きい方の入力木のサイズを n とする時，このアルゴリズムの時間計算量が $O(2.62^k \cdot poly(n))$ となることを示す．もう一つは，最大共通部分木の最大出次数が定数で抑えられた場合についての多項式時間アルゴリズムであり，出次数（子の個数）の最大値を D とした場合に $O(n^{2D})$ 時間で動作することを示す．

Exact Algorithms for Computing Tree Edit Distance between Unordered Trees

Tatsuya Akutsu,[†1] Daiji Fukagawa,[†2]
Atsuhiro Takasu[†2] and Takeyuki Tamura[†1]

This article (non-reviewed technical report) presents a fixed-parameter algorithm for the tree edit distance problem for unordered trees under the unit cost model that works in $O(2.62^k \cdot poly(n))$ time where the parameter k is the maximum bound of the edit distance and n is the maximum size of input trees. This article also presents a polynomial time algorithm for the case where the maximum outdegree of the largest common subtree is bounded by a constant. When the maximum outdegree is bounded by D, the algorithm works in $O(n^{2D})$ time.

†1 京都大学 化学研究所 バイオインフォマティクスセンター
Bioinformatics Center, Institute for Chemical Research, Kyoto University
†2 国立情報学研究所
National Institute of Informatics

1. Introduction

Tree pattern matching plays an important role in such application areas as computational biology, XML databases and image analysis[1],[10]. Though various measures have been proposed for computing the similarity between trees, the *edit distance* between rooted trees has been well-studied.

For the tree edit distance problem for ordered trees, Tai developed an $O(n^6)$ time algorithm[11], from which several improvements followed. Recently, Demaine *et al.* developed an $O(n^3)$ time algorithm and showed that this bound is optimal under some computation strategy[3].

However, in some applications, it is preferable to regard input trees as unordered trees. At least, in many applications, more flexible matching can be made possible if input trees are regarded as unordered trees. Unfortunately, Zhang *et al.* proved that the tree edit distance problem for unordered trees is NP-hard[12]. Furthermore, Zhang and Jiang proved that it is MAX SNP-hard[13], which means that there exists no polynomial time approximation scheme unless P=NP.

Some optimal algorithms were proposed for restricted cases[7],[10],[14]. Shasha *et al.* developed a fixed-parameter algorithm when the parameter is the number of leaves[10]. Halldórsson and Tanaka developed a polynomial time algorithm for the case of bounded number of branching nodes[7]. Zhang developed a polynomial time algorithm under restricted editing operations[14]. For a related tree inclusion problem, Kilpeläinen and Mannila showed that it is solved in polynomial time if the maximum degree is bounded by a constant[9]. However, the above cases do not cover all important cases and thus it is worthy to develop polynomial time algorithms for other cases.

In this article, we present a fixed-parameter algorithm that works in $O(2.62^k \cdot poly(n))$ time where the parameter k is the edit distance and n is the maximum size of input trees. We also present polynomial time algorithms for the case in which the maximum degree of the largest common subtree (based on editing operations) is bounded by a constant. Due to space limitation, some of the proofs are omitted in this article.

2. Preliminaries

We briefly review the definitions of tree edit distance, edit distance mapping and largest common subtree (see also Fig. 1) for rooted, labeled and unordered trees[1),12),13)].

Let T be a rooted unordered tree where each node v has a label $\ell(v)$ over an alphabet Σ. $r(T)$ and $V(T)$ denote the root of T and the set of nodes in T, respectively. For a node $v \in V(T)$, $T - v$ denotes the tree obtained by deleting v from T, $p(v)$ denotes the parent of v, $chd(v)$ denotes the set of children of v, $deg(v)$ denotes the outdegree of v, $anc(v)$ denotes the set of ancestors of v, $des(v)$ denotes the set of descendants of v. It should be noted that $deg(v) = |chd(v)|$ holds.

$T(v)$ denotes the subtree induced by v and its descendants. For a subtree T' of T, $T - T'$ denotes the tree obtained by deleting all nodes in T' from T. The *depth* of a node v is the length of the path from the root to v and is denoted by $depth(v)$. For a set of nodes $\{v_1, v_2, \ldots, v_h\} \subseteq V(T)$, $LCA(\{v_1, v_2, \ldots, v_h\})$ denotes the lowest common ancestor of $v_1, v_2, \ldots, v_h$. If T_1 and T_2 are isomorphic including label information, we write $T_1 \approx T_2$. In the analysis of algorithms, n denotes $\max\{|V(T_1)|, |V(T_2)|\}$.

An *edit operation on a tree* T is either a deletion, an insertion, or a substitution, where each operation is defined as follows (see also Fig. 1):

Deletion: Delete a non-root node v in T with parent u, making the children of v become children of u. The children are inserted in the place of v into the set of the children of u.

Insertion: Inverse of delete. Insert a node v as a child of u in T, making v the parent of some of the children of u.

Substitution: Change the label of a node v in T.

We assign a *cost* for each editing operation: $\gamma(a, b)$ denotes the cost of substituting a node with label a to label b, $\gamma(a, \epsilon)$ denotes the cost of deleting a node labeled with a, and $\gamma(\epsilon, a)$ denotes the cost of inserting a node labeled with a.

The *edit distance* between two unordered trees T_1 and T_2 is defined as the cost of the minimum cost sequence of editing operations that transforms T_1 to T_2. We use $dist(T_1, T_2)$ to denote the edit distance between T_1 and T_2. In this article, we adopt the following standard assumption so that $dist(T_1, T_2)$ becomes a distance metric[1),12)]: $\gamma(a, b) \geq 0$ for any $(a, b) \in \Sigma' \times \Sigma'$, $\gamma(a, a) = 0$ for any $a \in \Sigma'$, $\gamma(a, b) = \gamma(b, a)$ for any $(a, b) \in \Sigma' \times \Sigma'$, $\gamma(a, c) \leq \gamma(a, b) + \gamma(b, c)$ for any $a, b, c \in \Sigma' \times \Sigma' \times \Sigma'$, where $\Sigma' = \Sigma \cup \{\epsilon\}$. We call T_2 a *subtree* of T_1 if T_2 is obtained from T_1 only by deletion operations[⋆1].

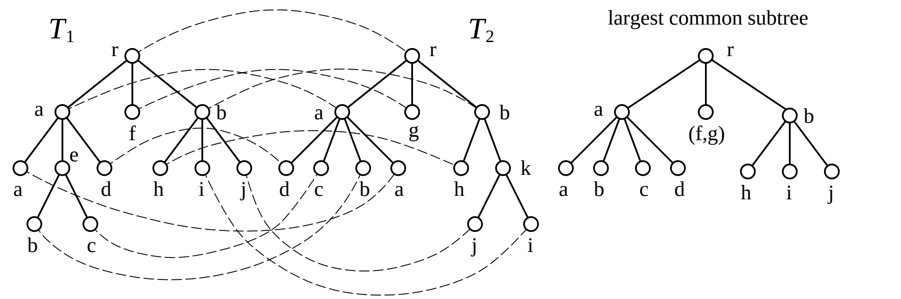

Fig.1 Example of tree edit operation, mapping, and largest common subtree under the unit cost model. T_2 is obtained from T_1 by deletion of node (labeled with) e, insertion of node k and substitution of node f. The corresponding mapping M is shown by broken curves. The largest common subtree is shown in the right-hand side, where the labels of the original nodes are shown in place of the original node pairs. The node labeled (f,g) is not included in the usual largest common subtree[13)].

It is known that there exists a close relationship between the edit distance and the *edit distance mapping* (or just *mapping*)[1),12)]. $M \subseteq V(T_1) \times V(T_2)$ is called a mapping if the following conditions are satisfied for any two pairs (v_1, w_1), $(v_2, w_2) \in M$: $v_1 = v_2$ iff $w_1 = w_2$, v_1 is an ancestor of v_2 iff w_1 is an ancestor of w_2. Let I_1 and I_2 be the sets of nodes in $V(T_1)$ and $V(T_2)$ not appearing in M, respectively. Then, it is known[1),12)] that the following relation holds:

$$dist(T_1, T_2) = \min_M \left\{ \sum_{v \in I_1} \gamma(\ell(v), \epsilon) + \sum_{v \in I_2} \gamma(\epsilon, \ell(v)) + \sum_{(u,v) \in M} \gamma(\ell(u), \ell(v)) \right\}.$$

Here we define a *score function* $f(u, v)$ for $(u, v) \in V(T_1) \times V(T_2)$ by

⋆1 We also use the *subtree* for denoting a subgraph of a tree. However, the meaning of the subtree is clear from the context and thus there is no confusion.

$f(u,v) = \gamma(\ell(u),\epsilon) + \gamma(\epsilon,\ell(v)) - \gamma(\ell(u),\ell(v))$.

It is seen that $f(u,v) = f(v,u) \geq 0$ holds. It should also be noted that under the unit cost model (i.e., $\gamma(a,b) = 1$ for all $\ell(a) \neq \ell(b)$), $f(v,v) = 2$ and $f(u,v) = 1$ hold for $\ell(u) \neq \ell(v)$. Let $score(M)$ be the score of a mapping M defined by $score(M) = \sum_{(u,v)\in M} f(u,v)$. Let M_{OPT} be the mapping with the maximum score. Then, we can see from the definition that the following property holds:

$$dist(T_1,T_2) = \sum_{u\in V(T_1)} \gamma(\ell(u),\epsilon) + \sum_{v\in V(T_2)} \gamma(\epsilon,\ell(v)) - score(M_{OPT}).$$

If M consists of pairs of identical labels, the subtree obtained by deleting nodes not appearing in M from T_1 is isomorphic to the subtree obtained by deleting nodes not appearing in M from T_2. Such a tree is called a *common subtree* between T_1 and T_2. In this article, a subtree of T_1 (or T_2) induced by the nodes appearing in M is also called a common subtree even if M contains some pairs of non-identical labels. The *largest common subtree* (LCST, in short) is defined as the common subtree with the maximum score.

Though the edit distance problem for unordered trees is NP-hard, it can be solved (in exponential time) using a dynamic programming (DP) algorithm[1),7)]. For a forest (i.e., a set of unordered trees) F, $roots(F)$ denotes a set of the roots of trees in F. We define $\delta(F_1,F_2)$ between two unordered forests F_1 and F_2 by the following DP procedure:

$$\delta(F_1,\epsilon) = \sum_{u\in V(F_1)} \gamma(\ell(u),\epsilon),$$

$$\delta(\epsilon,F_2) = \sum_{v\in V(F_2)} \gamma(\epsilon,\ell(v)),$$

$$\delta(F_1,F_2) = \min \begin{cases} \min_{u\in roots(F_1)} \{\delta(F_1 - u, F_2) + \gamma(\ell(u),\epsilon)\}, \\ \min_{v\in roots(F_2)} \{\delta(F_1, F_2 - v) + \gamma(\epsilon,\ell(v))\}, \\ \min_{(u,v)\in roots(F_1)\times roots(F_2)} \{\delta(F_1 - T_1(u), F_2 - T_2(v)) \\ \qquad +\delta(T_1(u) - u, T_2(v) - v) + \gamma(\ell(u),\ell(v))\}. \end{cases}$$

Then, it is seen that $dist(T_1,T_2) = \delta(T_1,T_2)$ holds from 1), 7).

3. Fixed-Parameter Algorithm

In this section, we present an $O(2.62^k \cdot poly(n))$ time algorithm for the tree edit distance problem between two unordered trees, where the parameter is the edit distance. Though we consider the unit cost model here for simplicity, the algorithm can be extended for a more general case in which costs of edit operations are integers. For details of fixed-parameter algorithms, refer to 5), 6).

The following lemmas (proofs omitted) are important for developing the fixed parameter algorithm.

Lemma3.1 (See also Fig. 2) Let r_1 and r_2 be the roots of T_1 and T_2 respectively. Suppose that for any pair $(u,v) \in chd(r_1) \times chd(r_2)$, $dist(T_1(u),T_2(v)) \geq 1$ holds. Suppose that u is an arbitrary child of r_1 or r_2 such that $|T(u)|$ is the largest, where we assume w.l.o.g. that $u \in chd(r_1)$. Then, one of the following holds, where M is the optimal edit distance mapping:

- u does not appear in M,
- $(u,v) \in M$ for some child v of r_2, where $dist(T_1(u),T_2(v)) \geq 1$,
- $(u,v) \in M$ for some descendant v of some child of r_2, where $dist(T_1(u),T_2(v)) \geq 1$.

Lemma3.2 Let $x_1,\ldots,x_{l_1}$ and $y_1,\ldots,y_{l_2}$ be children of $r(T_1)$ and $r(T_2)$, respectively. If $T_1(x_i) \approx T_2(y_j)$ holds, $dist(T_1,T_2) = dist(T_1 - T_1(x_i), T_2 - T_2(y_j))$ holds.

Before presenting an $O(2.62^k \cdot poly(n))$ time algorithm, we begin with an $O(2^k \cdot k! \cdot poly(n))$ time algorithm because it is easier to understand. The following is a pseudocode of this simpler algorithm, where it works in a recursive manner and decides whether or not $dist(T_1,T_2) \leq k$ holds for given trees T_1 and T_2. In the procedure, we let $mindist(u,v) = \mathrm{argmin}_h\{FpDist0(T_1(u),T_2(v),h) = TRUE\}$, where $mindist(u,v) = \infty$ if $FpDist0(T_1(u),T_2(v),h) = FALSE$ for all h. We can assume that this value can be computed before computation of $FpDist0(T_1,T_2,k)$ by execution of $FpDist0(T_1(u),T_2(v),h)$ for $h = 0,\ldots,k$ for all $(u,v) \in des(r(T_1)) \times des(r(T_2))$ in a bottom up manner (i.e., results of $FpDist0(T_1(u),T_2(v),h)$ are stored in a DP table). We can also assume $\ell(r(T_1)) = \ell(r(T_2))$. Otherwise, it is enough to execute $FpDist0(T_1,T_2,k-1)$ instead of $FpDist0(T_1,T_2,k)$.

```
Procedure $FpDist0(T_1, T_2, k)$
if $k < 0$ then return FALSE;
if $|T_1| = 0$ or $|T_2| = 0$ then
  if $\max(|T_1|, |T_2|) \leq k$ then return TRUE   else return FALSE;
if $|T_1| = |T_2| = 1$ then
  if $\gamma(\ell(r(T_1)), \ell(r(T_2))) \leq k$ then return TRUE   else return FALSE;
if $T_1(x_i) \approx T_2(y_j)$ for some $x_i \in chd(r(T_1))$, $y_j \in chd(r(T_2))$ then
  return $FpDist0(T_1 - T_1(x_i), T_2 - T_2(y_j), k)$;
$W_1 \leftarrow \emptyset$; $W_2 \leftarrow \emptyset$;
while $|W_1| \leq k$ and $|W_2| \leq k$ do
  Let $w$ be the node in $chd(r_1) \cup chd(r_2) - W_1 - W_2$ such that $|T(w)|$ is
    the largest;
  if such a node does not exist then                                   (#1)
    if $|W_1| \neq |W_2|$ then return FALSE;
    Compute the minimum weight bipartite matching between $W_1$ and $W_2$
      where $weight(u, v) = mindist(u, v)$;
    if the minimum cost $\leq k$ then return TRUE else return FALSE;
  if $w \in chd(r(T_1))$ and $FpDist0(T_1 - w, T_2, k - 1)$ is TRUE
  then return TRUE;                                                   (#2)
  if $w \in chd(r(T_2))$ and $FpDist0(T_1, T_2 - w, k - 1)$ is TRUE
  then return TRUE;                                                   (#3)
  if $w \in chd(r(T_1))$ then $W_1 \leftarrow W_1 \cup \{w\}$ else $W_2 \leftarrow W_2 \cup \{w\}$;
return FALSE;
```

Theorem3.3 $FpDist0(T_1, T_2, k)$ decides whether or not $dist(T_1, T_2) \leq k$ holds in $O(2^k \cdot k! \cdot poly(n))$ time.

Proof. First, we show the correctness of the algorithm. If either $|T_1| = 0$, $|T_2| = 0$ or $|T_1| = |T_2| = 1$ holds, the algorithm obviously returns the correct value (TRUE or FALSE). If $T_1(x_i) \approx T_2(y_j)$ holds, the correctness follows from induction on the total size of trees and Lemma 3.2. Otherwise, $dist(T_1(x_i), T_2(y_j)) \geq 1$ must hold for all pairs (x_i, y_j).

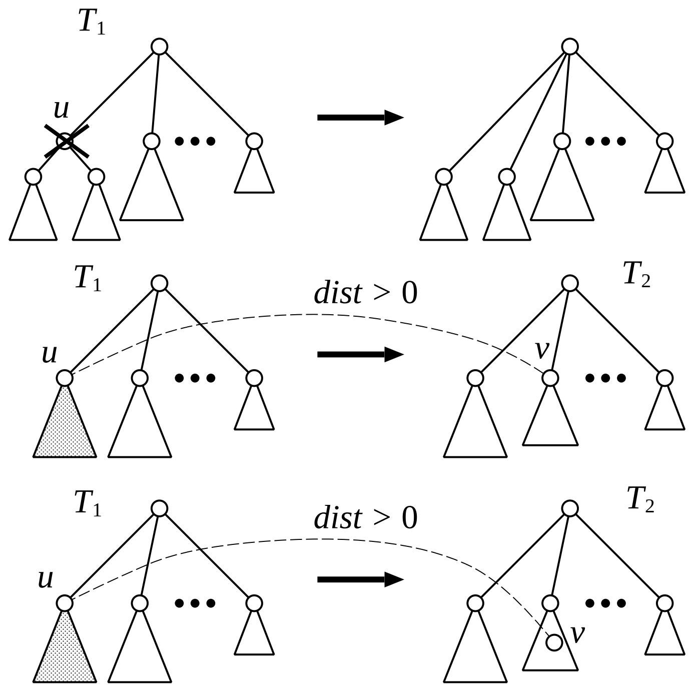

Fig.2 Three cases considered in Lemma 3.1.

Suppose that part (#1) is executed. Then, all the children of $r(T_1)$ and $r(T_2)$ are included in $W_1 \cup W_2$. Since deletion of any child should have been taken care by (#2) and (#3), FALSE should be returned if $|W_1| \neq |W_2|$ holds. Otherwise, there must exist a one-to-one mapping between $chd(r(T_1))$ and $chd(r(T_2))$. Then, $dist(T_1, T_2) \leq k$ holds if and only if the weight of the minimum weight matching is at most k.

Suppose that some of the children of $r(T_1)$ or $r(T_2)$ is deleted in the optimal mapping, then such a node should have been correctly taken care by (#2) or (#3).

Suppose that $|W_1| > k$ or $|W_2| > k$ holds. We can assume w.l.o.g. that $|W_1| > k$ holds. Since we can assume that no node in $W_1 \cup W_2$ is deleted, each node u in W_1 is mapped to some node in W_2, or some child v of $r(T_2)$ or its descendant such that $v \notin W_2$. If u is mapped to a node in W_2, it contributes to the total distance by at least

1 because all isomorphic pairs $(T_1(x_i), T_2(y_j))$s are removed beforehand. Otherwise, $|T_1(u)| \geq |T_2(v)|$ holds and thus we can see from Lemma 3.1 that u contributes to the total distance by at least 1. Thus, FALSE should be output in this case. This completes the proof of the correctness of $FpDist0(T_1, T_2, k)$.

Next, we analyze the time complexity. Let $f(k, n)$ be the time complexity of $FpDist0(T_1, T_2, k)$, where $n = \max(|T_1|, |T_2|)$. If either $|T_1| = 0$, $|T_2| = 0$ or $|T_1| = |T_2| = 1$ holds, $FpDist0(T_1, T_2, k)$ takes $O(1)$ time. $T_1(x_i) \approx T_2(y_j)$ can be tested in $O(1)$ time per pair if we pre-process T_1 and T_2 in linear time so that the signature (with $O(\log n)$ bits) of each $T_i(v)$ is computed4). Therefore, whether or not such a pair exists can be tested in $O(n^2)$ time. Execution of the while loop can be done in $O(kn)$ time in total except minimum weight matching and recursive call of $FpDist0$. Minimum weight bipartite matching can be done in $O(k^3)$ $(\leq O(n^3))$ time2). Therefore, we have $f(k, n) \leq 2k \cdot f(k-1, n) + O(n^3)$. From this, we can show that $f(k, n)$ is $O(2^k \cdot k! \cdot n^3)$ as follows:

$$\begin{aligned} f(k,n) &\leq O(n^3) \cdot \left[1 + 2k + 2k \cdot 2(k-1) + 2k \cdot 2(k-1) \cdot 2(k-2) + \cdots + 2^k \cdot k!\right] \\ &\leq O(n^3) \cdot \left[(1 + 2 + 2^2 + \cdots + 2^k) \cdot k!\right] \quad \leq \quad O(n^3) \cdot \left[2^{k+1} \cdot k!\right], \end{aligned}$$

where we can assume that $f(0, n)$ is $O(n)$ because the tree isomorphism problem can be solved in $O(n)$ time4). Here, it should be noted that $FpDist0$ should be executed $k + 1$ $(\leq O(n))$ times for $O(n^2)$ pairs. Hence, the total time complexity is $O(2^k \cdot k! \cdot n^6)$. □

Now, we present our main algorithm which we call $FpDist$. This algorithm is almost the same as the original $FpDist0$ except that we inherit the sets W_1 and W_2 of the caller. By inheriting the sets W_1 and W_2, we can reduce the number of iterations in the while loops. The following is the pseudo code of $FpDist0$, where it is invoked as $FpDist(T_1, T_2, k, \emptyset, \emptyset)$.

```
**Procedure** $FpDist(T_1, T_2, k, W_1, W_2)$
**if** $k < 0$ **then return** FALSE;
**if** $|T_1| = 0$ or $|T_2| = 0$ **then**
  **if** $\max(|T_1|, |T_2|) \leq k$ **then return** TRUE    **else return** FALSE;
**if** $|T_1| = |T_2| = 1$ **then**
  **if** $\gamma(\ell(r(T_1)), \ell(r(T_2))) \leq k$ **then return** TRUE    **else return** FALSE;
**if** $T_1(x_i) \approx T_2(y_j)$ for some $x_i \in chd(r(T_1))$, $y_j \in chd(r(T_2))$ **then**
  **return** $FpDist(T_1 - T_1(x_i), T_2 - T_2(y_j), k, W_1, W_2)$;                    (*)
**while** $|W_1| \leq k$ and $|W_2| \leq k$ **do**
  Let $w$ be the node in $chd(r_1) \cup chd(r_2) - W_1 - W_2$ such that$|T(w)|$ is
    the largest;
  **if** such a node does not exist **then**                                              (#1)
    **if** $|W_1| \neq |W_2|$ **then return** FALSE;
    Compute the minimum weight bipartite matching between $W_1$ and $W_2$
      where $weight(u, v) = mindist(u, v)$;
    **if** the minimum cost $\leq k$ **then return** TRUE **else return** FALSE;
  **if** $w \in chd(r(T_1))$ and $FpDist(T_1 - w, T_2, k-1, W_1, W_2)$ is TRUE
  **then return** TRUE;                                                                    (#2)
  **if** $w \in chd(r(T_2))$ and $FpDist(T_1, T_2 - w, k-1, W_1, W_2)$ is TRUE
  **then return** TRUE;                                                                    (#3)
  **if** $w \in chd(r(T_1))$ **then** $W_1 \leftarrow W_1 \cup \{w\}$ **else** $W_2 \leftarrow W_2 \cup \{w\}$;
**return** FALSE;
```

Theorem3.4 $FpDist(T_1, T_2, k)$ decides whether or not $dist(T_1, T_2) \leq k$ holds in $O(2.62^k \cdot poly(n))$ time.

Proof. We omit the proof of the correctness and analyze the time complexity only.

Let $g(k, n, s)$ denote the time complexity of $FpDist(T_1, T_2, k, W_1, W_2)$, where $n = \max(|T_1|, |T_2|)$ and $s = |W_1| + |W_2|$. The total time complexity is given by $g(k, n, 0)$. There exists a polynomial $G(n)$ that satisfies $g(k, n, s) \leq G(n)$ for $s > 2k$ since the algorithm does not execute the while loop in such a case. For the other cases, we have the following inequality:

$$g(k, n, s) \leq g(k-1, n, s) + g(k-1, n, s+1) + \cdots + g(k-1, n, 2k) + T(n)$$

where $T(n)$ is another polynomial which satisfies $T(n) = O(n^3)$. We can prove $g(k, n, 2k - i) \leq F_{i+2} \cdot G(n) + F_{i+1} \cdot T(n)$, where F_i is the i-th Fibonacci number, by mathematical induction.

$$\begin{aligned}
g(k,n,2k) &\le g(k-1,n,2k)+T(n)\le G(n)+T(n),\\
g(k,n,2k-1) &\le g(k-1,n,2k-1)+g(k-1,n,2k)+T(n)\le 2G(n)+T(n),\\
g(k,n,2k-i) &\le \sum_{j=0}^{i} g(k-1,n,2k-j)+T(n)\\
&\le \sum_{j=2}^{i} g(k-1,n,2k-j)+2G(n)+T(n)\\
&= \sum_{j=0}^{i-2} g(k-1,n,2(k-1)-j)+2G(n)+T(n)\\
&\le \sum_{j=0}^{i-2}\{F_{j+2}\cdot G(n)+F_{j+1}\cdot T(n)\}+2G(n)+T(n)\\
&= F_{i+2}\cdot G(n)+F_{i+1}\cdot T(n).
\end{aligned}$$

The last equality comes from $\sum_{j=0}^{i} F_j = F_{i+2}-1$. Hence the total time complexity is $g(k,n,0)=F_{2k+2}\cdot G(n)+F_{2k+1}\cdot T(n)=O(2.62^k\cdot poly(n))$ since $F_{2k}\le \frac{1}{\sqrt{5}}\cdot(\frac{1+\sqrt{5}}{2})^{2k}=O(2.62^k)$. □

4. Algorithms for Bounded Degree LCST

Though the edit distance problem is NP-hard for unordered trees, we can obtain an exact solution in polynomial time if the maximum degree of the corresponding LCST (i.e., LCST obtained from an optimal mapping) is bounded by a constant. In this section, we first present a basic algorithm and then present improved algorithms, where we only need to assume that the distance satisfies the conditions on a distance metric.

4.1 Basic Algorithm

The basic algorithm is quite simple and is based on a simple DP procedure. As explained in Section 2, computation of the edit distance is equivalent to computation of the LCST. Therefore, we focus on computation of the LCST in this section. For $x\in V(T_1)$ and $y\in V(T_2)$, let $S_D(x,y)$ be the size of LCST between $T_1(x)$ and $T_2(y)$ under the condition that the maximum outdegree (i.e., maximum $deg(v)$) of LCST is at most D. For the simplicity, we begin with the case of $D=2$. The following is a DP procedure of the algorithm.

$$S_2(x,y)=\max\begin{cases}
f(x,y), & -(*1)\\
\max_{x_1,x_2\in des(x),y_1,y_2\in des(y)}\{S_2(x_1,y_1)+S_2(x_2,y_2)+f(x,y)\}, & \\
 & -(*2)\\
\max_{x_1\in des(x),y_1\in des(x)}\{S_2(x_1,y_1)+f(x,y)\}, & -(*3)\\
\max_{y_1\in des(y)} S_2(x,y_1), & -(*4)\\
\max_{x_1\in des(x)} S_2(x_1,y), & -(*5)
\end{cases}$$

where $x_1\notin des(x_2)\cup\{x_2\}$, $x_2\notin des(x_1)\cup\{x_1\}$, $y_1\notin des(y_2)\cup\{y_2\}$ and $y_2\notin des(y_1)\cup\{y_1\}$ must hold. It is to be noted that the maximum degree of T_1 and T_2 need not be bounded. Let $BdDist_2$ denote the above DP algorithm. Then, we have the following theorem.

Theorem4.1 $BdDist_2$ computes the edit distance in $O(n^6)$ time if the maximum outdegree of the corresponding largest common subtree is at most 2.

Proof. We consider an optimal mapping M between $T_1(x)$ and $T_2(y)$. Then, either one of the following must hold:

(i) x corresponds to y,

(ii) x corresponds to a descendant of y,

(iii) y corresponds to a descendant of x.

It is to be noted that either x or y must appear in M. Otherwise, we can increase or keep the score of LCST by adding (x,y) to M since $f(x,y)\ge 0$ holds for any pair of nodes (x,y).

If x and y are leaves, LCST is clearly computed by (*1) and the other parts are not executed. Otherwise, cases (ii) and (iii) are covered by (*4) and (*5), respectively. For case (i), there are two possibilities:

- the root of $LCST(T_1(x),T_2(y))$ has two children,
- the root of $LCST(T_1(x),T_2(y))$ has only one child.

Then, the former case is covered by (*2) and the latter case is covered by (*3). Therefore, $BdDist_2$ correctly computes LCST if $D=2$.

Next, we analyze the time complexity. Clearly, $S_2(x,y)$ must be computed for $O(n^2)$ pairs. For each pair, $O(n^4)$ combinations of four children are examined in (*2), where

$O(1)$ time is enough per combination. Since (*2) is the most time consuming part, the total time complexity is $O(n^2) \times O(n^4) = O(n^6)$. □ □

We can easily extend $BdDist_2$ for arbitrary fixed D. The following is the DP algorithm, which is denoted by $BdDist_D$.

$$S_D(x,y) = \max\begin{cases} \max_{h=0,\ldots,D}\left\{\max_{x_1,\ldots,x_h\in des(x),y_1,\ldots,y_h\in des(y)}\left[\left(\sum_{i=1}^{h} S_D(x_i,y_i)\right)+f(x,y)\right]\right\}, \\ \max_{y_1\in des(y)} S_D(x,y_1), \\ \max_{x_1\in des(x)} S_D(x_1,y), \end{cases}$$

where $x_i \notin des(x_j) \cup \{x_j\}$ and $y_i \notin des(y_j) \cup \{y_j\}$ must be satisfied for any $i \neq j$. Clearly, we have:

Corollary4.2 $BdDist_D$ computes the edit distance between two unordered trees in $O(n^{2+2D})$ time if the maximum outdegree of the corresponding largest common subtree is at most D, where D is a constant.

4.2 Improvements

Though $BdDist_D$ works in polynomial time, it is not practical because the time complexity is $O(n^6)$ even for $D = 2$. For the case of $D = 2$, we can develop an improved algorithm that works in $O(n^2)$ time.

Theorem4.3 The edit distance between two unordered trees can be computed in $O(n^2)$ time if the maximum outdegree of the corresponding largest common subtrees is at most 2.

It is very unclear whether or not this result can be extended for the cases of $D > 2$. Therefore, it is worthy to try to develop a simple and improved algorithm based on another idea. Here, we present an algorithm (called $LcaBdDist_D$) using the lowest common ancestor, which works in $O(n^{2D})$ time for a fixed D. The following proposition directly follows from the definition of tree edit distance mapping.

Proposition4.4 If $(x_1,y_1),(x_2,y_2),\ldots,(x_h,y_h)$ are the children of (x,y) in the LCST, x and y are common ancestors of $x_1,x_2,\ldots,x_h$ and $y_1,y_2,\ldots,y_h$, respectively.

Based on this proposition, we can compute $S_D(x,y)$ by using the following procedure, where $LCA(x)$ is defined as $p(x)$.

```
Procedure $LcaBdDist_D(T_1,T_2)$
for all $(x,y) \in V(T_1)\times V(T_2)$ do $S_D(x,y) \leftarrow f(x,y)$;
for all $h \in \{1,\ldots,D\}$ do
  for all $x_1,\ldots,x_h$ such that $x_i \notin des(x_j)\cup\{x_j\}$ holds for all $i \neq j$ do
    $x_a \leftarrow LCA(x_1,\ldots,x_h)$;
    for all $y_1,\ldots,y_h$ such that $y_i \notin des(y_j)\cup\{y_j\}$ holds for all $i \neq j$ do
      $y_a \leftarrow LCA(y_1,\ldots,y_h)$;
      for all $(x,y)$ such that $x \in anc(x_a)\cup\{x_a\}$ and $y \in anc(y_a)\cup\{y_a\}$ do
        $S_D(x,y) \leftarrow \max\{S_D(x,y), S_D(x_1,y_1)+\cdots+S_D(x_h,y_h)+f(x,y)\}$;
```

In the above, the ordering of combinations is not specified. But, it must be ordered so that $S_D(x_i,y_i)$ is used only after its final value is determined. For that purpose, it is enough to partially sort the combinations by using the post ordering of their LCAs.

Though this algorithm still needs $O(n^{2D+2})$ time, we can reduce $O(n^2)$ factor. For that purpose, we use an additional DP table $S_D^{--}(x,y)$ which stores the score of LCST between $T_1(x)$ and $T_2(y)$ under the condition that x corresponds to y but $f(x,y)$ is not counted in $S_D^{--}(x,y)$. This means that x or y can be mapped to another node in later updates. The following is a pseudocode for the improved algorithm using $S_D^{--}(x,y)$.

```
Procedure $LcaBdDist2_D(T_1,T_2)$
for all $(x,y) \in V(T_1)\times V(T_2)$ do $S_D(x,y) \leftarrow f(x,y)$; $S_D^{--}(x,y) \leftarrow 0$;
for all $h \in \{1,\ldots,D\}$ do
  for all $x_1,\ldots,x_h$ such that $x_i \notin des(x_j)\cup\{x_j\}$ holds for all $i \neq j$ do
    $x_a \leftarrow LCA(x_1,\ldots,x_h)$;
    for all $y_1,\ldots,y_h$ such that $y_i \notin des(y_j)\cup\{y_j\}$ holds for all $i \neq j$ do
      $y_a \leftarrow LCA(y_1,\ldots,y_h)$;
      if $h = 1$ then
        for all $x \in anc(x_a)\cup\{x_a\}$ and $y \in anc(y_a)\cup\{y_a\}$ do
          $S_D(x,y) \leftarrow \max\{S_D(x,y), S_D^{--}(x_a,y_a)+f(x,y)\}$;
        for all $x \in anc(x_a)$ and $y \in anc(y_a)$ do
          $S_D(x,y) \leftarrow \max\{S_D(x,y), S_D(x_a,y_a)+f(x,y)\}$;
      else $S_D^{--}(x,y) \leftarrow \max\{S_D^{--}(x,y), S_D(x_1,y_1)+\cdots+S_D(x_h,y_h)+f(x_b,y_b)\}$;
```

It is to be noted that the case of $h = 1$ (i.e., (x_1, y_1)) must be examined after all the cases of $h \geq 2$ such that $(x_a, y_a) = (x_1, y_1)$ are examined. It is straight-forward to arrange the combinations in such a way.

The correctness of the improved algorithm directly follows from the definitions of $S_D(x, y)$ and $S_D^{--}(x, y)$. It is also straight-forward to see that the time complexity is $O(n^{2D})$ since examination of $O(n^2)$ ancestor pairs are avoided in the improved algorithm.

Theorem4.5 *LcaBdDist*2_D computes the edit distance between two unordered trees in $O(n^{2D})$ time if the maximum outdegree of the corresponding largest common subtrees is at most D, where D is a constant such that $D \geq 2$.

5. Concluding Remarks

We have presented an $O(2.62^k \cdot poly(n))$ time algorithm and an $O(n^{2D})$ time algorithm for the edit distance problem for unordered trees, where k is the maximum bound of the edit distance and D is the maximum degree of the largest common subtree. For the former algorithm, improvement of exponential factor is left as an open problem. However, the factor of 2.62^k is not large for moderate values of k. Therefore, it might be possible to develop a practical algorithm for comparing similar unordered trees based on this algorithm along with existing heuristics[8]. Such a development is left as future work. For the latter algorithm, it is unclear whether we can develop a fixed-parameter algorithm when D is regarded as a parameter. Therefore, deciding the complexity on D is left as an open problem.

References

1) Bille, P.: A survey on tree edit distance and related problem, *Theoretical Computer Science*, Vol.337, pp.217–239 (2005).
2) Cormen, T.H., Leiserson, C.E., Rivest, R. and Stein, C.: *Introduction to Algorithms*, second ed., The MIT Press (2001).
3) Demaine, E., Mozes, S., Rossman, B. and Weimann, O.: An optimal decomposition algorithm for tree edit distance, *Proc. 34th International Colloquium on Automata, Languages and Programming*, LNCS, Vol.4596, pp.146–157 (2007).
4) Dinitz, Y., Itai, A. and Rodeh, M.: On an algorithm of Zemlyachenko for subtree isomorphism, *Information Processing Letters*, Vol.70, pp.141–146 (1999).
5) Downey, R.G. and Fellows, M.R.: *Parameterized Complexity*, Springer (1999).
6) Flum, J. and Grohe, M.: *Parameterized Complexity*, Springer (2006).
7) Halldórsson, M.M. and Tanaka, K.: Approximation and special cases of common subtrees and editing distance, *Proc. 7th International Symposium on Algorithms and Computation*, LNCS, Vol.1178, pp.75–84 (1996).
8) Horesh, T., Mehr, R. and Unger, R.: Designing an A^* algorithm for calculating edit distance between rooted-unordered trees, *Journal of Computational Biology*, Vol.13, pp.1165–1176 (2006).
9) Kilpeläinen, P. and Mannila, H.: Ordered and unordered tree inclusion, *SIAM Journal on Computing*, Vol.24, pp.340–356 (1995).
10) Shasha, D., Wang, J.T-L., Zhang, K. and Shih, F.Y.: Exact and approximate algorithms for unordered tree matching, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol.24, pp.668–678 (1994).
11) Tai, K-C.: The tree-to-tree correction problem, *Journal of ACM*, Vol.26, pp.422–433 (1979).
12) Zhang, K., Statman, R. and Shasha, D.: On the editing distance between unordered labeled trees, *Information Processing Letters*, Vol.42, pp.133–139 (1992).
13) Zhang, K. and Jiang, T.: Some MAX SNP-hard results concerning unordered labeled trees, *Information Processing Letters*, Vol.49, pp.249–254 (1994).
14) Zhang, K.: A constrained edit distance between unordered labeled trees, *Algorithmica*, Vol.15 pp.205–222 (1996).