Theory Comput Syst (2017) 60:759–783
DOI 10.1007/s00224-016-9705-1

Tai Mapping Hierarchy for Rooted Labeled Trees Through Common Subforest

Takuya Yoshino[1] · **Kouichi Hirata**[1]

Published online: 2 September 2016

Abstract A *Tai mapping* between two rooted labeled trees (trees, for short) is a one-to-one node correspondence preserving ancestors and siblings (if trees are ordered). The variations of the Tai mapping are known to provide a hierarchy, called a *Tai mapping hierarchy*. In this paper, we characterize the Tai mapping hierarchy as a *common subforest* by focusing on the connections of nodes and the arrangements of subtrees in a common subforest. Then, we fill a gap in the Tai mapping hierarchy by introducing several new variations. Furthermore, we summarize and investigate the time complexity of computing the variations of the edit distance as the minimum cost of the variations of the Tai mapping.

Keywords Tai mapping · Tai mapping hierarchy · Common subforest · Tree edit distance

1 Introduction

Comparing tree-structured data such as HTML and XML data for web mining or DNA and glycan data for bioinformatics is one of the important tasks for data mining.

This work is partially supported by Grant-in-Aid for Scientific Research 16H02870, 16H01743, 15K12102, 26280085 and 24300060 from the Ministry of Education, Culture, Sports, Science and Technology, Japan.

✉ Kouichi Hirata
hirata@ai.kyutech.ac.jp

Takuya Yoshino
yoshino@dumbo.ai.kyutech.ac.jp

[1] Kyushu Institute of Technology, Kawazu 680-4, Iizuka 820-8502, Japan

The most famous distance measure between trees is the *edit distance* [2, 9, 14]. The edit distance is formulated as the minimum cost of *edit operations*, consisting of a *substitution*, a *deletion* and an *insertion*, applied to transform from a tree to another tree.

It is known that the edit distance is closely related to the notion of a *Tai mapping* (*mapping*, for short) [14], which is a one-to-one node correspondence between trees preserving ancestor (and sibling) relations. Note that the corresponding nodes with different labels through a mapping are regarded as nodes applied to substitution and the non-corresponding nodes to deletion in a tree or insertion in another tree. Then, the minimum cost of possible Tai mappings coincides with the edit distance [14].

Whereas the edit distance is the standard measure for comparing trees, it is too general for several applications. Therefore, more structurally sensitive distances of the edit distance are required for these applications. Such distances are formulated as the minimum cost of the variations of the Tai mapping such as a *top-down mapping* (TOP) [3, 13], an *LCA* (least common ancestor) *-preserving* (or *degree-2*) *mapping* (LCA) [22], an *accordant* (or *Lu's*) *mapping* (ACC) [9, 11], an *isolated-subtree* (or *constrained*) *mapping* (ILST) [17, 19, 20], a *less-constrained mapping* (LESS) [10], an *alignable mapping* (ALN) [9], a *bottom-up mapping* (BOT) [9, 15, 18], a *segmental mapping* (SG) [7] and a *top-down segmental mapping* (TOPSG) [7], respectively.

The above mappings provide a *Tai mapping hierarchy* illustrated in Fig. 1, which grows from left to right by adding new variations of the Tai mapping. Here, ISO denotes an isomorphism and it holds that "ALN=LESS" [9] and "TOP=TOPSG" [7].

The Tai mapping hierarchy in Fig. 1 just represents the inclusion relation of mappings but no other properties of mappings and, in particular, the right column in Fig. 1 (left) is sparser than the left column. Then, there arise a question whether or not there exists a unifying property between the variations of a Tai mapping.

In order to solve the above question, in this paper, we characterize the Tai mapping hierarchy as a *common subforest* between two trees consisting of pairs of nodes in the variations of a Tai mapping. Then, we focus on the *connections of nodes* and

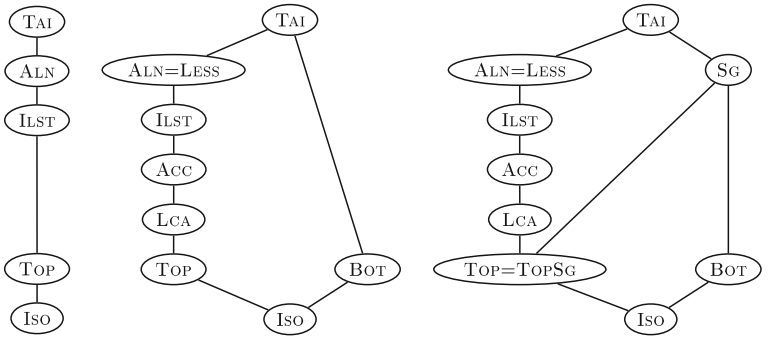

Fig. 1 Tai mapping hierarchies introduced by Wang and Zhang [17] (*left*), Kuboyama [9] (*center*) and Kan et al. [7] (*right*)

the *arrangements of subtrees* in a common subforest. Note that the *largest* common subforest or tree provides another similarity measure between trees [1, 5, 9, 16, 21].[1]

First, we focus on the following three connections of nodes in a common subforest. A common *embedded* subforest is a common subforest by connecting nodes in a mapping according to ancestor-descendant relation. A common *induced* subforest is a common subforest induced by a set of nodes. A common *complete* subforest is a common induced subforest containing all the descendants of a set of nodes. Then, we can characterize the mappings in TAI, ALN, ILST, ACC and LCA as common embedded subforests, the mappings in SG and TOP as common induced subforests and mappings in BOT as common complete subforests, respectively. The mappings characterized as common induced subforests require the operational condition that the deletion and the insertion are allowed to apply just roots or leaves and the mappings characterized as common complete subforests require the operational condition that the deletion and the insertion are allowed to apply just roots.

Next, we focus on the four arrangements of subtrees in a common subforest. A common subforest is *non-twisting* if there exists no triple of nodes in a common subforest such that the LCA of the first and the second nodes is an ancestor of the LCA of the second and third nodes in a tree iff the LCA of the second and the third nodes is an ancestor of the LCA of the first and second nodes in another tree. A common subforest is *parallel* if, for every triple of nodes in a common subforest, the LCA of the first and the second nodes is equal to the LCA of the first and the third nodes in a tree iff the LCA of the first and the second nodes is equal to the LCA of the first and the third nodes in another tree. A common subforest is a *subtree* if it is a tree. A common subtree is *root-preserving* if the root of the common subtree is equal to the roots of both trees. Then, we can characterize the mappings in TAI, SG and BOT as an arbitrary subforest, the mappings in ALN as a non-twisting subforest, the mappings in ILST and ACC as a parallel subforest, the mappings in LCA as a subtree and the mappings in TOP and ISO as a root-preserving subtree, respectively.

As a result, the Tai mapping hierarchy in Fig. 1 is not complete for the connections of nodes and the arrangements of subtrees in a common subforest. In order to fill a gap in the Tai mapping hierarchy in Fig. 1, in this paper, we introduce new mappings by intersecting SG and BOT in the right column in Fig. 1 with ALN, ILST, ACC and LCA in the left column in Fig. 1.

As intersecting SG, we introduce a *segmental alignable mapping* (SGALN), an *isolated-subtree segmental mapping* (ILSTSG), an *accordant segmental mapping* (ACCSG) and an *LCA-preserving segmental mapping* (LCASG). As intersecting BOT, we introduce a *bottom-up alignable mapping* (BOTALN), an *isolated-subtree bottom-up mapping* (ILSTBOT), an *accordant bottom-up mapping* (ACCBOT) and an *LCA-preserving bottom-up mapping* (LCABOT). Additionally, we introduce an *LCA- and root-preserving mapping* (LCART).

[1]The *largest common subtree* in the standard definition [1, 5, 9, 16, 21] is corresponding to the minimum cost of the *LCA-preserving segmental mapping* (LCASG, defined below) which consists of pairs of nodes with the same label.

Hence, we provide a new Tai mapping hierarchy illustrated in Fig. 2, where the previous mappings in Fig. 1 are illustrated as the nodes enclosed by gray lines and the new mappings are illustrated as the nodes enclosed by black lines. In particular, we show that "AccSg=IlstSg" and "AccBot=IlstBot."

In the vertical direction in Fig. 2, the mappings in the left column (Tai, Aln, Ilst, Acc, Lca and LcaRt), those in the center column (Sg, SgAln, AccSg, LcaSg and Top) and those in the right column are characterized as common embedded, induced and complete subforests, respectively. In the horizontal direction in Fig. 2, on the other hand, the mappings in the top row (Tai, Sg and Bot), those in the second row (Aln, SgAln and BotAln), those in the third row (Ilst, Acc, AccSg and AccBot), those in the fourth row (Lca, LcaSg, LcaBot) and those in the last row (LcaRt, Top and Iso) are characterized as arbitrary subforests, non-twisting subforests, parallel subforests, subtrees and root-preserving subtrees, respectively.

Next, we introduce the variations of the edit distance as the minimum cost of all the possible mappings in SgAln, AccSg, LcaSg, BotAln, AccBot, LcaBot and LcaRt, where we call them a *segmental alignment distance*, an *accordant segmental distance*, an *LCA-preserving segmental distance*, a *bottom-up alignment distance*, an *accordant bottom-up distance*, an *LCA-preserving bottom-up distance* and an *LCA- and root-preserving distance*. Then, we show that no distances characterized as non-twisting subforests are metrics, whereas the other distances are metrics.

Finally, we summarize and investigate the time complexity of computing the variations of the edit distance illustrated in Table 1. Here, n is the number of nodes in a tree, m is the number of nodes in another tree ($n \geq m$), D is the maximum degree of two trees and d is the minimum degree of two trees.

For ordered trees, we can compute the distances characterized as non-twisting subforests in $O(nmD^2)$ time and those as other subforests or subtrees except the (standard) edit distance and the tree isomorphism in $O(nm)$ time. For unordered

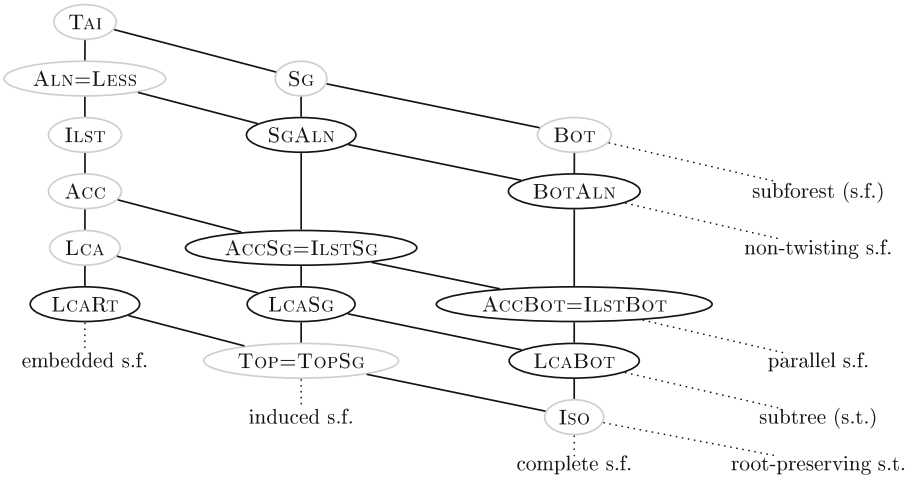

Fig. 2 A new Tai mapping hierarchy consisting of the mappings in Fig. 1 illustrated as the nodes enclosed by *gray lines* and the new mappings illustrated as the nodes enclosed by *black lines*

Table 1 The metricity of the variations $\tau_{\mathrm{A}}(T_1, T_2)$ for $\mathcal{M}_{\mathrm{A}}(T_1, T_2)$ in Fig. 2 and time complexity of computing them

Mapping		Ordered tree		Unordered tree			
$\mathcal{M}_{\mathrm{A}}(T_1, T_2)$	metric	τ^o_{A}		τ^u_{A}		(bounded degrees)	
TAI	yes	$O(nm^2(1+\log\frac{n}{m}))$	[4]	MAX SNP	[21]	MAX SNP	[5]
ALN	no	$O(nmD^2)$	[6]	MAX SNP	[6]	polynomial	[6]
ILST	yes	$O(nm)$	[19]	$O(nmd)$	[18]	$O(nm)$	←
ACC	yes	$O(nm)$	[9]	$O(nmd)$	[18]	$O(nm)$	←
LCA	yes	$O(nm)$	[22]	$O(nmd)$	[22]	$O(nm)$	←
LCART	yes	$O(nm)$	Thm 4	$O(nmd)$	Thm 4	$O(nm)$	←
SG	yes	$O(nm)$	[7]	MAX SNP	[18]	MAX SNP	[18]
SGALN	no	$O(nmD^2)$	Thm 6	MAX SNP	Thm 11	polynomial	Thm 12
ACCSG	yes	$O(mn)$	Thm 6	$O(nmd)$	Thm 7	$O(nm)$	←
LCASG	yes	$O(nm)$	Thm 5	$O(nmd)$	Thm 5	$O(nm)$	←
TOP	yes	$O(nm)$	[3, 13]	$O(nmd)$	[18]	$O(nm)$	←
BOT	yes	$O(nm)$	[18]	MAX SNP	[18]	MAX SNP	[18]
BOTALN	no	$O(nmD^2)$	Thm 9	MAX SNP	Thm 11	polynomial	Thm 12
ACCBOT	yes	$O(nm)$	Thm 9	$O(nmd)$	Thm 10	$O(nm)$	←
LCABOT	yes	$O(nm)$	Thm 8	$O(nm)$	Thm 8	←	
ISO	yes	$O(n+m)$	*cf.* [16]	$O(n+m)$	*cf.* [16]	←	

Here, n is the number of nodes in a tree, m is the number of nodes in another tree ($n \geq m$), D is the maximum degree of two trees and d is the minimum degree of two trees

trees, it is known that the problem of computing the distances characterized as arbitrary subforests is MAX SNP-hard [21] even if trees are binary [5, 18]. On the other hand, as in the case of alignment distance [6], the problems of computing the distances characterized as non-twisting subforests are MAX SNP-hard, whereas they are tractable if the degrees of trees are bounded by some constant. Furthermore, we can compute the distances characterized as parallel subforests, subtrees and root-preserving subtrees in $O(nmd)$ time except the LCA-preserving bottom-up distance, which we can compute in $O(nm)$ time.

2 Edit Distance and Tai Mapping

A *tree* T is a connected graph (V, E) without cycles, where V is the set of vertices and E is the set of edges and we denote V and E by $V(T)$ and $E(T)$. The *size* of T is $|V|$ and denoted by $|T|$. We sometime denote $v \in V(T)$ by $v \in T$. We denote an empty tree $(\emptyset, \emptyset)$ by $\emptyset$. A *rooted tree* is a tree with one node r chosen as its *root*. We denote the root of a rooted tree T by $r(T)$.

For each node v in a rooted tree with the root r, let $UP_r(v)$ be the unique path from v to r. The *parent* of $v(\neq r)$, which we denote by $par(v)$, is its adjacent node on $UP_r(v)$ and the *ancestors* of $v(\neq r)$ are the nodes on $UP_r(v) - \{v\}$. We denote the set of all ancestors of v by $anc(v)$. We say that u is a *child* of v if v is the parent of u and u is a *descendant* of v if v is an ancestor of u. A *leaf* is a node having no children. We denote the set of all leaves in T by $lv(T)$. A node neither a leaf nor a root is called an *internal node*. The *degree* of a node v, denoted by $d(v)$, is the number of children of v. The *degree* of a rooted tree T, denoted by $d(T)$, is the maximum number of $d(v)$ for every $v \in T$.

We use the ancestor orders $<$ and $\leq$, that is, $u < v$ if v is an ancestor of u and $u \leq v$ if $u < v$ or $u = v$. We denote neither $u \leq v$ nor $v \leq u$ by $u \# v$. We say that w is the *least common ancestor* (*LCA*, for abbreviation) of u and v, denoted by $u \sqcup v$, if $u \leq w$, $v \leq w$ and there exists no w' such that $w' < w$, $u \leq w'$ and $v \leq w'$. A (*complete*) *subtree of* $T = (V, E)$ *rooted by* v, denoted by $T[v]$, is a tree $T' = (V', E')$ such that $r(T') = v$, $V' = \{u \in V \mid u \leq v\}$ and $E' = \{(u, w) \in E \mid u, w \in V'\}$.

For nodes $u, v \in T$, u is *to the left of* v, denoted by $u \preceq v$, if $pre(u) \leq pre(v)$ for the preorder number *pre* and $post(u) \leq post(v)$ for the postorder number *post*. We say that a rooted tree is *ordered* if a left-to-right order among siblings is given; *unordered* otherwise. We say that a rooted tree is *labeled* if each node is assigned a symbol from a fixed finite alphabet Σ. For a node v, we denote the label of v by $l(v)$, and sometimes identify v with $l(v)$. We call a rooted labeled tree a *tree* simply.

A *forest* is a sequence $[T_1, \ldots, T_n]$ of trees. For a tree T and a node $i \in T$, $T(i)$ is a forest obtained by deleting the root i in $T[i]$. We denote the number of trees in a forest F by $||F||$, that is, $||F|| = n$ for $F = [T_1, \ldots, T_n]$.

In the following sections, we will use two trees T_1 and T_2 to obtain the distance between T_1 and T_2. Then, for nodes $i \in T_1$ and $j \in T_2$, suppose that the children of i and j are $i_1, \ldots, i_s$ and $j_1, \ldots, j_t$, respectively. Also, we sometimes denote the forests $T_1(i) = [T_1[i_1], \ldots, T_1[i_s]]$ and $T_2(j) = [T_2[j_1], \ldots, T_2[j_t]]$ by $F_1(i_1, i_s)$ and $F_2(j_1, j_t)$, respectively.

Definition 1 (Edit operations [14]) The *edit operations* of a tree T are defined as follows. See Fig. 3.

1. *Substitution*: Change the label of the node v in T.
2. *Deletion*:[2] Delete a node v in T with parent v', making the children of v become the children of v'. The children are inserted in the place of v as a subsequence in the left-to-right order (for ordered trees) or a subset (for unordered trees) of the children of v'. In particular, if v is the root in T, then the result applying the deletion is a forest consisting of the children of the root.
3. *Insertion*: The complement of deletion. Insert a node v as a child of v' in T making v the parent of a consecutive subsequence (for ordered trees) or a subset (for unordered trees) of the children of v'.

[2]In contrast to the standard definition of deletion and insertion [9, 14], this paper allows the deletion of a root, in order to characterize the bottom-up distance explicitly. Then, this paper also allows that the resulting tree applying to the deletion becomes a forest.

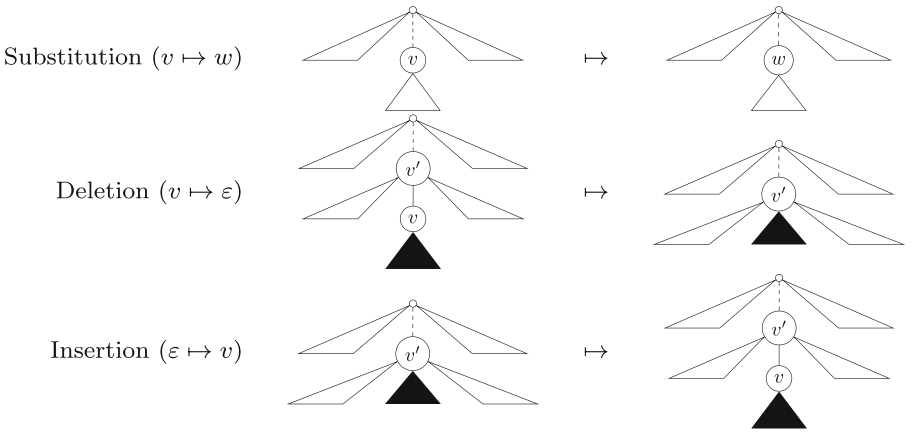

Fig. 3 Edit operations for trees

Let $\varepsilon \notin \Sigma$ denote a special *blank* symbol and define $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$. Then, we represent each edit operation by $(l_1 \mapsto l_2)$, where $(l_1, l_2) \in (\Sigma_\varepsilon \times \Sigma_\varepsilon - \{(\varepsilon, \varepsilon)\})$. The operation is a substitution if $l_1 \neq \varepsilon$ and $l_2 \neq \varepsilon$, a deletion if $l_2 = \varepsilon$, and an insertion if $l_1 = \varepsilon$. For nodes v and w, we also denote $(l(v) \mapsto l(w))$ by $(v \mapsto w)$. We define a *cost function* $\gamma : (\Sigma_\varepsilon \times \Sigma_\varepsilon - \{(\varepsilon, \varepsilon)\}) \mapsto \mathbf{R}^+$ on pairs of labels. We often constrain a cost function γ to be a *metric*, that is, $\gamma(l_1, l_2) \geq 0$, $\gamma(l_1, l_2) = 0$ iff $l_1 = l_2$, $\gamma(l_1, l_2) = \gamma(l_2, l_1)$ and $\gamma(l_1, l_3) \leq \gamma(l_1, l_2) + \gamma(l_2, l_3)$. We call the cost function that $\gamma(l_1, l_2) = 1$ if $l_1 \neq l_2$ a *unit cost function*.

Definition 2 (Edit distance [14]) For a cost function γ, the *cost* of an edit operation $e = l_1 \mapsto l_2$ is given by $\gamma(e) = \gamma(l_1, l_2)$. The *cost* of a sequence $E = e_1, \ldots, e_k$ of edit operations is given by $\gamma(E) = \sum_{i=1}^{k} \gamma(e_i)$. Then, an *edit distance* $\tau_{\mathrm{TAI}}(T_1, T_2)$ between trees T_1 and T_2 is defined as follows:

$$\tau_{\mathrm{TAI}}(T_1, T_2) = \min \left\{ \gamma(E) \;\middle|\; \begin{array}{l} E \text{ is a sequence of edit operations} \\ \text{transforming } T_1 \text{ to } T_2 \end{array} \right\}.$$

Definition 3 (Tai mapping [14]) Let T_1 and T_2 be trees. We say that a triple (M, T_1, T_2) is an *ordered Tai mapping* between T_1 and T_2 if $M \subseteq V(T_1) \times V(T_2)$ and every pair (u_1, v_1) and (u_2, v_2) in M satisfies that (1) $u_1 = u_2$ iff $v_1 = v_2$ (one-to-one condition), (2) $u_1 \leq u_2$ iff $v_1 \leq v_2$ (ancestor condition) and (3) $u_1 \preceq u_2$ iff $v_1 \preceq v_2$ (sibling condition). Also we say that a triple (M, T_1, T_2) is an *unordered Tai mapping* if $M \subseteq V(T_1) \times V(T_2)$ and every pair (u_1, v_1) and (u_2, v_2) in M satisfies (1) one-to-one condition and (2) ancestor condition. The set of all possible Tai mappings between T_1 and T_2 is denoted by $\mathcal{M}_{\mathrm{TAI}}(T_1, T_2)$. We will use M instead of (M, T_1, T_2) when there is no confusion, call both an ordered and an unordered Tai mapping a *mapping*.

In particular, we say that a mapping M is an *inclusion mapping* from T_1 to T_2 if, for every node $v \in T_1$, there exists a node $w \in T_2$ such that $(v, w) \in M$. In this case, we denote w by $M(v)$.

Let M be a mapping between T_1 and T_2. Let I_M and J_M be the sets of nodes in T_1 and T_2 but not in M, that is, $I_M = \{u \in T_1 \mid (u, v) \notin M\}$ and $J_M = \{v \in T_2 \mid (u, v) \notin M\}$. Then, the *cost* $\gamma(M)$ of M is given as follows.

$$\gamma(M) = \sum_{(u,v)\in M} \gamma(u, v) + \sum_{u\in I_M} \gamma(u, \varepsilon) + \sum_{v\in J_M} \gamma(\varepsilon, v).$$

Theorem 1 (Tai [14]) $\tau_{\mathrm{TAI}}(T_1, T_2) = \min\{\gamma(M) \mid M \in \mathcal{M}_{\mathrm{TAI}}(T_1, T_2)\}$.

Finally, we introduce the notion of a subforest, a common subforest and their arrangement of subtrees illustrated in Fig. 2. Note that we ignore labels of nodes.

Definition 4 (Subforest, common subforest) Let $T = (V, E)$ be a tree.

1. We say that $F = (V', E')$ is an *embedded subforest* in T if $V' \subseteq V$ and, for every $u, v \in V'$, $(u, v) \in E'$ if $u < v$ and there exists no $w \in V'$ such that $u < w$ and $w < v$.
2. We say that $F = (V', E')$ is an *induced subforest* in T if $V' \subseteq V$ and, for every $u, v \in V'$, $(u, v) \in E'$ if $(u, v) \in E$.
3. We say that $F = (V', E')$ is a *complete subforest* in T if F is an induced subforest in T and, for every $v \in V'$, all the descendants of v is in V'.

Furthermore, for trees T_1 and T_2, we say that F is a *common embedded* (*resp.*, *induced, complete*) *subforest* between T_1 and T_2 if F is an embedded (*resp.*, induced, complete) subforest in both T_1 and T_2. We call the above three common subforests a *common subforest* if it is not necessary to distinguish them.

Definition 5 (Arrangement of subtrees in a common subforest) Let F be a common subforest between T_1 and T_2. For a node $v \in F$, we denote $v \in T_1 \cap F$ and $v \in T_2 \cap F$ by v^1 and v^2, respectively.

1. We say that F is *twisting* if there exist $u, v, w \in F$ such that $u^1 \sqcup v^1 < v^1 \sqcup w^1$ in T_1 and $v^2 \sqcup w^2 < u^2 \sqcup v^2$ in T_2. Otherwise F is *non-twisting*.
2. We say that F is *parallel* if, for every $u, v, w \in F$, $u^1 \sqcup v^1 = u^1 \sqcup w^1$ in T_1 iff $u^2 \sqcup v^2 = u^2 \sqcup w^2$ in T_2.
3. We say that F is a *subtree* if F is a tree, that is, $||F|| = 1$.
4. We say that F is a *root-preserving subtree* if F is a tree T such that $r(T) = r(T_1) = r(T_2)$.

Example 1 Consider trees T_1 and T_2 in Fig. 4. Then, F_1, F_2 and F_3 illustrated in the rightmost column in Fig. 4 are the common embedded subforest, the common induced subforest and the common complete subforest between T_1 and T_2, respectively. Here, every v^1 in T_1 (*resp.*, v^2 in T_2) for a node $v \in F_i$ ($i = 1, 2, 3$) is illustrated as gray nodes. Also F_1 is a root-preserving subtree and F_2 and F_3 are parallel subforests.

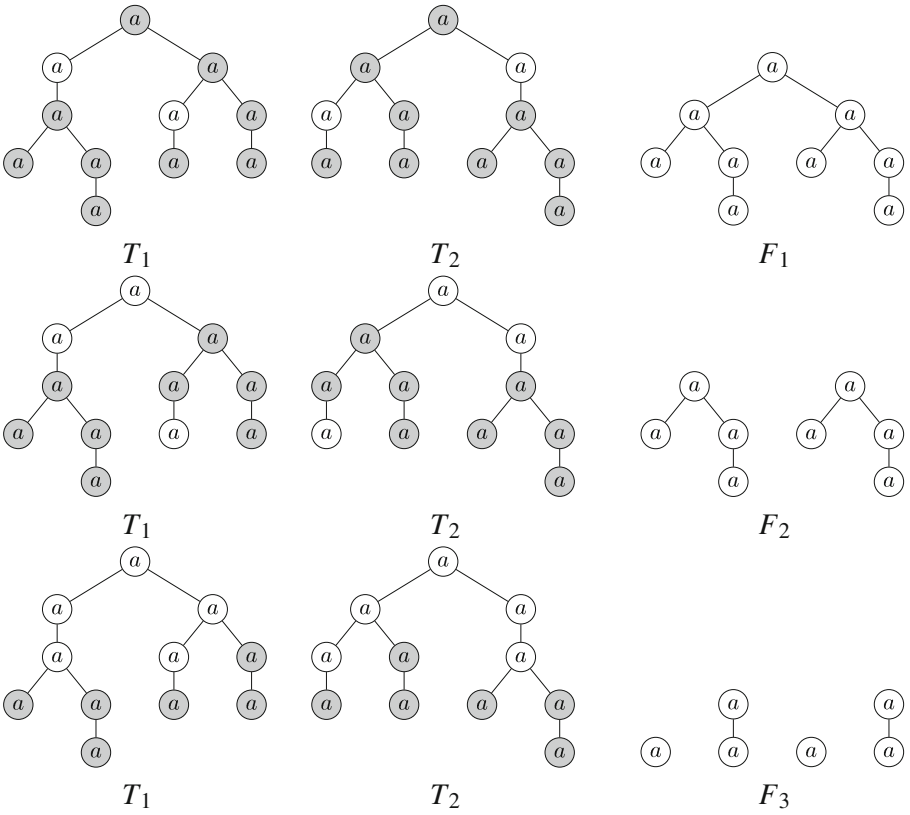

Fig. 4 The common embedded subforest F_1, the common induced subforest F_2 and the common complete subforest F_3 between T_1 and T_2

3 Tai Mapping Hierarchy

In this section, we introduce the variations of a Tai mapping.

Definition 6 (Variations of mapping) Let T_1 and T_2 be trees and $M \in \mathcal{M}_{\mathrm{TAI}}(T_1, T_2)$. We define M^- as $M - \{(r(T_1), r(T_2))\}$.

1. We say that M is a *less-constrained mapping* [10], denoted by $M \in \mathcal{M}_{\mathrm{LESS}}(T_1, T_2)$, if M satisfies the following condition.

$$\forall (u_1, v_1), (u_2, v_2), (u_3, v_3) \in M \, (u_1 \sqcup u_2 < u_1 \sqcup u_3 \Longrightarrow v_2 \sqcup v_3 = v_1 \sqcup v_3) \, .$$

2. We say that M is an *isolated-subtree mapping* [17] (or a *constrained mapping* [19]), denoted by $M \in \mathcal{M}_{\mathrm{ILST}}(T_1, T_2)$, if M satisfies the following condition.

$$\forall (u_1, v_1), (u_2, v_2), (u_3, v_3) \in M \, (u_3 < u_1 \sqcup u_2 \iff v_3 < v_1 \sqcup v_2) \, .$$

3. We say that M is an *accordant mapping* [9] (or a *Lu's mapping* [11]), denoted by $M \in \mathcal{M}_{\mathrm{ACC}}(T_1, T_2)$, if M satisfies the following condition.

$$\forall (u_1, v_1), (u_2, v_2), (u_3, v_3) \in M \, (u_1 \sqcup u_2 = u_1 \sqcup u_3 \iff v_1 \sqcup v_2 = v_1 \sqcup v_3) \, .$$

4. We say that M is an *LCA-preserving mapping* (or a *degree-2 mapping* [22]), denoted by $M \in \mathcal{M}_{\mathrm{LCA}}(T_1, T_2)$, if M satisfies the following condition.

$$\forall (u_1, v_1), (u_2, v_2) \in M^{-} \, ((u_1 \sqcup u_2, v_1 \sqcup v_2) \in M) \, .$$

5. We say that M is an *LCA- and root-preserving mapping*, denoted by $M \in \mathcal{M}_{\mathrm{LCART}}(T_1, T_2)$, if $M \in \mathcal{M}_{\mathrm{LCA}}(T_1, T_2)$ and $(r(T_1), r(T_2)) \in M$.
6. We say that M is a *top-down mapping* [3, 13] (or a *degree-1 mapping*), denoted by $M \in \mathcal{M}_{\mathrm{TOP}}(T_1, T_2)$, if M satisfies the following condition.

$$\forall (u, v) \in M^{-} \, ((par(u), par(v)) \in M) \, .$$

7. We say that M is a *bottom-up mapping* [9, 15, 18],[3] denoted by $M \in \mathcal{M}_{\mathrm{BOT}}(T_1, T_2)$, if M satisfies the following condition.

$$\forall (u, v) \in M \left(\begin{array}{l} \forall u' \in T_1[u] \exists v' \in T_2[v] \left((u', v') \in M \right) \\ \wedge \forall v' \in T_2[v] \exists u' \in T_1[u] \left((u', v') \in M \right) \end{array} \right) .$$

8. We say that M is a *segmental mapping* [7], denoted by $M \in \mathcal{M}_{\mathrm{SG}}(T_1, T_2)$, if M satisfies the following condition.

$$\forall (u, v) \in M^{-} \left(\begin{array}{c} \exists (u', v') \in M \Big(\big(u' \in anc(u) \big) \wedge \big(v' \in anc(v) \big) \Big) \\ \Longrightarrow ((par(u), par(v)) \in M) \end{array} \right) .$$

9. We say that M is a *top-down segmental mapping* [7], denoted by $M \in \mathcal{M}_{\mathrm{TOPSG}}(T_1, T_2)$, if M is a segmental mapping such that $(r(T_1), r(T_2)) \in M$.
10. We say that a mapping M is an *alignable mapping* [9], denoted by $M \in \mathcal{M}_{\mathrm{ALN}}(T_1, T_2)$, if there exist a forest F, an inclusion mapping M_1 from T_1 to F and an inclusion mapping M_2 from T_2 to F satisfying that $M_1(u) = M_2(v)$ for every $(u, v) \in M$.[4]
11. $\mathcal{M}_{\mathtt{AB}}(T_1, T_2) = \mathcal{M}_{\mathtt{A}}(T_1, T_2) \cap \mathcal{M}_{\mathtt{B}}(T_1, T_2)$ for mappings A and B. This paper deals with $\mathcal{M}_{\mathrm{SGALN}}(T_1, T_2)$, $\mathcal{M}_{\mathrm{ILSTSG}}(T_1, T_2)$, $\mathcal{M}_{\mathrm{ACCSG}}(T_1, T_2)$, $\mathcal{M}_{\mathrm{LCASG}}(T_1, T_2)$, $\mathcal{M}_{\mathrm{BOTALN}}(T_1, T_2)$, $\mathcal{M}_{\mathrm{ILSTBOT}}(T_1, T_2)$, $\mathcal{M}_{\mathrm{ACCBOT}}(T_1, T_2)$ and $\mathcal{M}_{\mathrm{LCABOT}}(T_1, T_2)$.

Example 2 Figure 5 illustrates mappings M_i $(1 \leq i \leq 8)$ [7, 9] such that $M_1 \in \mathcal{M}_{\mathrm{TOP}}(T_1, T_2)$ but $M_1 \notin \mathcal{M}_{\mathrm{BOT}}(T_1, T_2)$; $M_2 \in \mathcal{M}_{\mathrm{LCA}}(T_1, T_2)$ but $M_2 \notin \mathcal{M}_{\mathrm{TOP}}(T_1, T_2)$; $M_3 \in \mathcal{M}_{\mathrm{ACC}}(T_1, T_2)$ but $M_3 \notin \mathcal{M}_{\mathrm{LCA}}(T_1, T_2)$; $M_4 \in \mathcal{M}_{\mathrm{ILST}}(T_1, T_2)$

[3] Whereas Valiente [15] has introduced a bottom-up mapping that requires an isolated-subtree mapping, his algorithm computes a bottom-up distance that is not an isolated-subtree distance. Hence, we adopt the revised definition here not to be an isolated-subtree mapping. See [9, 18].

[4] In the definition of an alignable mapping [9], F is not a forest but a tree, because we can assume that the alignable mapping always contains the pair of the roots of two trees, corresponding to the alignment tree [6]. On the other hand, since the intersection of the alignable mapping to other mappings does not always contain the pair of the roots of two trees, we use a forest F in the definition of an alignable mapping.

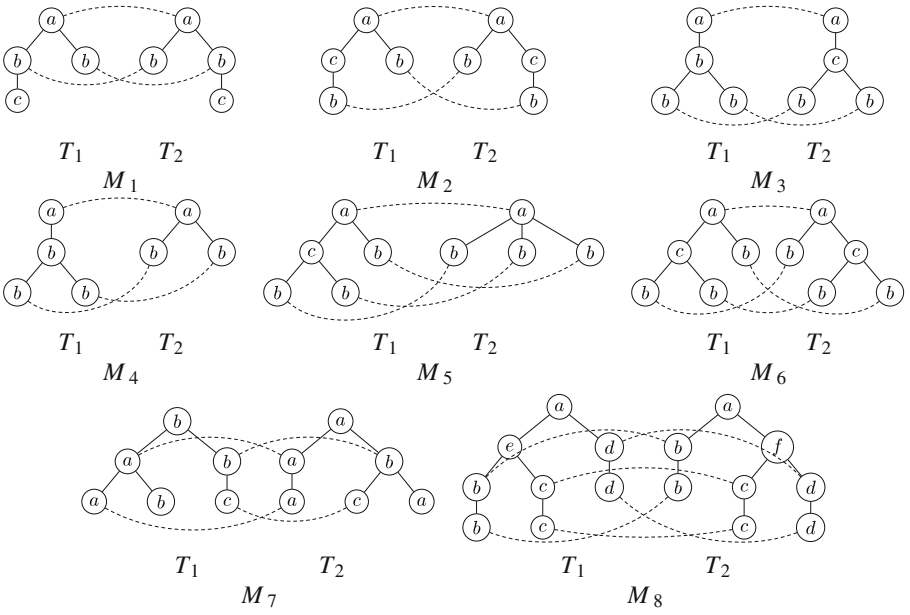

Fig. 5 Mappings M_i $(1 \leq i \leq 8)$ in Example 2

but $M_4 \notin \mathcal{M}_{\mathrm{ACC}}(T_1, T_2)$; $M_5 \in \mathcal{M}_{\mathrm{ALN}}(T_1, T_2)$ but $M_5 \notin \mathcal{M}_{\mathrm{ILST}}(T_1, T_2)$; $M_6 \in \mathcal{M}_{\mathrm{TAI}}(T_1, T_2)$ but $M_6 \notin \mathcal{M}_{\mathrm{ALN}}(T_1, T_2)$; $M_7 \in \mathcal{M}_{\mathrm{SG}}(T_1, T_2)$ but $M_7 \notin \mathcal{M}_{\mathrm{TOP}}(T_1, T_2)$ and $M_7 \notin \mathcal{M}_{\mathrm{BOT}}(T_1, T_2)$; $M_8 \in \mathcal{M}_{\mathrm{BOT}}(T_1, T_2)$ but $M_8 \notin \mathcal{M}_{\mathrm{ALN}}(T_1, T_2)$.

In particular, for a segmental mapping $M \in \mathcal{M}_{\mathrm{SG}}(T_1, T_2)$, the formula in Definition 6.8 claims that, for $(u, v), (u', v') \in M$ such that $u' \in anc(u)$ and $v' \in anc(v)$, there exist nodes $u_1, \ldots, u_k \in T_1$ and $v_1, \ldots, v_k \in T_2$ such that $u_1 = u$, $u_k = u'$, $v_1 = v$, $v_k = v'$, $u_{i+1} = par(u_i)$, $v_{i+1} = par(v_i)$ $(1 \leq i \leq k-1)$ and $(u_i, v_i) \in M$ $(1 \leq i \leq k)$.

The following lemma has been shown in [7, 9, 17] or holds from Definition 6.

Lemma 1 *The following statements hold.*

1. $\mathcal{M}_{\mathrm{ISO}}(T_1, T_2) \subset \mathcal{M}_{\mathrm{TOP}}(T_1, T_2) \subset \mathcal{M}_{\mathrm{LCA}}(T_1, T_2) \subset \mathcal{M}_{\mathrm{ACC}}(T_1, T_2) \subset \mathcal{M}_{\mathrm{ILST}}(T_1, T_2) \subset \mathcal{M}_{\mathrm{ALN}}(T_1, T_2) \subset \mathcal{M}_{\mathrm{TAI}}(T_1, T_2)$ [9, 17].
2. $\mathcal{M}_{\mathrm{ISO}}(T_1, T_2) \subset \mathcal{M}_{\mathrm{BOT}}(T_1, T_2) \subset \mathcal{M}_{\mathrm{SG}}(T_1, T_2) \subset \mathcal{M}_{\mathrm{TAI}}(T_1, T_2)$ [7].
3. $\mathcal{M}_{\mathtt{AB}}(T_1, T_2) \subseteq \mathcal{M}_{\mathtt{A}}(T_1, T_2)$ *for mappings* $\mathtt{A}$ *and* $\mathtt{B}$.
4. *If* $\mathcal{M}_{\mathtt{B}}(T_1, T_2) \subseteq \mathcal{M}_{\mathtt{C}}(T_1, T_2)$, *then* $\mathcal{M}_{\mathtt{AB}}(T_1, T_2) \subseteq \mathcal{M}_{\mathtt{AC}}(T_1, T_2)$ *for mappings* $\mathtt{A}$, $\mathtt{B}$ *and* $\mathtt{C}$.

Theorem 2 *The hierarchy illustrated in Fig. 2 holds, where the mapping* $\mathcal{M}_{\mathtt{A}}(T_1, T_2)$ *is denoted by its subscript* $\mathtt{A}$. *In particular, the following statements hold. Here,* $S \# S'$ *denotes that neither* $S \subseteq S'$ *nor* $S' \subseteq S$ *for two sets* S *and* S'.

1. $\mathcal{M}_{\text{ACCSG}}(T_1, T_2) = \mathcal{M}_{\text{ILSTSG}}(T_1, T_2)$ *and* $\mathcal{M}_{\text{ACCBOT}}(T_1, T_2) = \mathcal{M}_{\text{ILSTBOT}}(T_1, T_2)$
2. $\mathcal{M}_{\text{LCASG}}(T_1, T_2) \# \mathcal{M}_{\text{LCART}}(T_1, T_2)$, $\mathcal{M}_{\text{TOP}}(T_1, T_2) \subset \mathcal{M}_{\text{LCASG}}(T_1, T_2) \subset \mathcal{M}_{\text{LCA}}(T_1, T_2)$ *and* $\mathcal{M}_{\text{TOP}}(T_1, T_2) \subset \mathcal{M}_{\text{LCART}}(T_1, T_2) \subset \mathcal{M}_{\text{LCA}}(T_1, T_2)$
3. $\mathcal{M}_{\text{LCASG}}(T_1, T_2) \subset \mathcal{M}_{\text{ACCSG}}(T_1, T_2)$, $\mathcal{M}_{\text{ACCSG}}(T_1, T_2) \subset \mathcal{M}_{\text{ACC}}(T_1, T_2)$ *and* $\mathcal{M}_{\text{ACCSG}}(T_1, T_2) \subset \mathcal{M}_{\text{SGALN}}(T_1, T_2)$.
4. $\mathcal{M}_{\text{LCA}}(T_1, T_2) \# \mathcal{M}_{\text{ACCSG}}(T_1, T_2)$.
5. $\mathcal{M}_{\text{SGALN}}(T_1, T_2) \subset \mathcal{M}_{\text{SG}}(T_1, T_2)$ *and* $\mathcal{M}_{\text{BOTALN}}(T_1, T_2) \subset \mathcal{M}_{\text{BOT}}(T_1, T_2)$.
6. $\mathcal{M}_{\text{BOTALN}}(T_1, T_2) \subset \mathcal{M}_{\text{SGALN}}(T_1, T_2)$.
7. $\mathcal{M}_{\text{LCABOT}}(T_1, T_2) \subset \mathcal{M}_{\text{ACCBOT}}(T_1, T_2) \subset \mathcal{M}_{\text{BOTALN}}(T_1, T_2) \subset \mathcal{M}_{\text{BOT}}(T_1, T_2)$.

Proof 1. We just show that $\mathcal{M}_{\text{ACCSG}}(T_1, T_2) = \mathcal{M}_{\text{ILSTSG}}(T_1, T_2)$. We can show that $\mathcal{M}_{\text{ACCBOT}}(T_1, T_2) = \mathcal{M}_{\text{ILSTBOT}}(T_1, T_2)$ in the similar way. Since Lemma 1 implies that $\mathcal{M}_{\text{ACCSG}}(T_1, T_2) \subseteq \mathcal{M}_{\text{ILSTSG}}(T_1, T_2)$, we just show the converse direction that $\mathcal{M}_{\text{ILSTSG}}(T_1, T_2) \subseteq \mathcal{M}_{\text{ACCSG}}(T_1, T_2)$.

Suppose that $M \in \mathcal{M}_{\text{ILSTSG}}(T_1, T_2)$ and let $(u_1, v_1), (u_2, v_2), (u_3, v_3) \in M$. Then, it holds that $u_3 < u_1 \sqcup u_2 \iff v_3 < v_1 \sqcup v_2$. Also suppose that $M \notin \mathcal{M}_{\text{ACCSG}}(T_1, T_2)$. Consider the case that there exist $(u_1, v_1), (u_2, v_2), (u_3, v_3) \in M$ such that $u_1 \sqcup u_2 = u_1 \sqcup u_3$ but $v_1 \sqcup v_2 \neq v_1 \sqcup v_3$.

Suppose that $u_1 \# u_2$, $u_2 \# u_3$ and $u_3 \# u_1$. Since M is a mapping, it holds that $v_1 \# v_2$, $v_2 \# v_3$ and $v_3 \# v_1$. Since $v_1 \sqcup v_2 \neq v_1 \sqcup v_3$ and T_2 is a tree, it holds that either (1) $v_1 \sqcup v_2 < v_1 \sqcup v_3$ or (2) $v_1 \sqcup v_3 < v_1 \sqcup v_2$ in T_2. (See Fig. 6).

For the case (1), it holds that $v_3 \not< v_1 \sqcup v_2$, which is a contradiction that $u_3 < u_1 \sqcup u_2 \iff v_3 < v_1 \sqcup v_2$. For the case (2), it holds that $v_2 \not< v_1 \sqcup v_3$, which is a contradiction that $u_2 < u_1 \sqcup u_3 \iff v_2 < v_1 \sqcup v_3$. (Note that this discussion holds for a non-segmental mapping).

Suppose that $u_1 \# u_2$, $u_1 < u_3$ and $u_2 < u_3$. Since $u_1 \sqcup u_2 = u_1 \sqcup u_3$, it holds that $u_1 \sqcup u_2 = u_3$ in T_1. On the other hand, since $v_1 \sqcup v_2 \neq v_1 \sqcup v_3$, it holds that $v_1 \sqcup v_2 < v_1 \sqcup v_3 = v_3$ in T_2. (See the case (3) in Fig. 6).

Since M is a segmental mapping, there exists a node $u' \in T_1$ such that $(u', v_1 \sqcup v_2) \in M$. Since $v_1 \sqcup v_2 < v_3$, it holds that $u' < u_3 = u_1 \sqcup u_2$. On the other hand,

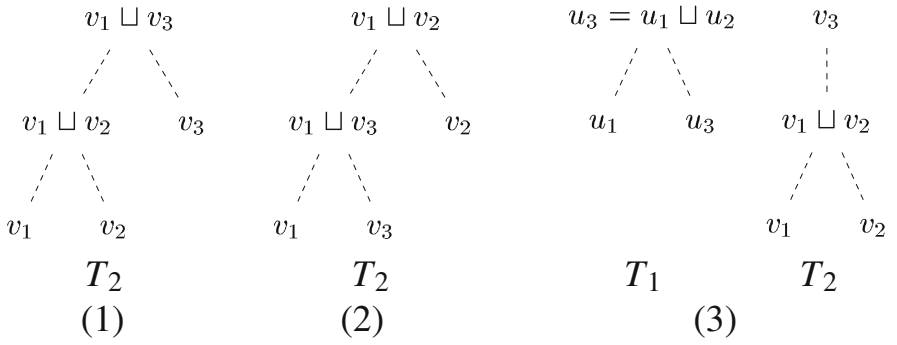

Fig. 6 The cases (1), (2) and (3)

since $v_1 < v_1 \sqcup v_2$ and $v_2 < v_1 \sqcup v_2$, it holds that $u_1 < u'$ and $u_2 < u'$, that is, $u_3 = u_1 \sqcup u_2 \leq u'$, which is a contradiction. The above discussion also holds for the case that there exist $(u_1, v_1), (u_2, v_2), (u_3, v_3) \in M$ such that $v_1 \sqcup v_2 = v_1 \sqcup v_3$ but $u_1 \sqcup u_2 \neq u_1 \sqcup u_3$.

Hence, for every $(u_1, v_1), (u_2, v_2), (u_3, v_3) \in M$, it holds that $u_1 \sqcup u_2 = u_1 \sqcup u_3 \iff v_1 \sqcup v_2 = v_1 \sqcup v_3$, which implies that $M \in \mathcal{M}_{\mathrm{ACCSG}}(T_1, T_2)$.

In order to show the other statements, it is necessary to show the inclusion and the properness.

For the inclusion, by Lemma 1, it is sufficient to show that $\mathcal{M}_{\mathrm{TOP}}(T_1, T_2) \subseteq \mathcal{M}_{\mathrm{LCART}}(T_1, T_2)$ and $\mathcal{M}_{\mathrm{TOP}}(T_1, T_2) \subseteq \mathcal{M}_{\mathrm{LCASG}}(T_1, T_2)$. Since $(r(T_1), r(T_2)) \in M$ for every $M \in \mathcal{M}_{\mathrm{TOP}}(T_1, T_2)$ and $\mathcal{M}_{\mathrm{TOP}}(T_1, T_2) \subset \mathcal{M}_{\mathrm{LCA}}(T_1, T_2)$, it holds that $M \in \mathcal{M}_{\mathrm{LCART}}(T_1, T_2)$. Since $\mathcal{M}_{\mathrm{TOP}}(T_1, T_2) = \mathcal{M}_{\mathrm{TOPSG}}(T_1, T_2)$ [7] and $\mathcal{M}_{\mathrm{TOP}}(T_1, T_2) \subseteq \mathcal{M}_{\mathrm{LCA}}(T_1, T_2)$, it holds that $\mathcal{M}_{\mathrm{TOP}}(T_1, T_2) \subseteq \mathcal{M}_{\mathrm{LCASG}}(T_1, T_2)$ by Lemma 1.4.

On the other hand, we show the properness by using the mappings M_i $(1 \leq i \leq 9)$ in Fig. 7.

2. It holds that $M_1 \in \mathcal{M}_{\mathrm{LCASG}}(T_1, T_2)$ but $M_1 \notin \mathcal{M}_{\mathrm{TOP}}(T_1, T_2)$ and $M_1 \notin \mathcal{M}_{\mathrm{LCART}}(T_1, T_2)$; $M_2 \in \mathcal{M}_{\mathrm{LCART}}(T_1, T_2)$ but $M_2 \notin \mathcal{M}_{\mathrm{TOP}}(T_1, T_2)$ and $M_2 \notin \mathcal{M}_{\mathrm{LCASG}}(T_1, T_2)$; $M_3 \in \mathcal{M}_{\mathrm{LCA}}(T_1, T_2)$ but $M_3 \notin \mathcal{M}_{\mathrm{LCASG}}(T_1, T_2)$ and $M_3 \notin \mathcal{M}_{\mathrm{LCART}}(T_1, T_2)$.

3. It holds that $M_4 \in \mathcal{M}_{\mathrm{ACCSG}}(T_1, T_2)$ but $M_4 \notin \mathcal{M}_{\mathrm{LCASG}}(T_1, T_2)$; $M_5 \in \mathcal{M}_{\mathrm{ACC}}(T_1, T_2)$ but $M_5 \notin \mathcal{M}_{\mathrm{ACCSG}}(T_1, T_2)$; $M_6 \in \mathcal{M}_{\mathrm{SGALN}}(T_1, T_2)$ but $M_6 \notin \mathcal{M}_{\mathrm{ACCSG}}(T_1, T_2)$.

4. It holds that $M_3 \in \mathcal{M}_{\mathrm{LCA}}(T_1, T_2)$ but $M_3 \notin \mathcal{M}_{\mathrm{ACCSG}}(T_1, T_2)$; $M_4 \in \mathcal{M}_{\mathrm{ACCSG}}(T_1, T_2)$ but $M_4 \notin \mathcal{M}_{\mathrm{LCA}}(T_1, T_2)$.

5. It holds that $M_7 \in \mathcal{M}_{\mathrm{SG}}(T_1, T_2)$ but $M_7 \notin \mathcal{M}_{\mathrm{SGALN}}(T_1, T_2)$. Also it holds that $M_7 \in \mathcal{M}_{\mathrm{BOT}}(T_1, T_2)$ but $M_7 \notin \mathcal{M}_{\mathrm{BOTALN}}(T_1, T_2)$.

6. It holds that $M_8 \in \mathcal{M}_{\mathrm{SGALN}}(T_1, T_2)$ but $M_8 \notin \mathcal{M}_{\mathrm{BOTALN}}(T_1, T_2)$.

7. It holds that $M_9 \in \mathcal{M}_{\mathrm{ACCBOT}}(T_1, T_2)$ but $M_9 \notin \mathcal{M}_{\mathrm{LCABOT}}(T_1, T_2)$; $M_6 \in \mathcal{M}_{\mathrm{BOTALN}}(T_1, T_2)$ but $M_6 \notin \mathcal{M}_{\mathrm{ACCBOT}}(T_1, T_2)$; $M_7 \in \mathcal{M}_{\mathrm{BOT}}(T_1, T_2)$ but $M_7 \notin \mathcal{M}_{\mathrm{BOTALN}}(T_1, T_2)$. □

Remark 1 We can characterize a common subforest between T_1 and T_2 by using a mapping M. Recall that I_M and J_M are the sets of nodes in T_1 and T_2 but not in M. Then, for $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$, M induces a common subforest $F = (V', E')$ between T_1 and T_2 such that $V' = V_1 - I_M = V_2 - J_M$. We call such a common subforest a *common subforest through* M.

Since the segmental mapping requires to preserve the parent-children relationship as possible, a common subforest through the segmental mapping and its sub-mappings, that is, SG, SGALN, ACCSG, LCASG and TOP, is an induced subforest. These mappings require the operational condition that the deletion and the insertion are allowed to apply just leaves or roots. In particular, the mappings in TOP require the operational condition that the deletion and the insertion are allowed to apply just leaves.

Since the bottom-up mapping requires to preserve the complete subtrees, a common subforest through the bottom-up mapping and its sub-mappings, that is, BOT,

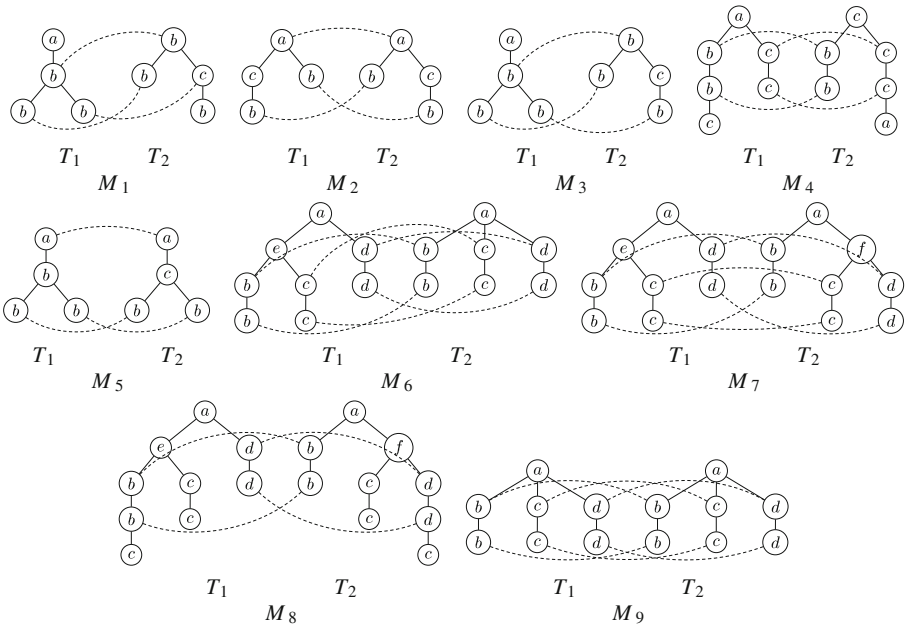

Fig. 7 Mappings M_i $(1 \leq i \leq 9)$ in the proof of Theorem 2

BOTALN, ACCBOT, LCABOT and ISO, is a complete subforest. These mappings require the operational condition that the deletion and the insertion are allowed to apply just roots.

Remark 2 Since the LCA-preserving mapping provides the root as an LCA of all the nodes in a common subforest, the common subforest through the LCA-preserving mapping and its sub-mappings, that is, LCA, LCASG, LCABOT, LCART, TOP and ISO, is a subtree. In particular, the LCA- and root-preserving mapping contains a pair of roots in given trees, the common subforest through the LCA- and root-preserving mapping and its sub-mappings, that is, LCART, TOP and ISO, is a root-preserving subtree.

Since the definition of the accordant mapping is same as a parallel common forest, the common subforest through the accordant mapping and its sub-mappings, that is, ACC, ACCSG and ACCBOT, is parallel.

As similar as a twisting common subforest, we say that a mapping M between trees T_1 and T_2 has a *twist* if there exist pairs $(u_1, v_1), (u_2, v_2), (u_3, v_3) \in M$ satisfying the following condition.

$$(u_1 \sqcup u_2 < u_2 \sqcup u_3) \wedge (v_2 \sqcup v_3 < v_1 \sqcup v_2).$$

We call such pairs (u_1, v_1), (u_2, v_2) and (u_3, v_3) a *twist* of M.

Since $(v_1 \sqcup v_2 \leq v_2 \sqcup v_3) \equiv (v_1 \sqcup v_3 = v_2 \sqcup v_3)$ and $\neg(v_2 \sqcup v_3 < v_1 \sqcup v_2) \equiv (v_1 \sqcup v_2 \leq v_2 \sqcup v_3)$ for every tree T and $v_1, v_2, v_3 \in T$, we can

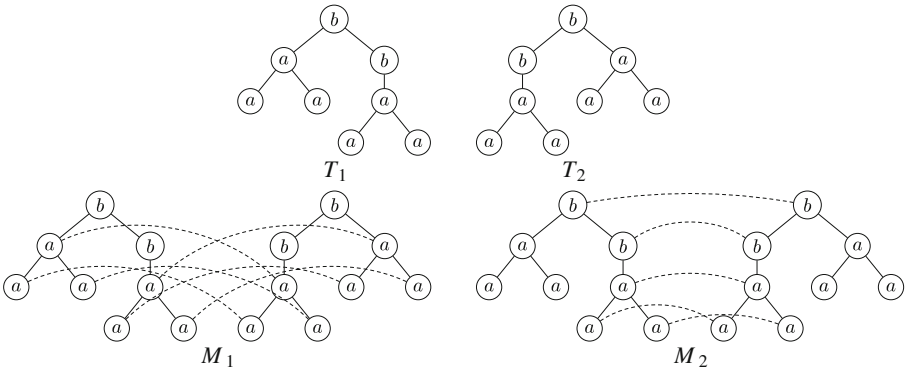

Fig. 8 Trees T_1 and T_2 and mappings M_1 and M_2 in Example 3

transform the formula representing the nonexistence of twists in M logically as follows.

$$
\begin{aligned}
&\neg\exists(u_1, v_1)(u_2, v_2)(u_3, v_3) \in M\,((u_1 \sqcup u_2 < u_2 \sqcup u_3) \wedge (v_2 \sqcup v_3 < v_1 \sqcup v_2))\\
\equiv\ &\forall(u_1, v_1)(u_2, v_2)(u_3, v_3) \in M \neg\,((u_1 \sqcup u_2 < u_2 \sqcup u_3) \wedge (v_2 \sqcup v_3 < v_1 \sqcup v_2))\\
\equiv\ &\forall(u_1, v_1)(u_2, v_2)(u_3, v_3) \in M\,(\neg(u_1 \sqcup u_2 < u_2 \sqcup u_3) \vee \neg(v_2 \sqcup v_3 < v_1 \sqcup v_2))\\
\equiv\ &\forall(u_1, v_1)(u_2, v_2)(u_3, v_3) \in M\,((u_1 \sqcup u_2 < u_2 \sqcup u_3) \Longrightarrow \neg(v_2 \sqcup v_3 < v_1 \sqcup v_2))\\
\equiv\ &\forall(u_1, v_1)(u_2, v_2)(u_3, v_3) \in M\,((u_1 \sqcup u_2 < u_2 \sqcup u_3) \Longrightarrow (v_1 \sqcup v_2 \leq v_2 \sqcup v_3))\\
\equiv\ &\forall(u_1, v_1)(u_2, v_2)(u_3, v_3) \in M\,((u_1 \sqcup u_2 < u_2 \sqcup u_3) \Longrightarrow (v_1 \sqcup v_3 = v_2 \sqcup v_3))\,.
\end{aligned}
$$

Hence, a mapping M is less-constrained if and only if M has no twists. As a result, the common subforest through the alignable mapping and its sub-mappings, that is, ALN, SGALN and BOTALN, is non-twisting.

Definition 7 (Variations of edit distance) For every $\mathcal{M}_{\mathbb{A}}(T_1, T_2)$ in a mapping hierarchy in Fig. 2, the variation $\tau_{\mathbb{A}}(T_1, T_2)$ of the edit distance τ_{TAI} is defined as the minimum cost of mappings in $\mathcal{M}_{\mathbb{A}}(T_1, T_2)$:

$\tau_{\mathbb{A}}(T_1, T_2) = \min\{\gamma(M) \mid M \in \mathcal{M}_{\mathbb{A}}(T_1, T_2)\}$.

Remark 3 If the cost function is a metric, then it holds that $\tau_{\mathrm{LCART}}(T_1, T_2) = \tau_{\mathrm{LCA}}(T_1, T_2) = \tau_{\mathrm{ACC}}(T_1, T_2)$ [9]. On the other hand, consider the trees T_1 and T_2 in Fig. 8. Then, we obtain the minimum cost mapping $M_1 \in \mathcal{M}_{\mathrm{ACCSG}}(T_1, T_2)$ and the minimum cost mapping $M_2 \in \mathcal{M}_{\mathrm{LCASG}}(T_1, T_2)$ under the unit cost function γ in Fig. 8. Hence, it holds that $\tau_{\mathrm{ACCSG}}(T_1, T_2) = \gamma(M_1) = 4 < 6 = \tau_{\mathrm{LCASG}}(T_1, T_2) = \gamma(M_2)$.

It is known that τ_{TAI}, τ_{ILST}, $\tau_{\mathrm{LCA}}(= \tau_{\mathrm{ACC}})$, τ_{TOP}, τ_{SG}, and τ_{BOT} are metrics, whereas τ_{ALN} is not [7, 9]. In the remainder of this section, we investigate whether or not $\tau_{\mathbb{A}}$ is a metric for the newly introduced mappings $\mathtt{A} \in$ {SGALN, ACCSG, LCASG, BOTALN, ACCBOT, LCABOT, LCART}.

Let $M_i \in \mathcal{M}_{\mathtt{A}}(T_i, T_{i+1})$ for $i = 1, 2$. Then, we define the *composition* $M_1 \circ M_2$ of mappings M_1 and M_2 as follows.

$$M_1 \circ M_2 = \left\{ \begin{array}{l} (u, w) \\ \quad \in V(T_1) \times V(T_3) \end{array} \middle| \begin{array}{l} \exists v \in V(T_2) \\ \text{ s.t. } (u, v) \in M_1 \text{ and } (v, w) \in M_2 \end{array} \right\}.$$

Lemma 2 ([7, 9]) *Let $M_i \in \mathcal{M}_{\mathtt{A}}(T_i, T_{i+1})$ for $i = 1, 2$, where:*

$$\mathtt{A} \in \{\textsc{Tai}, \textsc{Ilst}, \textsc{Acc}, \textsc{Lca}, \textsc{Top}, \textsc{Sg}, \textsc{Bot}\}.$$

Then, the following statements hold.

1. $M_1 \circ M_2 \in \mathcal{M}_{\mathtt{A}}(T_1, T_3)$.
2. *For a cost function γ that is a metric, it holds that $\gamma(M_1 \circ M_2) \leq \gamma(M_1) + \gamma(M_2)$.*

Lemma 3 *If both $\mathcal{M}_{\mathtt{A}}$ and $\mathcal{M}_{\mathtt{B}}$ satisfy the statements in Lemma 2, then so does $\mathcal{M}_{\mathtt{AB}}$.*

Proof Suppose that $M_i \in \mathcal{M}_{\mathtt{AB}}(T_i, T_{i+1})$ for $i = 1, 2$. By the definition that $\mathcal{M}_{\mathtt{AB}}(T_i, T_{i+1}) = \mathcal{M}_{\mathtt{A}}(T_i, T_{i+1}) \cap \mathcal{M}_{\mathtt{B}}(T_i, T_{i+1})$, it holds that $M_i \in \mathcal{M}_{\mathtt{A}}(T_i, T_{i+1})$ and $M_i \in \mathcal{M}_{\mathtt{B}}(T_i, T_{i+1})$. By Lemma 2.1 for $\mathcal{M}_{\mathtt{A}}$ and $\mathcal{M}_{\mathtt{B}}$, it holds that $M_1 \circ M_2 \in \mathcal{M}_{\mathtt{A}}(T_1, T_3)$ and $M_1 \circ M_2 \in \mathcal{M}_{\mathtt{B}}(T_1, T_3)$. Hence, it is obvious that $M_1 \circ M_2 \in \mathcal{M}_{\mathtt{AB}}(T_1, T_3)$, so Lemma 2.1 holds for $\mathcal{M}_{\mathtt{AB}}$. Since the proof of Lemma 2.2 just depends on Lemma 2.1, Lemma 2.2 also holds for $\mathcal{M}_{\mathtt{AB}}$. □

Theorem 3 (Metricity) *If a cost function is a metric, then so are $\tau_{\textsc{AccSg}}$, $\tau_{\textsc{LcaSg}}$, $\tau_{\textsc{AccBot}}$, $\tau_{\textsc{LcaBot}}$ and $\tau_{\textsc{LcaRt}}$. On the other hand, neither $\tau_{\textsc{SgAln}}$ nor $\tau_{\textsc{BotAln}}$ is a metric even if a cost function is a metric.*

Proof First, we show that $\tau_{\textsc{AccSg}}$ is a metric. To do this, it is sufficient to show that $\tau_{\textsc{AccSg}}(T_1, T_2) \geq 0$, $\tau_{\textsc{AccSg}}(T_1, T_2) = 0$ iff $T_1 \equiv T_2$, $\tau_{\textsc{AccSg}}(T_1, T_2) = \tau_{\textsc{AccSg}}(T_2, T_1)$ and $\tau_{\textsc{AccSg}}(T_1, T_3) \leq \tau_{\textsc{AccSg}}(T_1, T_2) + \tau_{\textsc{AccSg}}(T_2, T_3)$. The first three statements follow from the definition of $\mathcal{M}_{\textsc{AccSg}}(T_1, T_2)$. Let M_i be the minimum cost accordant segmental mapping between T_i and T_{i+1} ($i = 1, 2$). By Lemma 2 and 3, it holds that $\tau_{\textsc{AccSg}}(T_1, T_3) \leq \gamma(M_1 \circ M_2) \leq \gamma(M_1) + \gamma(M_2) = \tau_{\textsc{AccSg}}(T_1, T_2) + \tau_{\textsc{AccSg}}(T_2, T_3)$. By using the same proof, we can show that $\tau_{\textsc{LcaSg}}$, $\tau_{\textsc{AccBot}}$ and $\tau_{\textsc{LcaBot}}$ are metrics. Also it is obvious that $\tau_{\textsc{LcaRt}}$ is a metric, because $\tau_{\textsc{Lca}}$ is a metric.

On the other hand, in order to show that neither $\tau_{\textsc{SgAln}}$ nor $\tau_{\textsc{BotAln}}$ is a metric, consider the trees T_1, T_2 and T_3 in Fig. 9 and suppose that γ is a unit cost function. Then, we obtain the minimum cost mapping $M_{12} \in \mathcal{M}_{\textsc{BotAln}}(T_1, T_2)$, $M_{13} \in \mathcal{M}_{\textsc{BotAln}}(T_1, T_3)$ and $M_{23} \in \mathcal{M}_{\textsc{BotAln}}(T_2, T_3)$ under γ, respectively, in Fig. 9.

Hence, it holds that $8 = \gamma(M_{12}) = \tau_{\textsc{BotAln}}(T_1, T_2) > \tau_{\textsc{BotAln}}(T_1, T_3) + \tau_{\textsc{BotAln}}(T_2, T_3) = \gamma(M_{13}) + \gamma(M_{23}) = 3 + 3 = 6$. Since this statement holds

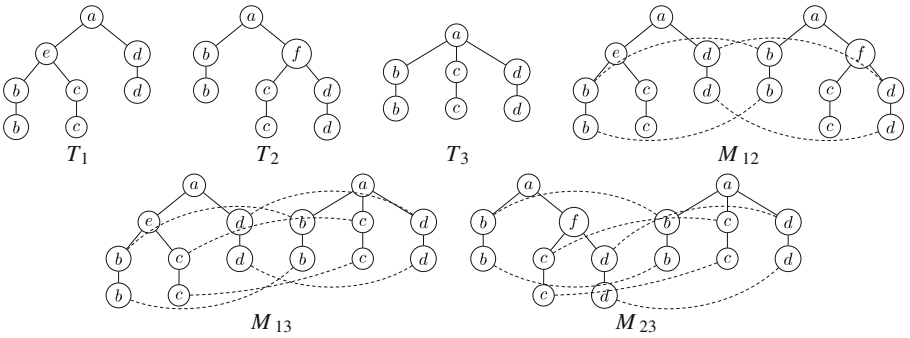

Fig. 9 Trees T_1, T_2 and T_3 and mappings M_{12}, M_{13} and M_{23} in the proof of Theorem 3

for τ_{SGALN}, neither τ_{SGALN} nor τ_{BOTALN} satisfies the triangle inequality, neither is a metric. □

4 Time Complexity of Computing Distances

In this section, we investigate the time complexity of computing the variations of the edit distance for both ordered and unordered trees. Here, we identify a node with its postorder number. In the remainder of this paper, let $n = |T_1|$, $m = |T_2|$, $D = \max\{d(T_1), d(T_2)\}$ and $d = \min\{d(T_1), d(T_2)\}$. For a mapping $\mathtt{A}$, we denote the distance between ordered and unordered trees by $\tau^o_{\mathtt{A}}$ and $\tau^u_{\mathtt{A}}$, respectively, whereas the distance between ordered and unordered forests by $\delta^o_{\mathtt{A}}$ and $\delta^u_{\mathtt{A}}$, respectively.

4.1 Tractable Cases

In this section, we investigate the tractable cases of computing the variations of the edit distance.

Remark 4 According to [7], for every pair $(i, j) \in T_1 \times T_2$ $(1 \le i \le n, 1 \le j \le m)$, we can compute the top-down distance $\tau^o_{\mathrm{TOP}}(T_1[i], T_2[j])$ between $T_1[i]$ and $T_2[j]$ in $O(nm)$ time [7].

On the other hand, we can compute $\tau^u_{\mathrm{TOP}}(T_1[i], T_2[j])$ by using the recurrences in Fig. 10 [18]. Here, $\mathtt{BM}$ is the maximum weighted bipartite matching with weight ω in a complete bipartite graph $G_M = (X, Y, E)$ such that X and Y are the sets

$$\tau^u_{\mathrm{TOP}}(T_1[i], T_2[j]) = \delta^u_{\mathrm{TOP}}(T_1(i), T_2(j)) + \gamma(i, j),$$
$$\delta^u_{\mathrm{TOP}}(T_1(i), T_2(j)) = \sum_{T \in T_1(i)} \sum_{i' \in T} \gamma(i', \varepsilon) + \sum_{T \in T_2(j)} \sum_{j' \in T} \gamma(\varepsilon, j') - \sum_{(i', j') \in \mathtt{BM}} \omega((i', j')).$$

Fig. 10 The recurrences of computing $\tau^u_{\mathrm{TOP}}(T_1, T_2)$ for unordered trees

of children of i in T_1 and j in T_2 and the weight of an edge $(i', j') \in E$ is set to $\tau^u_{\mathrm{TOP}}(T_1[i'], \emptyset) + \tau^u_{\mathrm{TOP}}(\emptyset, T_2[j']) - \tau^u_{\mathrm{TOP}}(T_1[i'], T_2[j'])$ [18, 22]. Hence, we can store $\tau^u_{\mathrm{TOP}}(T_1[i], T_2[j])$ for every $(i, j) \in T_1 \times T_2$ in $O(nmd)$ time.

Theorem 4 *We can compute $\tau^o_{\mathrm{LCART}}(T_1, T_2)$ in $O(nm)$ time and $\tau^u_{\mathrm{LCART}}(T_1, T_2)$ in $O(nmd)$ time.*

Proof We can design the recurrences of computing τ_{LCART} by adding

$$\tau_{\mathrm{LCART}}(T_1[r_1], T_2[r_2]) = \delta_{\mathrm{LCA}}(T_1(r_1), T_2(r_2)) + \gamma(r_1, r_2)$$

to the recurrences of computing τ_{LCA} [18, 22] (for both ordered and unordered trees), where r_i is the root of T_i $(i = 1, 2)$. Hence, we can compute $\tau^o_{\mathrm{LCART}}(T_1, T_2)$ in $O(nm)$ time and $\tau^u_{\mathrm{LCART}}(T_1, T_2)$ in $O(nmd)$ time. □

Theorem 5 *We can compute $\tau^o_{\mathrm{LCASG}}(T_1, T_2)$ in $O(nm)$ time and $\tau^u_{\mathrm{LCASG}}(T_1, T_2)$ in $O(nmd)$ time.*

Proof For every pair $(i, j) \in T_1 \times T_2$, we define the distance $d(T_1, i, T_2, j)$ between T_1 and T_2 based on $\tau^o_{\mathrm{TOP}}(T_1[i], T_2[j])$ as follows.

$$d(T_1, i, T_2, j) = \tau^o_{\mathrm{TOP}}(T_1[i], T_2[j]) + \sum_{i' \in T_1 - T_1[i]} \gamma(i', \varepsilon) + \sum_{j' \in T_2 - T_2[j]} \gamma(\varepsilon, j').$$

By Remark 4, it holds that

$$\tau^o_{\mathrm{LCASG}}(T_1, T_2) = \min\{d(T_1, i, T_2, j) \mid 1 \le i \le n, 1 \le j \le m\},$$

which we can compute in $O(nm)$ time. Also, by Remark 4, we can replace $\tau^o_{\mathrm{TOP}}(T_1[i], T_2[j])$ in $d(T_1, i, T_2, j)$ with $\tau^u_{\mathrm{TOP}}(T_1[i], T_2[j])$ for computing τ^u_{LCASG}, which runs in $O(nmd)$ time. □

Theorem 6 *We can compute $\tau^o_{\mathrm{ACCSG}}(T_1, T_2)$ in $O(nm)$ time and $\tau^o_{\mathrm{SGALN}}(T_1, T_2)$ in $O(nmD^2)$ time.*

Proof Consider the recurrences of computing τ^o_{ACCSG} and τ^o_{SGALN} in Fig. 11. The difference between the recurrences of computing τ^o_{ACCSG} and those of τ^o_{ACC} [9] is the first two formulas in τ^o_{ACCSG}, and the difference between the recurrences of computing τ^o_{SGALN} and those of τ^o_{ALN} [6] is the first formula in τ^o_{SGALN}, respectively. The first formula in τ^o_{ACCSG} and τ^o_{SGALN} computes the distance for segmentations of $(i, j) \in T_1 \times T_2$. In the second formula in τ^o_{ACCSG}, we replace $\gamma(i, j)$ in τ^o_{ACC} with $\gamma(i, \varepsilon) + \gamma(\varepsilon, j)$. Since we can compute them in constant time (after computing $\tau^o_{\mathrm{TOP}}(T_1[i], T_2[j])$ in advance according to Remark 4) and by the same discussion of [9] and [6], the statement holds. □

Theorem 7 *We can compute $\tau^u_{\mathrm{ACCSG}}(T_1, T_2)$ in $O(nmd)$ time.*

$$
\begin{aligned}
&\tau^o_{\mathrm{AccSG}}(T_1[i], T_2[j]) \\
&\quad = \min \left\{ \begin{array}{l} \tau^o_{\mathrm{TOP}}(T_1[i], T_2[j]) \cdots \text{use the value computed in advance}, \\ \delta^o_{\mathrm{AccSG}}(T_1(i), T_2(j)) + \gamma(i, \varepsilon) + \gamma(\varepsilon, j), \\ \tau^o_{\mathrm{AccSG}}(T_1[i], \emptyset) + \min\limits_{T \in T_1(i)} \{\tau^o_{\mathrm{AccSG}}(T, T_2[j]) - \tau^o_{\mathrm{AccSG}}(T, \emptyset)\}, \\ \tau^o_{\mathrm{AccSG}}(\emptyset, T_2[j]) + \min\limits_{T \in T_2(j)} \{\tau^o_{\mathrm{AccSG}}(T_1[i], T) - \tau^o_{\mathrm{AccSG}}(\emptyset, T)\} \end{array} \right\}, \\
&\delta^o_{\mathrm{AccSG}}(F_1(i_1, i_s), F_2(j_1, j_t)) \\
&\quad = \min \left\{ \begin{array}{l} \delta^o_{\mathrm{AccSG}}(F_1(i_1, i_{s-1}), F_2(j_1, j_t)) + \tau^o_{\mathrm{AccSG}}(T_1[i_s], \emptyset), \\ \delta^o_{\mathrm{AccSG}}(F_1(i_1, i_s), F_2(j_1, j_{t-1})) + \tau^o_{\mathrm{AccSG}}(\emptyset, T_2[j_t]), \\ \delta^o_{\mathrm{AccSG}}(F_1(i_1, i_{s-1}), F_2(j_1, j_{t-1})) + \tau^o_{\mathrm{AccSG}}(T_1[i_s], T_2[j_t]) \end{array} \right\}. \\
&\tau^o_{\mathrm{SGALN}}(T_1[i], T_2[j]) \\
&\quad = \min \left\{ \begin{array}{l} \tau^o_{\mathrm{TOP}}(T_1[i], T_2[j]) \cdots \text{use the value computed in advance}, \\ \delta^o_{\mathrm{SGALN}}(T_1(i), T_2(j)) + \gamma(i, j), \\ \tau^o_{\mathrm{SGALN}}(T_1[i], \emptyset) + \min\limits_{T \in T_1(i)} \{\tau^o_{\mathrm{SGALN}}(T, T_2[j]) - \tau^o_{\mathrm{SGALN}}(T, \emptyset)\}, \\ \tau^o_{\mathrm{SGALN}}(\emptyset, T_2[j]) + \min\limits_{T \in T_2(j)} \{\tau^o_{\mathrm{SGALN}}(T_1[i], T) - \tau^o_{\mathrm{SGALN}}(\emptyset, T)\} \end{array} \right\}, \\
&\delta^o_{\mathrm{SGALN}}(F_1(i_1, i_s), F_2(j_1, j_t)) \\
&\quad = \min \left\{ \begin{array}{l} \delta^o_{\mathrm{SGALN}}(F_1(i_1, i_{s-1}), F_2(j_1, j_t)) + \tau^o_{\mathrm{SGALN}}(T_1[i_s], \emptyset), \\ \delta^o_{\mathrm{SGALN}}(F_1(i_1, i_s), F_2(j_1, j_{t-1})) + \tau^o_{\mathrm{SGALN}}(\emptyset, T_2[j_t]), \\ \delta^o_{\mathrm{SGALN}}(F_1(i_1, i_{s-1}), F_2(j_1, j_{t-1})) + \tau^o_{\mathrm{SGALN}}(T_1[i_s], T_2[j_t]), \\ \gamma(i_s, \varepsilon) + \min\limits_{1 \le k < t} \left\{ \begin{array}{l} \delta^o_{\mathrm{SGALN}}(F_1(i_1, i_{s-1}), F_2(j_1, j_{k-1})) \\ + \delta^o_{\mathrm{SGALN}}(T_1(i_s), F_2(j_k, j_t)) \end{array} \right\}, \\ \gamma(\varepsilon, j_t) + \min\limits_{1 \le k < s} \left\{ \begin{array}{l} \delta^o_{\mathrm{SGALN}}(F_1(i_1, i_{k-1}), F_2(j_1, j_{t-1})) \\ + \delta^o_{\mathrm{SGALN}}(F_1(i_k, i_s), T_2(j_t)) \end{array} \right\} \end{array} \right\}.
\end{aligned}
$$

Fig. 11 The recurrences of computing $\tau^o_{\mathrm{AccSG}}(T_1, T_2)$ and $\tau^o_{\mathrm{SGALN}}(T_1, T_2)$ for ordered trees

Proof We can compute $\tau^u_{\mathrm{AccSG}}(T_1, T_2)$ in $O(nmd)$ time by using the recurrences in Fig. 12 after storing every $\tau^u_{\mathrm{TOP}}(T_1(i), T_2(j))$ $((i, j) \in T_1 \times T_2)$ in $O(nmd)$ time according to the recurrences in Fig. 10 of Remark 4. □

Next, we investigate the variations of the bottom-up distance τ_{BOT}.

Remark 5 We say that two trees T_1 and T_2 are *ordered* (*resp.*, *unordered*) *label-free isomorphic*, denoted by $T_1 \equiv^o_l T_2$ (*resp.*, $T_1 \equiv^u_l T_2$), if there exists a mapping $M \in \mathcal{M}^o_{\mathrm{TAI}}(T_1, T_2)$ (*resp.*, $M \in \mathcal{M}^u_{\mathrm{TAI}}(T_1, T_2)$), called a *label-free isomorphism*, such that $I_M = J_M = \emptyset$. Also we define $\mathit{diff}^r(T_1, T_2)$ ($r \in \{o, u\}$) as follows, where M is a

$$
\begin{aligned}
&\tau^u_{\mathrm{AccSG}}(T_1[i], T_2[j]) \\
&\quad = \min \left\{ \begin{array}{l} \tau^u_{\mathrm{TOP}}(T_1[i], T_2[j]), \\ \delta^u_{\mathrm{AccSG}}(T_1(i), T_2(j)) + \gamma(i, \varepsilon) + \gamma(\varepsilon, j), \\ \tau^u_{\mathrm{AccSG}}(T_1[i], \emptyset) + \min\limits_{T \in T_1(i)} \{\tau^u_{\mathrm{AccSG}}(T, T_2[j]) - \tau^u_{\mathrm{AccSG}}(T, \emptyset)\}, \\ \tau^u_{\mathrm{AccSG}}(\emptyset, T_2[j]) + \min\limits_{T \in T_2(j)} \{\tau^u_{\mathrm{AccSG}}(T_1[i], T) - \tau^u_{\mathrm{AccSG}}(\emptyset, T)\} \end{array} \right\}, \\
&\delta^u_{\mathrm{AccSG}}(T_1(i), T_2(j)) \\
&\quad = \min \left\{ \begin{array}{l} \delta^u_{\mathrm{AccSC}}(T_1(i), \emptyset) + \min\limits_{1 \le i' \le d(i)} \{\delta^u_{\mathrm{AccSC}}(T_1(i'), T_2(j)) - \delta^u_{\mathrm{AccSC}}(T_1(i'), \emptyset)\}, \\ \delta^u_{\mathrm{AccSC}}(\emptyset, T_2(j)) + \min\limits_{1 \le j' \le d(j)} \{\delta^u_{\mathrm{AccSC}}(T_1(i), T_2(j')) - \delta^u_{\mathrm{AccSC}}(\emptyset, T_2(j'))\}, \\ \delta^u_{\mathrm{TOP}}(T_1(i), T_2(j)) \end{array} \right\}.
\end{aligned}
$$

Fig. 12 The recurrences of computing $\tau^u_{\mathrm{AccSG}}(T_1, T_2)$ for unordered trees

label-free isomorphism between T_1 and T_2.

$$diff^r(T_1, T_2) = \begin{cases} \sum_{(i,j)\in M} \gamma(i, j) & \text{if } T_1 \equiv_l^r T_2, \\ \sum_{i\in T_1} \gamma(i, \varepsilon) + \sum_{j\in T_2} \gamma(\varepsilon, j) & \text{if } T_1 \not\equiv_l^r T_2. \end{cases}$$

We can check $T_1 \equiv_l^r T_2$ and compute $diff^r(T_1, T_2)$ in $O(n + m)$ time [15].

Theorem 8 *We can compute $\tau^o_{\mathrm{LCABOT}}(T_1, T_2)$ and $\tau^u_{\mathrm{LCABOT}}(T_1, T_2)$ in $O(nm)$ time.*

Proof For $r \in \{o, u\}$, by replacing $\tau^r_{\mathrm{TOP}}(T_1[i], T_2[j])$ in $d(T_1, i, T_2, j)$ in Theorem 5 with $diff^r(T_1[i], T_2[j])$ and by Remark 5, we can compute $\tau^r_{\mathrm{LCABOT}}(T_1, T_2)$ in $O(nm)$ time. □

Theorem 9 *We can compute $\tau^o_{\mathrm{ACCBOT}}(T_1, T_2)$ in $O(nm)$ time and $\tau^o_{\mathrm{BOTALN}}(T_1, T_2)$ in $O(nmD^2)$ time.*

Proof Remark 5 implies that we can compute τ^o_{ACCBOT} and τ^o_{BOTALN} by replacing $\tau^o_{\mathrm{TOP}}(T_1[i], T_2[j])$ in the recurrences of computing τ^o_{ACCSG} and τ^o_{SGALN} in Fig. 11 with $diff^o(T_1[i], T_2[j])$ in $O(nm)$ time and in $O(nmD^2)$ time, respectively. □

Theorem 10 *We can compute $\tau^u_{\mathrm{ACCBOT}}(T_1, T_2)$ in $O(nmd)$ time.*

Proof We extend the function $diff^u$ for unordered trees in Remark 5 to a function $fdiff^u$ for forests. Let $F_1 = [S_1, \ldots, S_k]$ and $F_2 = [T_1, \ldots, T_l]$ be forests. First, for $X = \{1, \ldots, k\}$ and $Y = \{1, \ldots, l\}$, we construct a weighted bipartite graph $G = (X, Y, E)$ such that $(i, j) \in E$ if and only if $S_i \equiv_l^u T_j$ and the weight $\omega((i, j)) = |S_i| + |T_j| - diff^u(S_i, T_j)$. Also, let $\mathtt{BM} \subseteq X \times Y$ be the maximum weighted bipartite matching of G, $\mathtt{BM}_X^- = \{i \in X \mid (i, j) \notin \mathtt{BM}\}$ and $\mathtt{BM}_Y^- = \{j \in Y \mid (i, j) \notin \mathtt{BM}\}$. Then, we define $fdiff^u(F_1, F_2)$ as follows.

$$fdiff^u(F_1, F_2) = \sum_{(i,j)\in \mathtt{BM}} diff^u(S_i, T_j) + \sum_{i\in \mathtt{BM}_X^-} |S_i| + \sum_{j\in \mathtt{BM}_X^-} |T_j|.$$

It is obvious that $fdiff^u(F_1, F_2)$ computes the smallest value when selecting pairs of label-free isomorphic trees in F_1 and F_2, which is $\delta^u_{\mathrm{ACCBOT}}(F_1, F_2)$. Hence, by replacing $\tau^u_{\mathrm{TOP}}(T_1[i], T_2[j])$ and $\delta^u_{\mathrm{TOP}}(T_1(i), T_2(j))$ in Fig. 12 with $diff^u(T_1[i], T_2[j])$ and $fdiff^u(T_1(i), T_2(j))$, we can compute $\tau^u_{\mathrm{ACCBOT}}(T_1, T_2)$ in $O(nmd)$ time. □

4.2 Intractable Cases

Suppose that Π_1 and Π_2 be two optimization problems. Then, we say that Π_1 *L-reduces* to Π_2 if there exist polynomial-time algorithms f, g and constants $\alpha, \beta > 0$ satisfying the following statements for an instance I of Π_1:

1. $opt(f(I)) \le \alpha \cdot opt(I)$.

2. For a solution of $f(I)$ with weight s_2, the algorithm g produces in polynomial time a solution of I with weight s_1 such that $|s_1 - opt(I)| \leq \beta \cdot |s_2 - opt(f(I))|$.

If Π_1 L-reduces Π_2, and Π_2 can be approximated in polynomial time within a factor $1+\varepsilon$, then Π_1 can be approximated within the factor $1+\alpha\beta\varepsilon$. If Π_2 has a polynomial time approximation scheme (PTAS), then so does Π_1 [12].

A problem is MAX SNP-*hard* if every problem in MAX SNP can be L-reduced to it. Since the composition of two L-reductions is also an L-reduction, a problem is MAX SNP-hard if a MAX SNP-hard problem can be L-reduced it. It is known that if any MAX SNP-hard problem has a PTAS, then P = NP. Hence, it is very unlikely for a MAX SNP-hard problem to have a PTAS [12].

Concerned with this paper, Zhang and Jiang [21] have first shown that the problem of computing $\tau^u_{\mathrm{TAI}}(T_1, T_2)$ is MAX SNP-hard. Akutsu et al. [1] and Hirata et al. [5] have shown that this problem is also MAX SNP-hard even if the height of T_1 and T_2 is at most 2 and T_1 and T_2 are binary trees, respectively. Furthermore, Yamamoto et al. [18] have shown that the problem of computing $\tau^u_{\mathrm{BOT}}(T_1, T_2)$ is MAX SNP-hard even if the degrees of T_1 and T_2 are binary, which implies the same MAX SNP-hardness of the problem of computing $\tau^u_{\mathrm{SG}}(T_1, T_2)$.

Note that we cannot apply this proof to showing that the problems of computing $\tau^u_{\mathrm{SGALN}}(T_1, T_2)$ and $\tau^u_{\mathrm{BOTALN}}(T_1, T_2)$ are MAX SNP-hard, with preserving alignable mappings. Hence, in this paper, we give the similar proof of [18] to show that the problems of computing $\tau^u_{\mathrm{SGALN}}(T_1, T_2)$ and $\tau^u_{\mathrm{BOTALN}}(T_1, T_2)$ are MAX SNP-hard, by using L-reduction from MAX 3SC-3:

> MAXIMUM BOUNDED COVERING BY 3-SETS (MAX 3SC-3) [8]:
> INSTANCE: A finite set S and a collection $\mathcal{C}$ of 3-element subsets of S, where every element of S occurs in at most three of the subsets in $\mathcal{C}$.
> SOLUTION: The largest covering of S, where a *covering* is a collection of mutually disjoint sets in $\mathcal{C}$.

Let $S = \{s_1, \ldots, s_m\}$, $\mathcal{C} = \{C_1, \ldots, C_n\}$ and $C_i = \{s_{i1}, s_{i2}, s_{i3}\}$ be an instance of MAX 3SC-3, where $s_{i1}, s_{i2}, s_{i3} \in S$. Also, let $\mathcal{C}^*$ be the largest covering of S. We assume that the cost function γ is a unit cost function.

We construct three trees $\hat{C}_i$ for C_i $(1 \leq i \leq n)$, $\hat{s}_j$ for s_j $(1 \leq j \leq m)$ and $\hat{D}$ illustrated in Fig. 13 (left), where λ is a new label. Then, we construct two trees T_1 and T_2 illustrated in Fig. 13 (right). We call the transformation from an instance of MAX 3SC-3 to trees T_1 and T_2 f. Then, we can show the following lemma as same as [18].

Lemma 4 *For* $\mathtt{A} \in \{\mathrm{SGALN}, \mathrm{BOTALN}\}$*, let M be the minimum cost mapping in $\mathcal{M}_{\mathtt{A}}(T_1, T_2)$ Then, for every $\hat{C}_i$ in T_1, M maps (a) all of the three subtree $\hat{s_{i1}}$, $\hat{s_{i2}}$ and $\hat{s_{i3}}$ in $\hat{C}_i$ to the same three subtrees in T_2 or (b) $\hat{C}_i$ to some dummy subtree $\hat{D}$ in T_2.*

Remark 6 Lemma 4 does not hold for a mapping in $\mathcal{M}_{\mathtt{A}}(T_1, T_2)$ such that $\mathtt{A} \in \{\mathrm{ILST}, \mathrm{ACC}, \mathrm{ACCSG}, \mathrm{ACCBOT}\}$. For i and i', suppose that $\hat{C}_i$ satisfies (a) and $\hat{C}_{i'}$ satisfies (b) in Lemma 4. Let v_{ij} (*resp.*, w_{ij}) denotes the leaf in T_1 (*resp.*, T_2) labeled by $\hat{s_{ij}}$ and w_λ denotes some leaf in T_2 labeled by λ.

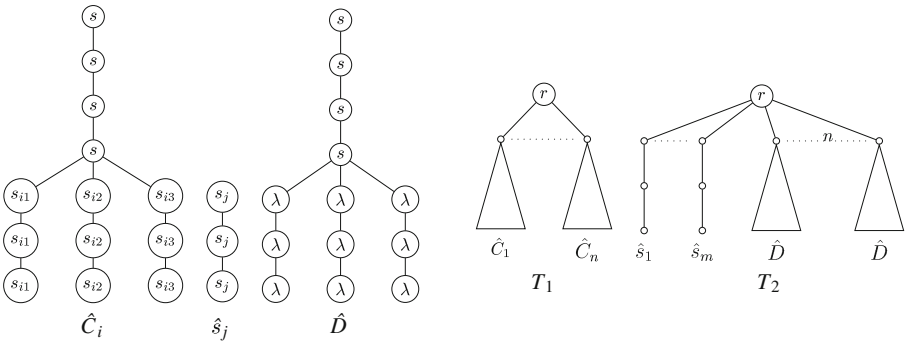

Fig. 13 Trees $\hat{C}_i$ ($1 \le i \le n$), $\hat{s}_j$ ($1 \le j \le m$) and $\hat{D}$ (*left*) and trees T_1 and T_2 (*right*)

Then, $M = \{(v_{i1}, w_{i1}), (v_{i2}, w_{i2}), (v_{i'1}, w_\lambda)\}$ is corresponding to the part of the mapping between T_1 and T_2 in Lemma 4. However, it holds that $w_\lambda < w_{i1} \sqcup w_{i2}$ but $v_{i'1} \not< v_{i1} \sqcup v_{i2}$. Hence, M is not an isolated-subtree mapping.

Lemma 5 *For* $\mathtt{A} \in \{\text{SGALN,BOTALN}\}$, $\tau^u_{\mathtt{A}}(T_1, T_2) = 9n + 3m - |\mathcal{C}^*| + 2$.

Proof Let M be the minimum cost mapping in $\mathcal{M}_{\mathtt{A}}(T_1, T_2)$, and $k = |\mathcal{C}^*|$.

By Lemma 4, for every $C_i \in \mathcal{C}^*$, M maps the three subtrees $\hat{s}_{i1}$, $\hat{s}_{i2}$ and $\hat{s}_{i3}$ in $\hat{C}_i$ to the same three subtrees in T_2. Since the cost of a mapping between $\hat{C}_i$ in T_1 and $\hat{s}_{i1}$, $\hat{s}_{i2}$ and $\hat{s}_{i3}$ in T_2 is 4 (which is the number of non-mapped nodes in $\hat{C}_i$ in T_1) and $|\mathcal{C}^*| = k$, the cost of M concerned with $\mathcal{C}^*$ is $4k$. Also, by Lemma 4, for every $C_i \in \mathcal{C} - \mathcal{C}^*$, M maps $\hat{C}_i$ to some dummy tree $\hat{D}$ in T_2. Since the cost of a mapping between $\hat{C}_i$ in T_1 and $\hat{D}$ in T_2 is 9 (which is the number of pairs of nodes labeled by s_{ij} ($j = 1, 2, 3$) in T_1 and λ in T_2) and $|\mathcal{C} - \mathcal{C}^*| = n - k$, the cost of M concerned with $\mathcal{C} - \mathcal{C}^*$ is $9(n - k)$. Since M does not touch $m - 3k$ subtrees $\hat{s}_j$ of s_j in T_2, $n - (n - k) = k$ dummy subtrees $\hat{D}$ in T_2 and 2 roots labeled by r, the concerned cost of M is $3(m - 3k) + 13k + 2$.

Hence, it holds that $\tau^u_{\mathtt{A}}(T_1, T_2) = \gamma(M) = 4k + 9(n-k) + 3(m-3k) + 13k + 2 = 9n + 3m - k + 2$. □

Theorem 11 *For* $\mathtt{A} \in \{\text{SGALN,BOTALN}\}$, *the problem of computing* $\tau^u_{\mathtt{A}}(T_1, T_2)$ *is MAX SNP-hard.*

Proof Consider the algorithm g to transform from a mapping $M \in \mathcal{M}_{\mathtt{A}}(T_1, T_2)$ between T_1 and T_2 to a covering $\mathcal{C}'$ of $\mathcal{C}$ as follows:

If M maps $\hat{s}_{i1}$, $\hat{s}_{i2}$ and $\hat{s}_{i3}$ of $\hat{C}_i$ in T_1 to the same three subtrees in T_2, then add C_i to $\mathcal{C}'$.

$$\delta^u_{\mathrm{SGALN}}(A,B) = \min \left\{ \begin{array}{l} \min\limits_{T_1[i'] \in A, T_2[j'] \in B} \left\{ \begin{array}{l} \delta^u_{\mathrm{SGALN}}(A - \{T_1[i']\}, B - \{T_2[j']\}) \\ \quad +\tau^u_{\mathrm{SGALN}}(T_1[i'], T_2[j']) \end{array} \right\}, \\ \min\limits_{T_1[i'] \in A, B' \subseteq B} \left\{ \begin{array}{l} \delta^u_{\mathrm{SGALN}}(A - \{T_1[i']\}, B - B') \\ \quad +\delta^u_{\mathrm{SGALN}}(T_1(i'), B') + \gamma(i', \varepsilon) \end{array} \right\}, \\ \min\limits_{A' \subseteq A, T_2[j_t] \in B} \left\{ \begin{array}{l} \delta^u_{\mathrm{SGALN}}(A - A', B - \{T_2[j']\}) \\ \quad +\delta^u_{\mathrm{SGALN}}(A', T_2(j')) + \gamma(\varepsilon, j') \end{array} \right\} \end{array} \right\}.$$

Fig. 14 The recurrences of computing $\delta^u_{\mathrm{SGALN}}$ for unordered trees with bounded degrees

It holds that $n/7 \leq opt(I)$ from the proof of Theorem 7 in [21]. Then, by Lemma 5 and since $m \leq 3n$ and $opt(I) \geq 1$, the following inequalities hold.

$$\begin{aligned} opt(f(I)) &= 9n + 3m - opt(I) + 2 \leq 18n - opt(I) + 2 \leq 127 \cdot opt(I), \\ s_2 - opt(f(I)) &= \gamma(M) - \tau^u_{\mathrm{A}}(T_1, T_2) \\ &\geq 9n + 3m - |\mathcal{C}'| + 2 - (9n + 3m - opt(I) + 2) \\ &= opt(I) - |\mathcal{C}'| = opt(I) - s_1. \end{aligned}$$

Hence, (f, g) is an L-reduction from MAX 3SC-3 to this problem. □

Whereas the problem of computing $\tau^u_{\mathrm{ALN}}(T_1, T_2)$ is MAX-SNP hard, it is tractable if the degrees of trees are bounded by some constant [6]. As same as $\tau^u_{\mathrm{ALN}}(T_1, T_2)$, the following theorem also holds.

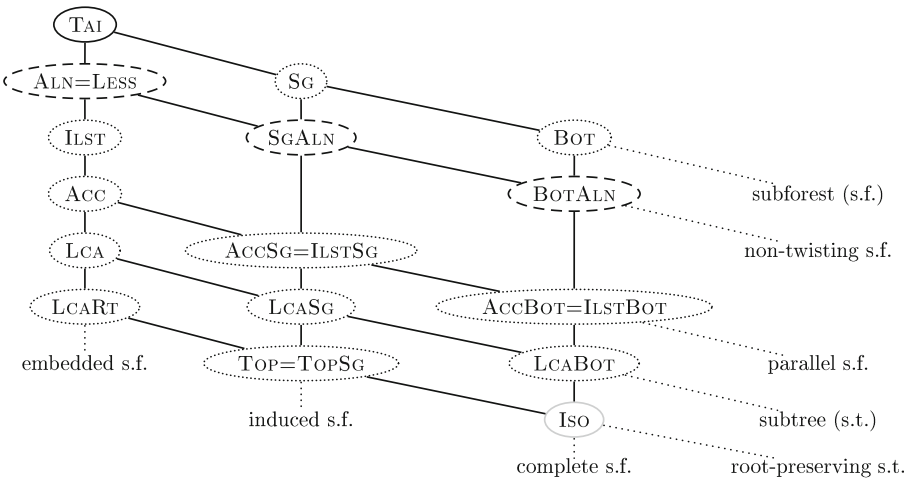

Fig. 15 The time complexity of computing $\tau^o_{\mathrm{A}}(T_1, T_2)$ for $\mathcal{M}_{\mathrm{A}}(T_1, T_2)$. Here, the nodes enclosed by *black solid lines, black dashed lines, black dotted lines* and *gray solid lines* illustrate that the problem of computing $\tau^o_{\mathrm{A}}(T_1, T_2)$ is $O(nm^2(1 + \log \frac{n}{m}))$ time, $O(nmD^2)$ time, $O(nm)$ time and $O(n + m)$ time, respectively

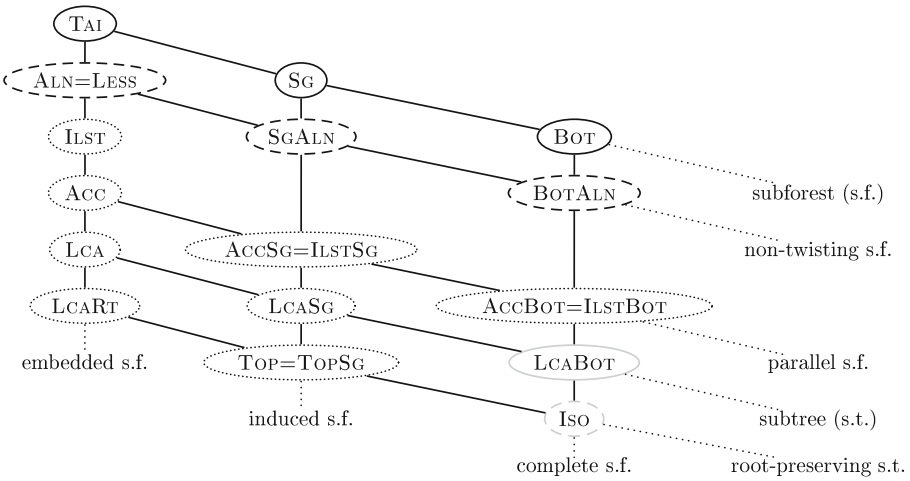

Fig. 16 The time complexity of computing $\tau^u_{\mathbb{A}}(T_1, T_2)$ for $\mathcal{M}_{\mathbb{A}}(T_1, T_2)$. Here, the nodes enclosed by *black solid lines, black dashed lines, black dotted lines, gray solid lines* and *gray dashed lines* illustrate that the problem of computing $\tau^o_{\mathbb{A}}(T_1, T_2)$ is MAX SNP-hard even if T_1 and T_2 are binary, MAX SNP-hard but tractable if the degrees of T_1 and T_2 are bounded, $O(nmd)$ time, $O(nm)$ time and $O(n + m)$ time, respectively

Theorem 12 *The problems of computing* $\tau^u_{\text{SGALN}}(T_1, T_2)$ *and* $\tau^u_{\text{BOTALN}}(T_1, T_2)$ *are tractable if the degrees of* T_1 *and* T_2 *are bounded by some constant.*

Proof We can design the algorithm to compute τ^u_{SGALN} by replacing τ^o_{TOP} in τ^o_{SGALN} in Fig. 11 with τ^u_{TOP} in Fig. 10, and by improving the recurrences of computing δ^o_{SGALN} in Fig. 11 to the recurrences in Fig. 14, which is same as the improvement of the recurrences from computing δ^o_{ALN} to computing δ^u_{ALN} with bounded degrees [6], where $A \subseteq T_1(i)$ and $B \subseteq T_2(j)$. Since the degrees of T_1 and T_2 are bounded by some constant, the number of combinations of A and B are bounded, so $\delta^u_{\text{SGALN}}(A, B)$ can be computed in polynomial time.

Furthermore, by replacing $diff^o$ in the recurrences of computing τ^o_{BOTALN} (*cf.*, Theorem 9) with $diff^u$ and by using the same discussion as above, we can design the recurrences of computing δ^u_{BOTALN} in polynomial time. □

5 Conclusion

In this paper, we have characterized a Tai mapping hierarchy as several common subforests, that is, as an embedded subforest, an induced subforest and a complete subforest by focusing on the connections of nodes in a common subforest and as a non-twisting subforest, a parallel subforest, a subtree and a root-preserving subtree by focusing on the arrangements of subtrees in a common subforest. Then, we have introduced new mappings into the Tai mapping hierarchy illustrated in Fig. 2.

Next, we have investigated the metricity of the variations of the edit distance as the minimum cost of the above mappings and the time complexity of computing them. We summarize the results as Table 1 in Section 1. Also Figs. 15 and 16 illustrate the relationship between the Tai mapping hierarchy and time complexity of computing $\tau_{\mathbb{A}}(T_1, T_2)$ for $\mathcal{M}_{\mathbb{A}}(T_1, T_2)$ as diagram forms.

Acknowledgments The authors would like to thank Prof. Tetsuji Kuboyama at Gakushuin University, Prof. Kilho Shin at University of Hyogo and Prof. Tetsuhiro Miyahara at Hiroshima City University for fruitful discussion about tree edit distance and its variations. Also they would like to thank anonymous referees of Theory of Computing Systems for valuable comments to revise the submitted version of this paper.

References

1. Akutsu, T., Fukagawa, D., Halldórsson, M.M., Takasu, A., Tanaka, K.: Approximation and parameterized algorithms for common subtrees and edit distance between unordered trees. Theor. Comput. Sci. **470**, 10–22 (2013)
2. Bille, P.: A survey on tree edit distance and related problems. Theor. Comput. Sci. **337**, 217–239 (2005)
3. Chawathe, S.S.: Comparing hierarchical data in external memory. Proc. VLDB'99, 90–101 (1999)
4. Demaine, E.D., Mozes, S., Rossman, B., Weiman, O.: An optimal decomposition algorithm for tree edit distance. ACM Trans. Algorithms, 6 (2009)
5. Hirata, K., Yamamoto, Y., Kuboyama, T.: Improved MAX SNP-hard results for finding an edit distance between unordered trees. Proc. CPM 2011. LNCS **6661**, 402–415 (2011)
6. Jiang, T., Wang, L., Zhang, K.: Alignment of trees – an alternative to tree edit. Theor. Comput. Sci. **143**, 137–148 (1995)
7. Kan, T., Higuchi, S., Hirata, K.: Segmental mapping and distance for rooted ordered labeled trees. Fundamenta Informaticae **132**, 1–23 (2014)
8. Kann, V.: Maximum bounded 3-dimensional matching is MAX SNP-complete. Inf. Process. Lett. **37**, 27–35 (1991)
9. Kuboyama, T.: Matching and Learning in Trees. Ph.D thesis, University of Tokyo (2007)
10. Lu, C.L., Su, Z.-Y., Yang, C.Y.: A new measure of edit distance between labeled trees, Proc. COCOON'01. LNCS **2108**, 338–348 (2001)
11. Lu, S.-Y.: A tree-to-tree distance and its application to cluster analysis. IEEE Trans. Pattern Anal. Mach. Intell. **1**, 219–224 (1979)
12. Papadimitoriou, C.H., Yannakakis, M.: Optimization, approximation and complexity. J. Comput. System Sci. **43**, 425–440 (1991)
13. Selkow, S.M.: The tree-to-tree editing problem. Inform. Process. Lett. **6**, 184–186 (1977)
14. Tai, K.-C.: The tree-to-tree correction problem. J. ACM **26**, 422–433 (1979)
15. Valiente, G.: An efficient bottom-up distance between trees. Proc. SPIRE'01, 212–219 (2001)
16. Valiente, G.: Algorithms on Trees and Graphs. Springer (2002)
17. Wang, J.T.L., Zhang, K.: Finding similar consensus between trees: an algorithm and a distance hierarchy. Pattern Recogn. **34**, 127–137 (2001)
18. Yamamoto, Y., Hirara, K., Kuboyama, T.: Tractable and intractable variations of unordered tree edit distance. Int. J. Found. Comput. Sci. **25**, 307–330 (2014)
19. Zhang, K.: Algorithms for the constrained editing distance between ordered labeled trees and related problems. Pattern Recog. **28**, 463–474 (1995)
20. Zhang, K.: A constrained edit distance between unordered labeled trees. Algorithmica **15**, 205–222 (1996)
21. Zhang, K., Jiang, T.: Some MAX SNP-hard results concerning unordered labeled trees. Inf. Process. Lett. **49**, 249–254 (1994)
22. Zhang, K., Wang, J., Shasha, D.: On the editing distance between undirected acyclic graphs. Int. J. Found. Comput. Sci. **7**, 43–58 (1996)