PERGAMON

Pattern Recognition 34 (2001) 127–137

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Finding similar consensus between trees: an algorithm and a distance hierarchy

Jason T.L. Wang[a,*,1], Kaizhong Zhang[b]

[a]Department of Computer and Information Science, New Jersey Institute of Technology, University Heights, Newark, NJ 07102, USA
[b]Department of Computer Science, The University of Western Ontario, London, Ontario, Canada N6A 5B7

Received 18 February 1999; received in revised form 15 September 1999; accepted 15 September 1999

Abstract

The problem of finding this similar consensus (also known as the largest approximately common substructures) of two trees arises in many pattern recognition applications. This paper presents a dynamic programming algorithm to solve the problem based on the distance measure originated from Tanaka and Tanaka. The algorithm runs as fast as the best-known algorithm for comparing two trees using Tanaka's distance measure when the allowed distance between the common substructures is a constant independent of the input trees. In addition, we establish a hierarchy among Tanaka's distance measure and three other edit-based distance measures published in the literature. © 2000 Pattern Recognition Society. Published by Elsevier Science Ltd. All rights reserved.

Keywords: Dynamic programming; Edit distance; Pattern matching; Trees; Computational biology

1. Introduction

Ordered labeled trees are trees in which each node has a label and the left-to-right order among siblings matters. Such trees have many applications in computational biology, image processing, pattern recognition, natural language processing and structured document management, including the representations of RNA secondary structures [1], classification trees [2], images [3], patterns [4], grammar parses [5], and structured documents [6]. There have been several edit-based distance measures proposed for comparing the ordered labeled trees. These include the edit distance [7,8], isolated-subtree distance [2,9], top-down distance [10–12] and alignment distance [13]. When comparing two trees, the first three distances are measured based on a weighted number of edit operations, under different constraints, used to transform one tree to the other whereas the last distance is measured by aligning the two trees. The edit operations include node insertions, node deletions and node relabelings (substitutions). Table 1 summarizes some of the applications where the distance measures are useful.

Motivated by the applications listed in Table 1, we study in this paper the similar consensus (SC) problem for trees based on one of the distance measures, namely, the isolated-subtree distance. Specifically, we are interested in finding the similar consensus, or the largest approximately common substructures, of two trees. We consider a substructure of a tree T to be a connected subgraph of T. Given two trees T_1, T_2 and an integer d, the SC problem is to find a substructure U_1 of T_1 and a substructure U_2 of T_2 such that U_1 is within distance d of U_2 and where there does not exist any other substructure V_1 of T_1 and V_2 of T_2 such that V_1 and V_2 satisfy the distance constraint and the sum of the sizes of V_1 and V_2 is greater than the sum of the sizes of U_1 and U_2.

Similar problems for strings have been investigated by several researchers [19–21]. In Ref. [22] we studied the

* Corresponding author. Tel.: 001-973-596-3396; fax: 001-973-596-5777.

E-mail address: jason@cis.njit.edu (J.T.L. Wang).

[1] Part of the work of this author was done while visiting Courant Institute of Mathematical Sciences, New York University.

Table 1
Four edit-based distance measures for comparing trees and their applications

Measure	Applications	Reason
Edit distance	Comparison of RNA secondary structures [1], images [3], patterns [4], dictionary definitions [5] and structured documents [6]	Insert, delete and substitute arise naturally in gene evolution and correspond to common edit operations on the tree-structured data.
Isolated-subtree distance	Comparison of classification [2], taxonomic [14], evolutionary [15], phylogenetic [16] and pedigree trees [17]	Disjoint subfamilies (subtrees) should be compared with disjoint subfamilies (subtrees) in these trees.
Top-down distance	Identifying syntactic differences between two programs [12]	The measure exploits the knowledge of grammars and respects the parent–child relationship in a parse tree.
Alignment distance	Merging UNIX files/directories and documents and detecting the differences between them [18]	Aligning two trees amounts to overlaying one tree over the other and finding their topological differences.

Table 2
Complexities of the SC algorithms for the four edit-based distance measures

Distance measure	Time complexity
Edit distance	$O(d^2 \times \|T_1\| \times \|T_2\| \times \min(H_1, L_1) \times \min(H_2, L_2))$
Alignment distance	$O(d^2 \times \|T_1\| \times \|T_2\| \times (G_1 + G_2)^2)$
Isolated-subtree distance	$O(d^2 \times \|T_1\| \times \|T_2\|)$
Top-down distance	$O(d^2 \times \|T_1\| \times \|T_2\|)$

SC problem for trees and proposed an algorithm to solve the problem based on the edit distance. More recently, we investigated the SC problem based on the top-down distance [11] and the alignment distance [23]. We present here a dynamic programming algorithm to solve the SC problem based on the isolated-subtree distance. Table 2 shows the time complexities of the four SC algorithms, where $|T_i|, i = 1, 2$, is the size (i.e., the number of nodes) of tree T_i, H_i is the height of T_i, L_i is the number of leaves in T_i and G_i is the maximum degree of T_i. When the allowed distance between the common substructures is a constant independent of the input trees, the asymptotic complexities of these algorithms are the same as for the best-known algorithms for comparing two trees using the corresponding distance measures, and the constants remain moderate.

In addition to solving the SC problem based on the isolated-subtree distance, we establish a hierarchy among the four edit-based distance measures, describing their relationships. We have implemented the SC algorithms into a graphics toolkit [24]. The toolkit has been used for natural language comparison in the Unisys Lab and for finding frequently occurring motifs in RNA secondary structures in the National Cancer Institute [25].

The rest of the paper is organized as follows. Section 2 reviews the four distance measures. Section 3 establishes the hierarchy among them. Section 4 formalizes the SC problem based on the isolated-subtree distance. Section 5 presents a dynamic programming algorithm to solve the problem and analyzes its complexity. Section 6 concludes the paper.

2. Four distance measures

2.1. Edit operations

Since all the four distance measures studied here are defined in terms of edit operations, i.e., *relabeling*, *deleting*, and *inserting* a node, we start by reviewing the operations and then defining each measure in turn. Relabeling node n means changing the label of n. Deleting a node n means making the children of n become the children of the parent of n and removing n. Insert is the inverse of delete. Inserting node n as the child of node n' makes n the parent of a consecutive subsequence of the current children of n'. Fig. 1 illustrates the edit operations.

Suppose each node label is a symbol chosen from an alphabet Σ. Let λ, a unique symbol not in Σ, denote the null symbol. We represent an edit operation as a pair $u \rightarrow v$, where each u is either λ or a label of a node in tree T_1 and v is either λ or a label of a node in tree T_2. We call $u \rightarrow v$ a relabeling operation if $u \neq \lambda$ and $v \neq \lambda$, a delete operation if $v = \lambda$, and an insert operation if $u = \lambda$. Let T_2 be the tree that results from the application of an edit operation $u \rightarrow v$ to tree T_1; this is written $T_1 \Rightarrow T_2$ via $u \rightarrow v$.

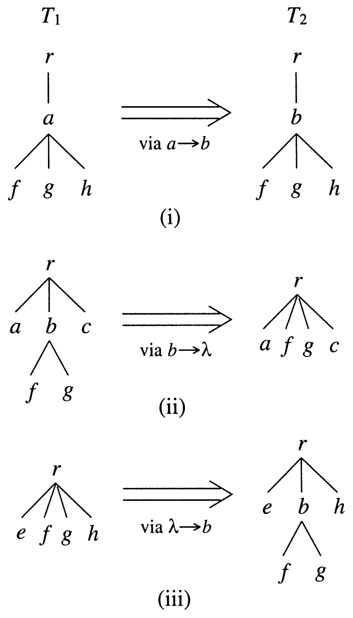

Fig. 1. (i) Relabeling: To change one node label (a) to another (b). (ii) Delete: To delete a node; all children of the deleted node (labeled *b*) become children of the parent (labeled *r*). (iii) Insert: To insert a node; a consecutive sequence of siblings among the children of the node labeled *r* (here, *f* and *g*) become the children of the node labeled *b*.

Let S be a sequence $s_1, s_2, \ldots, s_k$ of edit operations. S transforms tree T to tree T' if there is a sequence of trees $T_0, T_1, \ldots, T_k$ such that $T = T_0, T' = T_k$ and $T_{i-1} \Rightarrow T_i$ via s_i for $1 \leqslant i \leqslant k$. For the purpose of this work, we assume that the cost of all edit operations $u \to v$ is 1 if $u \neq v$ and 0 otherwise. By extension, the cost of the sequence S, denoted $\gamma(S)$, is simply the sum of the costs of the constituent edit operations in S.

2.2. Edit distance

The *edit distance* from tree T_1 to tree T_2, denoted $tdist_e(T_1, T_2)$, is the cost of a minimum cost sequence of edit operations transforming T_1 to T_2 [7,8]. This measure is best illustrated through the concept of *mappings*. A mapping is a graphical specification of which edit operations apply to each node in the two trees. For example, the mapping in Fig. 2 shows a way to transform T_1 to T_2. A dashed line from a node u in T_1 to a node v in T_2 indicates that u should be changed to v if $u \neq v$, or that u remains unchanged if $u = v$. The nodes of T_1 not touched by a dashed line are to be deleted and the nodes of T_2 not touched by a dashed line are to be inserted. Thus, the transformation in Fig. 2 includes deleting the two nodes labeled a and m in T_1 and inserting them into T_2.

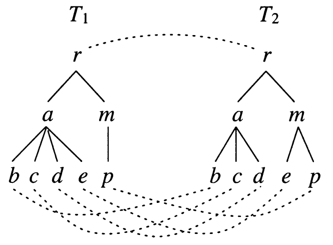

Fig. 2. A mapping from tree T_1 to tree T_2.

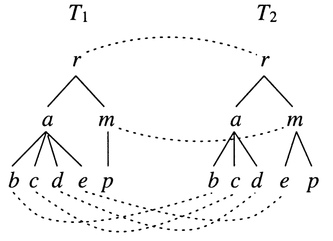

Fig. 3. $t_1[4]$ (labeled *e*) is mapped to $t_2[5]$ (labeled *e*) and $t_1[7]$ (labeled *m*) is mapped to $t_2[7]$ (labeled *m*). $t_2[7]$ is an ancestor of $t_2[5]$ whereas $t_1[7]$ is not an ancestor of $t_1[4]$, thus violating the ancestor order preservation condition.

We use a postorder numbering of the nodes in the trees. Let $t[i]$ represent the node of T whose position in the left-to-right postorder traversal of T is i. When there is no confusion, we also use $t[i]$ to represent the label of node $t[i]$. Formally, a mapping from tree T_1 to tree T_2 is a triple (M_e, T_1, T_2) (or simply M_e if the context is clear), where M_e is any set of ordered pairs of integers (i, j) satisfying:

(*Mapping conditions*)

1. $1 \leqslant i \leqslant |T_1|$ and $1 \leqslant j \leqslant |T_2|$;
2. 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 condition);
(b) $t_1[i_1]$ is to the left of $t_1[i_2]$ iff $t_2[j_1]$ is to the left of $t_2[j_2]$ (sibling order preservation condition);
(c) $t_1[i_1]$ is an ancestor of $t_1[i_2]$ iff $t_2[j_1]$ is an ancestor of $t_2[j_2]$ (ancestor order preservation condition).

Thus, the lines in Fig. 3 do not form a mapping since they violate the ancestor order preservation condition. The cost of M_e, denoted $\gamma(M_e)$, is the cost of deleting nodes of T_1 not touched by a mapping line plus the cost of inserting nodes of T_2 not touched by a mapping line plus the cost of relabeling nodes in those pairs related by mapping lines with different labels. It can be proved [8] that given S, a sequence of edit operations from T_1 to T_2, there exists a mapping M_e from T_1 to T_2 such that $\gamma(M_e) \leqslant \gamma(S)$, and conversely, for any mapping M_e, there

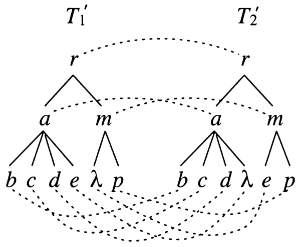

Fig. 4. An alignment between the two trees in Fig. 2.

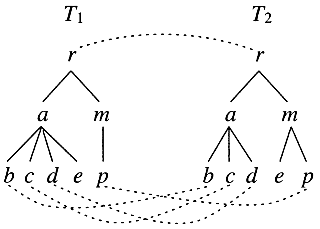

Fig. 5. An isolated-subtree mapping between the two trees in Fig. 2.

exists a sequence S of edit operations such that $\gamma(S) = \gamma(M_e)$. Thus, $tdist_e(T_1, T_2)$ also equals the cost of a minimum cost mapping from T_1 to T_2.

2.3. Alignment distance

Given two trees T_1 and T_2, an alignment of T_1 and T_2 is just a mapping from T_1 to T_2, obtained by first inserting a minimum number of nodes labeled with λ into T_1 and T_2 such that the two resulting trees T_1' and T_2' are topologically isomorphic, i.e., they are identical if the labels are ignored, and then drawing a mapping line from each node n_1 in T_1' to its corresponding node n_2 in T_2', i.e., aligning n_1 with n_2. (The number of λs one can add is the minimum necessary to form an isomorphism without aligning a λ in one tree with a λ in the other.) Thus, the mapping in Fig. 2 is not an alignment since no matter how we insert λs to make T_1 isomorphic to T_2, we are unable to align $e \in T_1$ with $e \in T_2$ and align $p \in T_1$ with $p \in T_2$ simultaneously. Fig. 4 shows an alignment between the two trees in Fig. 2.

The tree alignment defined here is a natural extension of string alignment [26,27]. The *alignment distance* from tree T_1 to tree T_2, denoted $tdist_a(T_1, T_2)$, is the cost of a minimum cost alignment between T_1 and T_2.

2.4. Isolated-subtree distance

This isolated-subtree distance from tree T_1 to tree T_2 is obtained by imposing on mappings the constraint that two disjoint subtrees of T_1 must be mapped to two disjoint subtrees of T_2. This distance measure was first proposed by Tanaka and Tanaka [2], who argued that such structure preserving mappings are more meaningful when comparing two classification trees. Formally, an *isolated-subtree mapping* from T_1 to T_2 is a triple (M_b, T_1, T_2) (or simply M_b if the context is clear), where M_b is any set of ordered pairs of integers (i, j) satisfying the mapping conditions in Section 2.2 as well as the following constraint. For any triple $(i_1, j_1), (i_2, j_2)$ and (i_3, j_3) in M_b, let $t_1[h_1]$ represent $lca(\{t_1[i_1], t_1[i_2]\})$ and let $t_2[h_2]$ represent $lca(\{t_2[j_1], t_2[j_2]\})$, where $lca(.)$ denotes the least common ancestor of the nodes in the indicated set. Suppose $(h_1, h_2) \notin M_b$. Then $t_1[h_1]$ is a proper ancestor of $t_1[i_3]$ iff $t_2[h_2]$ is a proper ancestor of $t_2[j_3]$. Thus, the mapping M_e in Fig. 2 is not an isolated-subtree mapping, since $(1,1), (2,2), (4,5) \in M_e$, $lca(\{t_1[1], t_1[2]\}) = t_1[5]$, $lca(\{t_2[1], t_2[2]\}) = t_2[4]$, $(5,4) \notin M_e$, $t_1[5]$ is a proper ancestor of $t_1[4]$ whereas $t_2[4]$ is not a proper ancestor of $t_2[5]$. Fig. 5 shows an isolated-subtree mapping between the two trees in Fig. 2.

The cost of an isolated-subtree mapping M_b is well defined since M_b is also a mapping. The *isolated-subtree distance* from tree T_1 to tree T_2, denoted $tdist_b(T_1, T_2)$, is the cost of a minimum cost isolated-subtree mapping from T_1 to T_2.

2.5. Top-down distance

The top-down distance between two trees was originated from Selkow [10]. Yang [12] represented a program by a parse tree and found the largest common substructure of two parse trees based on the notion of the top-down distance. Formally, a *top-down mapping* from tree T_1 to tree T_2 is a triple (M_ω, T_1, T_2) (or simply M_ω is the context is clear), where M_ω is any set of ordered pairs of integers (i, j) satisfying the mapping conditions in Section 2.2 as well as the following constraint. Let $par(i)$ represent the postorder number of the parent of $t[i]$. If (i, j) is in M_ω, then $(par(i), par(j))$ is also in M_ω. Thus, the mapping in Fig. 2 is not a top-down mapping, since $b \in T_1$ is mapped to $b \in T_2$ whereas their parents are not in the mapping. Fig. 6 shows a top-down mapping between the two trees in Fig. 2. Note that the definition of a top-down mapping M_ω implies that $(|T_1|, |T_2|)$ must be in M_ω, i.e., the root of T_1 must be mapped to the root of T_2. Furthermore, if a node n is to be deleted (inserted, respectively) in M_ω, then the subtree rooted at n, if any, is to be deleted (inserted, respectively).

The cost of a top-down mapping M_ω is well defined since M_ω is also mapping. The *top-down distance* from tree T_1 to tree T_2, denoted $tdist_\omega(T_1, T_2)$, is the cost of a minimum cost top-down mapping from T_1 to T_2. Intutively, this distance measure gives more weight to upper level nodes than to lower level nodes, i.e., the upper level nodes are more important than lower level ones. This is reasonable when comparing two parse trees.

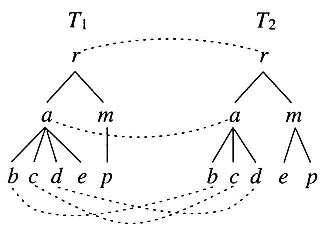

Fig. 6. A top-down mapping between the two trees in Fig. 2.

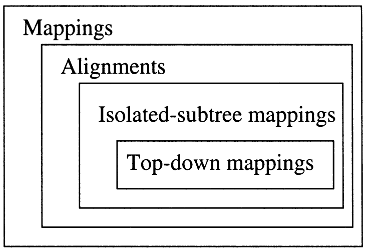

Fig. 7. Containment relationships among the four types of mappings.

3. The distance hierarchy

The four types of mappings described in the previous section yield a hierarchy as illustrated in Fig. 7. Mappings closer to the center in the hierarchy are more restricted. The following lemmas establish the hierarchy.

Lemma 3.1. *Let T_1 and T_2 be two trees and let M_b be an isolated-subtree mapping from T_1 to T_2. Then M_b is also an alignment between T_1 and T_2.*

Proof. Let $F_i, i = 1, 2$, be the forest resulted from deleting the root of T_i. Let $Height(T_i)$ ($Height(F_i)$, respectively), be the height of T_i (F_i, respectively). The proof is by induction on $Height(T_1) + Height(T_2)$ and $Height(F_1) + Height(F_2)$. Without loss of generality, we consider only non-empty trees and forests.

Basis step: $Height(T_i) \leqslant 2$ and $Height(F_i) \leqslant 2$ for $i = 1, 2$. This is trivial for the tree case. For two forests F_1 and F_2, since disjoint trees in F_1 are mapped to disjoint trees in F_2, one can treat each tree as a single node and each forest as a sequence of nodes. Thus M_b is isomorphic to a string mapping, and that mapping corresponds to an alignment between the two resulting strings.

Induction step for trees: $Height(T_i) = k$ or $k - 1$ where $k > 2, i = 1, 2$, and $2k - 1 \leqslant Height(T_1) + Height(T_2) \leqslant 2k$. The inductive hypothesis is that for any two trees T and T', $Height(T) + Height(T') < Height(T_1) + Height(T_2)$, if M is an isolated-subtree mapping from T to T', then M is an alignment between T and T'. Moreover, for any two forests F and F', $Height(F) + Height(F') < 2k - 1$, if M is an isolated-subtree mapping from F to F', then M is also an alignment between F and F'.

There are three cases to examine: in M_b, (i) the root of T_1 is mapped to a descendant n of the root of T_2; (ii) the root of T_2 is mapped to a descendant of the root of T_1; (iii) the root of T_1 is mapped to the root of T_2. For (i), let T represent the subtree rooted at n. Since $Height(T_1) + Height(T) < Height(T_1) + Height(T_2)$, T_1 can be aligned with T, by the inductive hypothesis. Therefore M_b results in an alignment between T_1 and T_2. (ii) is symmetric to (i). For (iii), $Height(F_1) + Height(F_2) < 2k - 1$, by the inductive hypothesis, F_1 can be aligned with F_2, and hence the result follows.

Induction step for forests: $Height(F_i) = k$ or $k - 1$ where $k > 2, i = 1, 2$, and $2k - 1 \leqslant Height(F_1) + Height(F_2) \leqslant 2k$. The inductive hypothesis is that for any two trees T and T', $2k - 1 \leqslant Height(T) + Height(T') \leqslant 2k$, if M is an isolated-subtree mapping from T to T', then M is an alignment between T and T'. Moreover, for any two forests F and F', $Height(F) + Height(F') < Height(F_1) + Height(F_2)$, if M is an isolated-subtree mapping from F to F', then M is also an alignment between F and F'.

There are four cases to examine: in M_b, (i) exactly one tree in each forest is touched by mapping lines; (ii) more than one tree in F_1 but only one tree T in F_2 are touched by mapping lines; (iii) more than one tree in F_2 but only one tree in F_1 are touched by mapping lines; (iv) more than one tree in F_1 and more than one tree in F_2 are touched by mapping lines. For (i), by the inductive hypothesis, the two trees can be aligned. Therefore, M_b results in an alignment between F_1 and F_2. For (ii), the root of T cannot be touched by any line in M_b. Let F represent the trees in F_1 touched by mapping lines. Let F' represent the forest resulted from deleting the root of T. By the inductive hypothesis, F can be aligned with F'. (iii) is symmetric to (ii). For (iv), from the definition of M_b, two disjoint trees in F_1 must be mapped to two disjoint trees in F_2. Consider each tree T in F_1 and the corresponding tree T' in F_2 that T is mapped to. By the inductive hypothesis, T can be aligned with T'. Therefore M_b results in an alignment between F_1 and F_2. □

Lemma 3.2. *Let T_1 and T_2 be two trees and let M_ω be a top-down mapping from T_1 to T_2. Then M_ω is also an isolated-subtree mapping from T_1 to T_2.*

Proof. By definition, if $(i, j) \in M_\omega$, then $(par(i), par(j)) \in M_\omega$. Thus if (i_1, j_1) and (i_2, j_2) are in M_ω, then $(lca(\{i_1, i_2\}), lca(\{j_1, j_2\})) \in M_\omega$, which implies that M_ω must be an isolated-subtree mapping. □

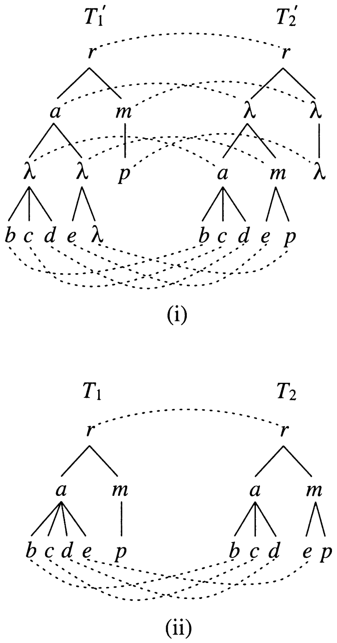

Fig. 8. (i) An alignment between the two trees in Fig. 2; equivalently, the alignment can be represented as the mapping in (ii).

Note that the containment relationships among the four types of mappings are strict. For example, the mapping in Fig. 2 is not an alignment. Fig. 8 shows an alignment between the same pair of trees that is not an isolated-subtree mapping. Fig. 5 shows an isolated-subtree mapping that is not a top-down mapping. Thus, for any two trees T_1 and T_2, we have $tdist_e(T_1, T_2) \leqslant tdist_a(T_1, T_2) \leqslant tdist_b(T_1, T_2) \leqslant tdist_\omega(T_1, T_2)$. Comparing with Table 2, the complexities of the SC algorithms coincide with the distance hierarchy, i.e., the more restricted a distance measure, the faster the corresponding algorithm.

4. The SC problem for trees

4.1. Cut operations

We now formalize the notion of the similar consensus (SC), or the largest approximately common substructures, of two trees based on the isolated-subtree distance. We define a substructure U of tree T to be a connected subgraph of T. That is, U is rooted at a node n in T and is generated by cutting off some subtrees in the subtree rooted at n. Formally, let $T[i]$ represent the subtree rooted at $t[i]$. The operation of cutting at node $t[i]$ means removing $T[i]$ (see Fig. 9). A set S of nodes of $T[k]$ is said to be a set of consistent subtree cuts in $T[k]$ if (i) $t[i] \in S$ implies that $t[i]$ is a node in $T[k]$, and (ii)

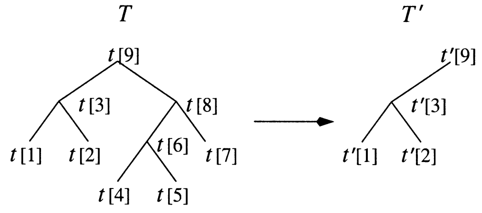

Fig. 9. Cutting at node $t[8]$ to get T' from T.

$t[i], t[j] \in S$ implies that neither is an ancestor of the other in $T[k]$. Intuitively, S is the set of all roots of the removed subtrees in $T[k]$.

We use $Cut(T, S)$ to represent the tree T with subtree removals at all nodes in S. Let $Subtrees(T)$ be the set of all possible sets of consistent subtree cuts in T. Given two trees T_1 and T_2 and an integer d, the size of the similar root-containing consensus within distance $k, 0 \leqslant k \leqslant d$, of $T_1[i]$ and $T_2[j]$, denoted $tsize_b(T_1[i], T_2[j], k)$ (or simply $tsize_b(i, j, k)$ when the context is clear), is max $\{|Cut(T_1[i], S_1)| + |Cut(T_2[j], S_2)|\}$ subject to $tdist_b(Cut(T_1[i], S_1), Cut(T_2[j], S_2)) \leqslant k$, $S_1 \in Subtrees(T_1[i])$, $S_2 \in Subtrees(T_2[j])$. Finding the similar consensus (SC), within distance d, of $T_1[i]$ and $T_2[j]$ amounts to calculating $\max_{1 \leqslant u \leqslant i, 1 \leqslant v \leqslant j}\{tsize_b(T_1[u], T_2[v], d)\}$ and locating the $Cut(T_1[u], S_u)$ and $Cut(T_2[v], S_v)$, $S_u \in Subtrees(T_1[u])$, $S_v \in Subtrees(T_2[v])$ that achieve the maximum size. The size of SC, within distance d, of T_1 and T_2 is $\max_{1 \leqslant i \leqslant |T_1|, 1 \leqslant j \leqslant |T_2|}\{tsize_b(T_1[i], T_2[j], d)\}$. We shall focus on computing the maximum size. By memorizing the size information during the computation and by a simple backtracking technique, one can find both the maximum size and one of the corresponding substructure pairs yielding the size in the same time complexity.

4.2. Notation

We use $F[i]$ to represent the ordered forest obtained by deleting $t[i]$ from $T[i]$ (see Fig. 10). $|F|$ is the size of forest F. A set S of nodes of F is said to be a set of consistent subtree cuts in F if (i) $t[p] \in S$ implies that $t[p]$ is a node in F, and (ii) $t[p], t[q] \in S$ implies that neither is an ancestor of the other in F. We use $Cut(F, S)$ to represent the subforest F with subtree removals at all nodes in S. For example, consider Fig. 10 again. $S = \{t[2], t[5], t[7]\}$ is a set of consistent subtree cuts in $F[9]$. $Cut(F[9], S)$ is obtained by cutting at nodes $t[2]$, $t[5]$ and $t[7]$ from $F[9]$.

Let $Subtrees(F)$ be the set of all possible sets of consistent subtree cuts in F. Let $fdist_b(F_1, F_2)$ be the isolated-subtree distance from forest F_1 to forest F_2. We define the size of the similar root-containing consensus, within distance k, of F_1 and F_2, denoted $fsize_b(F_1, F_2, k)$, as $\max\{|Cut(F_1, S_1)| + |Cut(F_2, S_2)|\}$ subject to

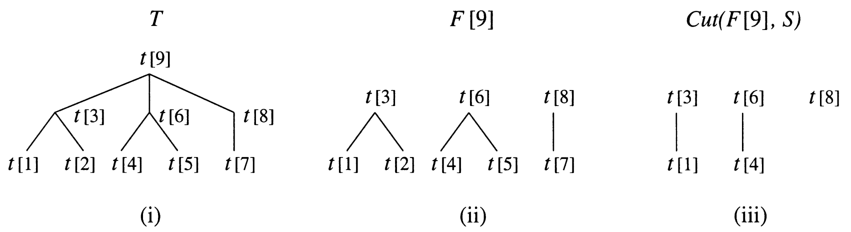

Fig. 10. (i) A tree T. (ii) Deleting node $t[9]$ to get $F[9]$ from T. (iii) Cutting at nodes in $S = \{t[2], t[5], t[7]\}$ from $F[9]$.

$fdist_b(Cut(F_1, S_1), Cut(F_2, S_2)) \leqslant k$, $S_1 \in Subtrees(F_1)$, $S_2 \in Subtrees(F_2)$.

5. The SC algorithm for trees

5.1. Basic properties

Let $t_1[i]$ be a node in tree T_1 and let $t_2[j]$ be a node in tree T_2. We use $t_1[i_1], t_1[i_2], \ldots, t_1[i_{m_i}]$ to represent the children of $t_1[i]$ and use $t_2[j_1], t_2[j_2], \ldots, t_2[j_{n_j}]$ to represent the children of $t_2[j]$. $F_1[i_s, i_t]$, $1 \leqslant s \leqslant t \leqslant m_i$, represents the forest consisting of the subtrees $T_1[i_s], \ldots, T_1[i_t]$. $F_1[i_1, i_{m_i}] = F_1[i]$; $F_1[i_p, i_{p-1}] = \emptyset$ for all $1 \leqslant p \leqslant m_i$. Note $F_1[i_p, i_p] = T_1[i_p] \neq F_1[i]$. $F_1[i : i_p]$ represents the forest $F_1[i]$ with the subtree $T_1[i_p]$ being removed. Lemmas 5.1 and 5.2 below summarize the formulas used for initializing $tsize_b$ and $fsize_b$ for all k, $0 \leqslant k \leqslant d$.

Lemma 5.1. *For all i, j, $1 \leqslant i \leqslant |T_1|$ and $1 \leqslant j \leqslant |T_2|$, and for any p, q, s, t, $1 \leqslant p \leqslant s \leqslant m_i$ and $1 \leqslant q \leqslant t \leqslant n_j$,*

(i) $tsize_b(\emptyset, \emptyset, 0) = 0$;
(ii) $fsize_b(\emptyset, \emptyset, 0) = 0$;
(iii) $tsize_b(T_1[i], \emptyset, 0) = 0$;
(iv) $tsize_b(\emptyset, T_2[j], 0) = 0$;
(v) $fsize_b(F_1[i_p, i_s], \emptyset, 0) = 0$;
(vi) $fsize_b(\emptyset, F_2[j_q, j_t], 0) = 0$;
(vii) $fsize_b(F_1[i : i_s], \emptyset, 0) = 0$;
(viii) $fsize_b(\emptyset, F_2[j : j_t], 0) = 0$.

Proof. Immediate from definitions. □

Lemma 5.2. *For all i, j, $1 \leqslant i \leqslant |T_1|$ and $1 \leqslant j \leqslant |T_2|$, and for any p, q, s, t, $1 \leqslant p \leqslant s \leqslant m_i$ and $1 \leqslant q \leqslant t \leqslant n_j$, and for any k, $1 \leqslant k \leqslant d$,*

(i) $tsize_b(\emptyset, \emptyset, k) = 0$;
(ii) $fsize_b(\emptyset, \emptyset, k) = 0$;
(iii) $tsize_b(T_1[i], \emptyset, k) = fsize_b(F_1[i], \emptyset, k-1) + 1$;
(iv) $tsize_b(\emptyset, T_2[j], k) = fsize_b(\emptyset, F_2[j], k-1) + 1$;
(v) $fsize_b(F_1[i_p, i_s], \emptyset, k) = \mathbf{min}\{|F_1[i_p, i_s]|, k\}$;
(vi) $fsize_b(\emptyset, F_2[j_q, j_t], k) = \mathbf{min}\{|F_2[j_q, j_t]|, k\}$;
(vii) $fsize_b(F_1[i : i_s], \emptyset, k) = \mathbf{min}\{|F_1[i : i_s]|, k\}$;
(viii) $fsize_b(\emptyset, F_2[j : j_t], k) = \mathbf{min}\{|F_2[j : j_t]|, k\}$.

Proof. (i) and (ii) are obvious since the similar root-containing consensus of two empty trees or forests is empty. For (iii), the similar root-containing consensus of $F_1[i]$ and $\emptyset$ plus $t_1[i]$ is the similar root-containing consensus of $T_1[i]$ and $\emptyset$. This means that $tsize_b(T_1[i], \emptyset, k) = fsize_b(F_1[i], \emptyset, k-1) + 1$ where the 1 is obtained by including the node $t_1[i]$. (iv) is proved similarly as for (iii).

For (v), if $k \leqslant |F_1[i_p, i_s]|$, since the allowed distance is at most k, we have to remove some subtrees from $F_1[i_p, i_s]$ until only k nodes are left in the forest. If $k > |F_1[i_p, i_s]|$, then all the nodes in $F_1[i_p, i_s]$ constitute the similar root-containing consensus of $F_1[i_p, i_s]$ and $\emptyset$. Thus the result follows. (vi)–(viii) are proved similarly as for (v). □

Lemma 5.3. *For all i, j, $1 \leqslant i \leqslant |T_1|$ and $1 \leqslant j \leqslant |T_2|$, and for any s, t, $1 \leqslant s \leqslant m_i$ and $1 \leqslant t \leqslant n_j$,*

$$fsize_b(F_1[i_1, i_s], F_2[j_1, j_t], 0) = \max \begin{cases} fsize_b(F_1[i_1, i_{s-1}], F_2[j_1, j_t], 0), \\ fsize_b(F_1[i_1, i_s], F_2[j_1, j_{t-1}], 0), \\ fsize_b(F_1[i_1, i_{s-1}], F_2[j_1, j_{t-1}], 0) \\ \quad + tsize_b(i_s, j_t, 0). \end{cases}$$

Proof. Suppose $S_1 \in Subtrees(F_1[i_1, i_s])$ and $S_2 \in Subtrees(F_2[j_1, j_t])$ are two smallest sets of consistent subtree cuts that maximize $|Cut(F_1[i_1, i_s], S_1)| + |Cut(F_2[j_1, j_t], S_2)|$ where $fdist_b(Cut(F_1[i_1, i_s], S_1), Cut(F_2[j_1, j_t], S_2)) = 0$. Then at least one of the following cases must hold: (i) $t_1[i_s] \in S_1$; (ii) $t_2[j_t] \in S_2$; (iii) $t_1[i_s] \notin S_1$ and $t_2[j_t] \notin S_2$.

For (i), $fsize_b(F_1[i_1, i_s], F_2[j_1, j_t], 0) = fsize_b(F_1[i_1, i_{s-1}], F_2[j_1, j_t], 0)$. For (ii), $fsize_b(F_1[i_1, i_s], F_2[j_1, j_t], 0) = fsize_b(F_1[i_1, i_s], F_2[j_1, j_{t-1}], 0)$. For (iii), let M_b be the isolated-subtree mapping (with cost 0) from $Cut(F_1[i_1, i_s], S_1)$ to $Cut(F_2[j_1, j_t], S_2)$. In M_b, $T_1[i_s]$

must be mapped to $T_2[j_t]$ because otherwise we cannot have distance zero between $Cut(F_1[i_1, i_s], S_1)$ and $Cut(F_2[j_1, j_t], S_2)$. Therefore $fsize_b(F_1[i_1, i_s], F_2[j_1, j_t], 0) = fsize_b(F_1[i_1, i_{s-1}], F_2[j_1, j_{t-1}], 0) + tsize_b(i_s, j_t, 0)$. Thus, $fsize_b(F_1[i_1, i_s], F_2[j_1, j_t], 0)$ is the maximum of these three cases. □

Lemma 5.4. *For all i, j,* $1 \leqslant i \leqslant |T_1|$ and $1 \leqslant j \leqslant |T_2|$,

$$tsize_b(i, j, 0) = \begin{cases} fsize_b(F_1[i], F_2[j], 0) + 2 & \text{if } t_1[i] = t_2[j], \\ 0 & \text{otherwise.} \end{cases}$$

Proof. First, consider the case where $t_1[i] = t_2[j]$. Suppose $S_1 \in Subtrees(T_1[i])$ and $S_2 \in Subtrees(T_2[j])$ are two smallest sets of consistent subtree cuts that maximize $|Cut(T_1[i], S_1)| + |Cut(T_2[j], S_2)|$ where $tdist_b(Cut(T_1[i], S_1), Cut(T_2[j], S_2)) = 0$. Let M_b be the isolated-subtree mapping (with cost 0) from $Cut(T_1[i], S_1)$ to $Cut(T_2[j], S_2)$. Clearly, in M_b, $t_1[i]$ must be mapped to $t_2[j]$. Furthermore, the similar root-containing consensus of $F_1[i]$ and $F_2[j]$ plus $t_1[i]$ (or $t_2[j]$) must be the similar root-containing consensus of $T_1[i]$ and $T_2[j]$. This means that $tsize_b(i, j, 0) = fsize_b(F_1[i], F_2[j], 0) + 2$, where 2 is obtained by including the two nodes $t_1[i]$ and $t_2[j]$.

Next, consider the case where $t_1[i] \neq t_2[j]$. To get distance zero between the two trees after applying cut operations to them, we have to remove both trees entirely. Thus, $tsize_b(i, j, 0) = 0$. □

In computing $fsize_b(F_1[i], F_2[j], k)$ and $tsize_b(i, j, k)$ for $k > 0$, we need some definitions before showing the lemmas. Define a *restricted mapping* $RM(i_s, j_t)$ from forest $F_1[i_1, i_s]$ to forest $F_2[j_1, j_t]$ to be an isolated-subtree mapping between the two forests satisfying the following condition. If (x, y) is in $RM(i_s, j_t)$ and $t_1[x]$ is in $T_1[i_p]$, $1 \leqslant p \leqslant s$, and $t_2[y]$ is in $T_2[j_q]$, $1 \leqslant q \leqslant t$, then for any (x_1, y_1) in $RM(i_s, j_t)$, $t_1[x_1]$ is in $T_1[i_p]$ iff $t_2[y_1]$ is in $T_2[j_q]$. Thus, in the restricted mapping, suppose $T_1[i_p]$ is mapped to $T_2[j_q]$ and $T_1[i_x]$ is mapped to $T_2[j_y]$. Then $i_p = i_x$ iff $j_q = j_y$.

Now, let $I = (i_1, i_2, \ldots, i_s)$ and $J = (j_1, j_2, \ldots, j_t)$ be two ordered lists of integers. Let $PF(i_s, j_t)$ denote a partial function between I and J, i.e., $PF(i_s, j_t)$ is a set of ordered pairs of integers (p, q) satisfying: (i) $1 \leqslant p \leqslant s$ and $1 \leqslant q \leqslant t$; and (ii) for any (p, q) and (x, y) in $PF(i_s, j_t)$, $p = x$ iff $q = y$ and $p < x$ iff $q < y$. Clearly, each restricted mapping $RM(i_s, j_t)$ corresponds to a partial function $PF(i_s, j_t)$ where for $(x, y) \in PF(i_s, j_t)$, $T_1[i_x]$ is mapped to $T_2[j_y]$ as illustrated in Fig. 11. Define $Z(i_s, j_t, k)$ with respect to a restricted mapping $RM(i_s, j_t)$ to be $\max_{h_{(u,v)}}\{\Sigma_{(u,v) \in PF(i_s, j_t)} tsize_b(i_u, j_v, h_{(u,v)})\}$ where $\Sigma_{(u,v) \in PF(i_s, j_t)} h_{(u,v)} = k$ and $PF(i_s, j_t)$ is the mapping's corresponding partial function. Intuitively, $Z(i_s, j_t, k)$ is the maximum possible size achieved by the restricted mapping.

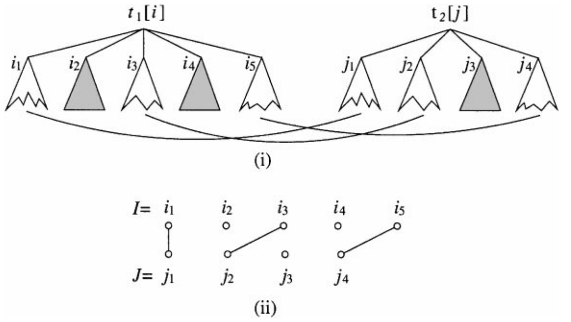

Fig. 11. (i) A restricted mapping $RM(i_5, j_4)$ from $F_1[i_1, i_5]$ to $F_2[j_1, j_4]$ (removed subtrees are shaded); (ii) the corresponding partial function $PF(i_5, j_4) = \{(1, 1), (3, 2), (5, 4)\}$.

Lemma 5.5. *For all i, j,* $1 \leqslant i \leqslant |T_1|$ *and* $1 \leqslant j \leqslant |T_2|$, *and for any k,* $1 \leqslant k \leqslant d$,

$$fsize_b(F_1[i], F_2[j], k) = \mathbf{max}\begin{cases} \max_{i_p}\max_{0 \leqslant h \leqslant k-1}\{fsize_b(F_1[i : i_p], \emptyset, h) \\ \quad + fsize_b(F_1[i_p], F_2[j], k-1-h) \mid 1 \leqslant p \leqslant m_i\} + 1, \\ \max_{j_q}\max_{0 \leqslant h \leqslant k-1}\{fsize_b(\emptyset, F_2[j : j_q], h) \\ \quad + fsize_b(F_1[i], F_2[j_q], k-1-h) \mid 1 \leqslant q \leqslant n_j\} + 1, \\ \max_{RM(i_{m_i},\, j_{n_j})}\{Z(i_m, j_{n_j}, k)\}. \end{cases}$$

Proof. Suppose $S_1 \in Subtrees(F_1[i])$ and $S_2 \in Subtrees(F_2[j])$ are two smallest sets of consistent subtree cuts that maximize $|Cut(F_1[i], S_1)| + |Cut(F_2[j], S_2)|$ where $fdist_b(Cut(F_1[i], S_1), Cut(F_2[j], S_2)) \leqslant k$. Let M_b be a minimum cost isolated-subtree mapping from $Cut(F_1[i], S_1)$ to $Cut(F_2[j], S_2)$. There are four cases to examine:

Case 1: There exists a p, $1 \leqslant p \leqslant m_i$, such that for every (x, y) in M_b, $t_1[x]$ is a node in subtree $T_1[i_p]$. Moreover, there are (x_1, y_1) and (x_2, y_2) in M_b such that $t_2[y_1]$ is a node in $T_2[j_{q1}]$ and $t_2[y_2]$ is a node in $T_2[j_{q2}]$, where $1 \leqslant q_1 \neq q_2 \leqslant n_j$. Note that by the isolated-subtree mapping conditions, $t_1[i_p]$ won't be touched by any line in M_b as illustrated in Fig. 12. Thus, the forest $F_1[i : i_p]$ must be mapped to an empty forest. We may remove nodes from $F_1[i : i_p]$ until only h nodes are left where $0 \leqslant h \leqslant k-1$. Therefore, $fsize_b(F_1[i], F_2[j], k) = \max_{0 \leqslant h \leqslant k-1}\{fsize_b(F_1[i : i_p], \emptyset, h) + fsize_b(F_1[i_p], F_2[j], k-1-h)\} + 1$, where the 1 is obtained by including $t_1[i_p]$. The value of p ranges from 1 to m_i, and thus we take the maximum of the corresponding sizes.

Case 2: There exists a q, $1 \leqslant q \leqslant n_j$, such that for every (x, y) in M_b, $t_2[y]$ is a node in subtree $T_2[j_q]$. Moreover,

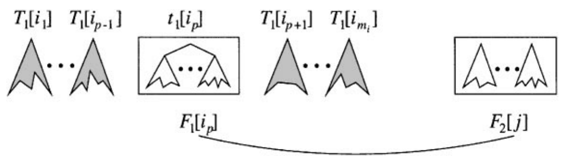

Fig. 12. Illustration of Case 1 of Lemma 5.5; shaded subtrees constitute $F_1[i:i_p]$.

there are (x_1, y_1) and (x_2, y_2) in M_b such that $t_1[x_1]$ is a node in $T_1[i_{p1}]$ and $t_1[x_2]$ is a node in $T_1[i_{p2}]$, where $1 \leqslant p_1 \neq p_2 \leqslant m_i$. The proof is analogous to that for Case 1.

Case 3: There exist p and q, $1 \leqslant p \leqslant m_i$, $1 \leqslant q \leqslant n_j$, such that if (x, y) is in M_b, then $t_1[x]$ is a node in $T_1[i_p]$ and $t_2[y]$ is a node in $T_2[j_q]$. Thus, M_b is a restricted mapping. So, $fsize_b(F_1[i], F_2[j], k) = \max_{RM(i_{m_i},\, j_{n_j})}\{Z(i_{m_i}, j_{n_j}, k)\}$.

Case 4: There exist $(f_1, g_1), (f_2, g_2), (x_1, y_1), (x_2, y_2)$ in M_b such that $t_1[f_1]$ is a node in $T_1[i_{p1}]$, $t_1[f_2]$ is a node in $T_1[i_{p2}]$, $t_2[y_1]$ is a node in $T_2[j_{q1}]$, $t_2[y_2]$ is a node in $T_2[j_{q2}]$, where $1 \leqslant p_1 \neq p_2 \leqslant m_i$ and $1 \leqslant q_1 \neq q_2 \leqslant n_j$. Clearly, M_b is a restricted mapping from $F_1[i]$ to $F_2[j]$. Thus, Case 4 results in the same formula as in Case 3. □

Lemma 5.6. *For all i, j, $1 \leqslant i \leqslant |T_1|$ and $1 \leqslant j \leqslant |T_2|$, and for any k, $1 \leqslant k \leqslant d$,*

$$tsize_b(i, j, k)$$
$$= \mathbf{max}\begin{cases} \max_{i_p} \max_{0 \leqslant h \leqslant k-1}\{fsize_b(F_1[i:i_p], \emptyset, h) \\ \quad + tsize_b(i_p, j, k-1-h) | 1 \leqslant p \leqslant m_i\} + 1, \\ \max_{j_q} \max_{0 \leqslant h \leqslant k-1}\{fsize_b(\emptyset, F_2[j:j_q], h) \\ \quad + tsize_b(i, j_q, k-1-h) | 1 \leqslant q \leqslant n_j\} + 1, \\ fsize_b(F_1[i], F_2[j], k-c) + 2, \end{cases}$$

where

$$c = \begin{cases} 0 & \text{if } t_1[i] = t_2[j], \\ 1 & \text{otherwise.} \end{cases}$$

Proof. Clearly, in order to get the maximum size, neither $T_1[i]$ nor $T_2[j]$ can be removed. Now suppose $S_1 \in Subtrees(T_1[i])$ and $S_2 \in Subtrees(T_2[j])$ are two smallest sets of consistent subtree cuts that maximize $|Cut(T_1[i], S_1)| + |Cut(T_2[j], S_2)|$ where $tdist_b(Cut(T_1[i], S_1),\ Cut(T_2[j], S_2)) \leqslant k$. Let M_b be a minimum cost isolated-subtree mapping from $Cut(T_1[i], S_1)$ to $Cut(T_2[j], S_2)$. Then at least one of the following cases must hold: (i) $t_1[i]$ is not touched by a line in M_b whereas $t_2[j]$ is touched by a line in M_b; (ii) $t_2[j]$ is not touched by a line in M_b whereas $t_1[i]$ is touched by a line in M_b; (iii) both $t_1[i]$ and $t_2[j]$ are touched by lines in M_b; (iv) neither $t_1[i]$ nor $t_2[j]$ is touched by a line in M_b.

For (i), let (u, j) be in M_b for some integer u. Since $t_1[i]$ is not touched by a line in M_b, $t_1[u]$ must be a node in $F_1[i]$. Let $t_1[i_p]$ be the child of $t_1[i]$ on the path from $t_1[u]$ to $t_1[i]$. Then the forest $F_1[i:i_p]$ must be mapped to an empty forest. We may remove nodes from $F_1[i:i_p]$ until only h nodes are left, where $0 \leqslant h \leqslant k-1$. Therefore $tsize_b(i, j, k) = \max_{0 \leqslant h \leqslant k-1}\{fsize_b(F_1[i:i_p], \emptyset, h) + tsize_b(i_p, j, k-1-h)\} + 1$, where the 1 is obtained by including the node $t_1[i]$. The value of p ranges from 1 to m_i, and hence we take the maximum of the corresponding sizes. (ii) is symmetric to (i). For (iii), by the isolated-subtree mapping conditions, (i, j) must be in M_b. Thus, $tsize_b(i, j, k) = fsize_b(F_1[i], F_2[j], k) + 2$ if $t_1[i] = t_2[j]$ and $tsize_b(i, j, k) = fsize_b(F_1[i], F_2[j], k-1) + 2$ otherwise. (iv) is covered by (iii). □

5.2. The algorithm

From the definition of $RM(i_s, j_t)$ and $PF(i_s, j_t)$ it's easy to see that one can use a dynamic programming algorithm similar to that for approximate string matching [28] to compute $\max_{RM(i_{m_i},\, j_{n_j})}\{Z(i_{m_i}, j_{n_j}, k)\}$. (Procedure Find-Consensus-1 in Fig. 13 details its calculation.) The algorithm for computing $tsize_b(i, j, k)$ for $0 \leqslant k \leqslant d$ is summarized in Fig. 13.

Note that $tsize_b(i, j, k)$ represents the size of the similar root-containing consensus, within distance k, of $T_1[i]$ and $T_2[j]$. To calculate the size of the similar consensus, within distance d, of $T_1[i]$ and $T_2[j]$, we build, in a bottom-up fashion, another array $\beta(i, j, d)$, $1 \leqslant i \leqslant |T_1|$, $1 \leqslant j \leqslant |T_2|$, using $tsize_b(i, j, d)$ as follows. Let $L = \max_{1 \leqslant u \leqslant m_i}\beta(i_u, j, d)$ where $i_1, \ldots i_{m_i}$ are the postorder numbers of the childred of $t_1[i]$ or $L = 0$ if $t_1[i]$ is a leaf. Let $R = \max_{1 \leqslant v \leqslant n_j}\beta(i, j_v, d)$ where $j_1 \ldots j_{n_j}$ are the postorder numbers of the children of $t_2[j]$ or $R = 0$ is $t_2[j]$ is a leaf. Calculate $\beta(i, j, d) = \max\{tsize_b(i, j, d), L, R\}$. The size of the similar consensus, within distance d, of $T_1[i]$ and $T_2[j]$ is $\beta(i, j, d)$. The size of the similar consensus, within distance d, of T_1 and T_2 is $\beta(|T_1|, |T_2|, d)$.

Time complexity. For an input i, j, k, the complexity of Procedure Find-Consensus-1 is $O(k \times m_i \times n_j)$. By Lemma 5.5, given $\max_{RM(i_{m_i},\, j_{n_j})}\{Z(i_{m_i}, j_{n_j}, k)\}$, computing $fsize_b(F_1[i], F_2[j], k)$ takes $O(k \times (m_i + n_j))$ time. By Lemma 5.6, computing $tsize_b(i, j, k)$ also takes $O(k \times (m_i + n_j))$ time. Therefore, the complexity of Procedure Find-Consensus is bounded by

$$\sum_{k=1}^{d} \sum_{i=1}^{|T_1|} \sum_{j=1}^{|T_2|} O(k \times m_i \times n_j) \leqslant O(d^2 \times \sum_{i=1}^{|T_1|} m_i \times \sum_{j=1}^{|T_2|} n_j)$$
$$\leqslant O(d^2 \times |T_1| \times |T_2|).$$

Computing the array β takes $O(|T_1| \times |T_2|)$ time. Therefore, the total time used by our algorithm is $O(d^2 \times |T_1| \times |T_2|)$.

```
Procedure Find-Consensus
Input: Trees $T_1$, $T_2$ and an integer $d$.
Output: $tsize_b(i, j, k)$ where $1 \le i \le |T_1|$, $1 \le j \le |T_2|$ and $0 \le k \le d$.
initialize $fsize_b()$ and $tsize_b()$ for $0 \le k \le d$ as in Lemma 5.1 and Lemma 5.2;
for $i := 1$ to $|T_1|$ do
    for $j := 1$ to $|T_2|$ do begin
        for $s := 1$ to $m_i$ do
            for $t := 1$ to $n_j$ do
                compute $fsize_b(F_1[i_1, i_s], F_2[j_1, j_t], 0)$ as in Lemma 5.3;
        compute $tsize_b(i, j, 0)$ as in Lemma 5.4;
    end;
for $k := 1$ to $d$ do
    for $i := 1$ to $|T_1|$ do
        for $j := 1$ to $|T_2|$ do begin
            run Procedure Find-Consensus-1 on input $(i, j, k)$;
            compute $fsize_b(F_1[i], F_2[j], k)$ as in Lemma 5.5;
            compute $tsize_b(i, j, k)$ as in Lemma 5.6;
        end;

Procedure Find-Consensus-1
Input: $i, j, k$.
Output: $\max_{RM(i_{m_i}, j_{n_j})}\{Z(i_{m_i}, j_{n_j}, k)\}$.
/* Let $Z[i_s, j_t, k] = \max_{RM(i_s, j_t)}\{Z(i_s, j_t, k)\}$. */
/* Let $i_0 = 0$ and $j_0 = 0$. */
$Z[0, 0, k] : = 0$;
for $s := 1$ to $m_i$ do
    $Z[i_s, 0, k] :=$ min $\{|F_1[i_1, i_s]|, k\}$;
for $t := 1$ to $n_j$ do
    $Z[0, j_t, k] :=$ min $\{|F_2[j_1, j_t]|, k\}$;
for $s := 1$ to $m_i$ do
    for $t := 1$ to $n_j$ do
        $Z[i_s, j_t, k] :=$ max {
            $Z[i_{s-1}, j_t, k]$,
            $Z[i_s, j_{t-1}, k]$,
            $\max_{0 \le h \le k}\{Z[i_{s-1}, j_t, k-h] + tsize_b(T_1[i_s], \emptyset, h)\}$,
            $\max_{0 \le h \le k}\{Z[i_s, j_{t-1}, k-h] + tsize_b(\emptyset, T_2[j_t], h)\}$,
            $\max_{0 \le h \le k}\{Z[i_{s-1}, j_{t-1}, k-h] + tsize_b(i_s, j_t, h)\}$,
            };
```

Fig. 13. Procedure for computing $tsize_b(i, j, k)$.

When d is a constant, that is the same as the complexity of the best current algorithm for calculating the isolated-subtree distance of two trees [9], even though the SC problem addressed here appears to be harder than finding the distance.

6. Conclusion

In this paper we reviewed four edit-based distance measures for ordered labeled trees and established a hierarchy among the four measures. We presented a new algorithm for finding the similar consensus of two trees based on one of the measures, namely the isolated-subtree distance, originated from Tanaka and Tanaka [2]. The algorithm runs as fast as the best-known algorithm [9] for comparing two trees using the isolated-subtree distance when the allowed distance between the common substructures is a constant independent of the input trees.

Acknowledgements

We thank Professor Dennis Shasha of New York University for helpful discussions concerning this work. The work was partially supported by the National Science Foundation under Grant No. IRI-9531548 and by the Natural Sciences and Engineering Research Council of Canada under Grant No. OGP0046373.

References

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

[2] E. Tanaka, K. Tanaka, The tree-to-tree editing problem, Int. J. Pattern Recognition Artif. Intell. 2 (1988) 221–240.
[3] H. Samet, Distance transform for images represented by quadtrees, IEEE Trans. Pattern Anal. Mach. Intell. 4 (1982) 298–303.
[4] B. Moyer, K.S. Fu, A tree system approach for fingerprint pattern recognition, IEEE Trans. Pattern Anal. Mach. Intell. 8 (1986) 376–387.
[5] M. Neff, B.K. Boguraev, Dictionaries, dictionary grammars and dictionary entry parsing, Proceedings 27th Annual Meeting of the Association for Computational Linguistics, 1989.
[6] P. Kilpelainen, H. Mannila, Retrieval from hierarchical texts by partial patterns, Proceedings 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1993, pp. 214–222.
[7] K.-C. Tai, The tree-to-tree correction problem, J. Ass. Comput. Mach 26 (1979) 422–433.
[8] K. Zhang, D. Shasha, Simple fast algorithms for the editing distance between trees and related problems, SIAM J. Comput. 18 (1989) 1245–1262.
[9] K. Zhang, Fast algorithms for the constrained editing distance between ordered labeled trees and related problems, Pattern Recognition 28 (1995) 463–474.
[10] S.M. Selkow, The tree-to-tree editing problem, Inform. Process. Lett. 6 (1977) 184–186.
[11] J.T.L. Wang, K. Zhang, C.Y. Chang, Identifying approximately common substructures in trees based on a restricted edit distance, Inform. Sci. accepted.
[12] W. Yang, Identifying syntactic differences between two programs, Software — Practice Exper. 21 (1991) 739–755.
[13] T. Jiang, L. Wang, K. Zhang, Alignment of trees – an alternative to tree edit, Theoret. Comput. Sci. 143 (1995) 137–148.
[14] E.N. Adams, Consensus techniques and the comparison of taxonomic trees, Systematic Zool. 21 (1972) 390–397.
[15] D. Gusfield, Efficient algorithms for inferring evolutionary trees, Networks 21 (1991) 19–28.
[16] D.F. Robinson, L.R. Foulds, Comparison of phylogenetic trees, Math. Biosci. 53 (1981) 131–147.
[17] J. Ott, Analysis of Human Genetic Linkage, 2nd Edition, Johns Hopkins University Press, Baltimore, London, 1991.
[18] M.J. Rochkind, The source code control system, IEEE Trans. Software Eng. 1 (1975) 364–370.
[19] D. Angluin, Finding patterns common to a set of strings, J. Comput. System Sci. 21 (1980) 46–62.
[20] M. Crochemore, Transducers and repetitions, Theoreti, Comput. Sci. 45 (1986) 63–86.
[21] M.G. Main, R.J. Lorentz, An O(NlogN) algorithm for finding all repetitions in a string, J. Algorithms 5 (1984) 422–432.
[22] J.T.L. Wang, B.A. Shapiro, D. Shasha, K. Zhang, K.M. Currey, An algorithm for finding the largest approximately common substructures of two trees, IEEE Trans. Pattern Anal. Mach. Intell. 20 (1998) 889–895.
[23] J.T.L. Wang, K. Zhang, Identifying consensus of trees through alignment, submitted for publication.
[24] J.T.L. Wang, K. Zhang, K. Jeong, D. Shasha, A system for approximate tree matching, IEEE Trans. Knowledge Data Eng. 6 (1994) 559–571.
[25] J.T.L. Wang, T.G. Marr, D. Shasha, B.A. Shapiro, G-W. Chirn, Discovering active motifs in sets of related protein sequences and using them for classification, Nucleic Acids Res. 22 (1994) 2769–2775.
[26] S.B. Needleman, C.D. Wunsch, A general method applicable to the search for similarities in the amino acid sequence of two proteins, J. Mol. Biol. 48 (1970) 443–453.
[27] T.F. Smith, M.S. Waterman, Identification of common molecular subsequences, J. Mol. Biol. 147 (1981) 195–197.
[28] E. Ukkonen, Finding approximate patterns in strings, J. Algorithms 6 (1985) 132–137.

About the Author—JASON T.L. WANG received the B.S. degree in mathematics from National Taiwan University, Taipei, Taiwan, and the Ph.D. degree in computer science from the Courant Institute of Mathematical Sciences, New York University. Currently, he is an associate professor in the Department of Computer and Information Science at the New Jersey Institute of Technology, Newark, USA. Dr. Wang's research interests include pattern recognition, data mining and computational biology. He is an editor and author of the book *Pattern Discovery in Biomolecular Data* (Oxford University Press, 1999).

About the Author—KAIZHONG ZHANG recieved the M.S. degree in mathematics from Peking University, Beijing, People's Republic of China, in 1981, and the M.S. and Ph.D. degrees in computer science from the Courant Institute of Mathematical Sciences, New York University, New York, NY, USA, in 1986 and 1989, respectively. Currently, he is an associate professor in the Department of Computer Science, University of Western Ontario, London, ON, Canada. His research interests include pattern recognition, computational biology, sequential and parallel algorithms.