Theoretical Computer Science 410 (2009) 837–846

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

A constrained edit distance algorithm between semi-ordered trees

Aïda Ouangraoua *, Pascal Ferraro

LaBRI - Université de Bordeaux 1, 351 Cours de la Libération, 33405 Talence Cedex, France

ARTICLE INFO

Article history:
Received 19 February 2008
Received in revised form 17 September 2008
Accepted 17 November 2008
Communicated by M. Crochemore

Keywords:
Semi-ordered trees
Theory of computation
Tree editing
Dynamic programming

ABSTRACT

In this paper, we propose a formal definition of a new class of trees called *semi-ordered trees* and a polynomial dynamic programming algorithm to compute a constrained edit distance between such trees. The core of the method relies on a similar approach to compare unordered [Kaizhong Zhang, A constrained edit distance between unordered labeled trees, Algorithmica 15 (1996) 205–222] and ordered trees [Kaizhong Zhang, Algorithms for the constrained editing distance between ordered labeled trees and related problems, Pattern Recognition 28 (3) (1995) 463–474]. The method is currently applied to evaluate the similarity between architectures of apple trees [Vincent Segura, Aida Ouangraoua, Pascal Ferraro, Evelyne Costes, Comparison of tree architecture using tree edit distances: Application to two-year-old apple tree, Euphytica 161 (2007) 155–164].

1. Introduction

Edit distances, initially introduced for string-to-string comparison [1], were later extended to compare ordered trees [2, 3] and unordered trees [4]. Ordered trees are trees in which the left-to-right order of each node's children is fixed [2] while unordered trees are trees in which no order is considered on the set of children of any vertex.

In this context, the tree-to-tree edition problem consists in determining a sequence of edit operations (substitutions, insertions and deletions of vertices) of minimum cost needed to transform one tree into the other, the cost of a sequence being the sum of the costs of its edit operations. The edit operations are constrained in order to preserve the topology of trees after their application. Basically, in an optimal sequence of edit operations no more than one edit operation can be applied on any vertex of a tree and the edit operations must maintain the ancestor–descendant relation between vertices. Furthermore, in ordered tree comparisons, an additional constraint is added to maintain the left-to-right order of each node's children [2] while in the case of unordered trees an additional constraint preserves the descendants of the nearest common ancestor of two vertices [4].

In this paper, we propose unifying these two approaches by introducing a new class of trees called *semi-ordered trees*. A *semi-ordered tree* is a tree with a semi-order relation defined on the set of children of each vertex. Thus, ordered and unordered trees can be considered as semi-ordered trees using an appropriate definition of the semi-order relations between children of vertices. Finally, we propose an algorithm to compute a constrained edit distance between two semi-ordered trees using the dynamic programming principle.

The introduction of *semi-ordered trees* is motivated by a biological application. Indeed, in most botanical applications, trees are used to represent the topological structure of plant architectures [5] and then to evaluate the similarity between these architectures using edit distances [6]. Generally, the topological structure of plant architectures can be modeled by ordered trees. However, the order between the components of a plant is often partially measured and in some cases, only semi-order relations instead of order relations are measured between the components. In these cases, plant architectures can be modeled by semi-ordered trees.

* Corresponding author. Tel.: +33 5 40 00 35 10.
E-mail addresses: ouangrao@labri.fr (A. Ouangraoua), ferraro@labri.fr (P. Ferraro).

0304-3975/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2008.11.022

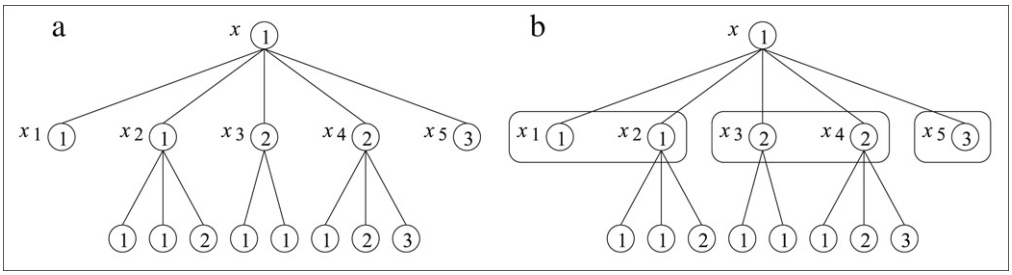

Fig. 1. (a) A semi-ordered tree T, (b) the ordered partition of the children of the vertex x induced by the total semi-order relation $\sqsubseteq_x$ on $C(x)$.

In Section 2, we formally define a semi-ordered tree. In Section 3, we first define the notion of *mapping between two semi-ordered trees*, and then a polynomial algorithm for the computation of a constrained edit distance between semi-ordered trees is described.

2. Semi-ordered trees

A directed graph $G = (V, E)$ consists of a set V of vertices, a set E of edges, each edge being represented by an ordered pair of vertices. The order (i.e. number of vertices) of a graph $G = (V, E)$ is denoted by $|G|$ and the number of vertices in a subset of vertices $X \subseteq V$ is denoted by $|X|$. If G is a directed graph then for any edge (x, y) in E, x and y are called a *parent-child* pair. A *rooted tree* is then an acyclic, connected, directed graph $T = (V, E)$ in which every vertex except one (the root r) has exactly one parent vertex: the root has no parent. The *parent* of a vertex x is denoted by $P(x)$ and the set of *children* of x is denoted by $C(x)$. A path between two vertices x and y is a (possibly empty) sequence of edges $\{(x_i, x_{i+1})_{i=1,n}\}$ such that $x_1 = x$ and $x_{n+1} = y$. A vertex y is then a *descendant* of a vertex x (and reciprocally x is an *ancestor* of y) if there exists a path between x and y. The ancestor–descendant relation on the set of vertices of a rooted tree is a partial order relation on the set of its vertices denoted by $\leq$: by $x \leq y$ we mean that x is an ancestor of y. The *nearest common ancestor* of two vertices x and y of T, denoted by $x \wedge y$, is the common ancestor of x and y such that for any common ancestor z of x and y, $z \leq (x \wedge y)$. In the following, $T[x]$ denotes the subtree of T rooted at x which contains all the descendants of x, and the empty tree and the empty graph are both denoted by $\emptyset = (\emptyset, \emptyset)$.

A *forest* is a set of rooted trees. The set of the roots of the trees composing a forest F is denoted by roots(F). Let T be a rooted tree. Let x be a vertex of T. Let $X = \{x_1, \ldots, x_n\}$ be a subset of $C(x)$. $F[X]$ denotes the subforest of T consisting of subtrees rooted at vertices of X, $F[X] = \{T[x_1], \ldots, T[x_n]\}$. Thus, the subforest of a vertex x denoted by $F[C(x)]$ is obtained from $T[x]$ by removing the root x and all edges incident with x. $F[C(x)]$ is simply denoted by $F[x]$.

An ordered tree is a pair $(T, S_{\preceq})$ where T is a rooted tree and $S_{\preceq} = \{\preceq_x,\ x \in T\}$ is a set of total order relations such that for any vertex x of T, $\preceq_x$ is a total order relation on $C(x)$. The set of total order relations $S_{\preceq}$ and the ancestor–descendant relation $\leq$ on the vertices of T induce two total order relations on the set of vertices of T (namely *prefix* or *postfix* order relations). An unordered tree is a rooted tree T such that the only significant relation between the vertices of T is the ancestor–descendant relation (all the children of a vertex of T are equivalent). We propose unifying these two classes of trees in a single one called semi-ordered trees (Fig. 1.a). A *semi-order relation* [7] $\sqsubseteq$ on a set V is a reflexive and transitive binary relation on V. $\sqsubseteq$ is a *total semi-order relation* if for any pair (x, y) of elements of V, $x \sqsubseteq y$ or $y \sqsubseteq x$, otherwise $\sqsubseteq$ is called *partial semi-order relation*. In Fig. 1, the total semi-order relation on the set of children of a given vertex in a tree is defined by the relation "less than or equal to" on the numbers associated to the vertices.

Definition 1 (*Semi-ordered Tree*). A semi-ordered tree is a pair $(T, S_{\sqsubseteq})$ where T is a rooted tree and $S_{\sqsubseteq} = \{\sqsubseteq_x,\ x \in T\}$ is a set of total semi-order relations such that for any vertex x of T, $\sqsubseteq_x$ is a total semi-order relation on $C(x)$.

Note that an unordered tree T is a semi-ordered tree $(T, S_{\sqsubseteq})$ such that for any vertex x of T, $\sqsubseteq_x$ is an equivalence relation having only one equivalence class, *i.e.* for any pair (x_1, x_2) in $C(x) \times C(x)$, $x_1 \sqsubseteq_x x_2$ and $x_2 \sqsubseteq_x x_1$ (all the children of x are equivalent). Similarly, an ordered tree T is a semi-ordered tree $(T, S_{\sqsubseteq})$ such that for any vertex x of T, $\sqsubseteq_x$ is a total order relation on $C(x)$. A semi-order (or pre-order) relation on $C(x)$ means that some elements of $C(x)$ are equivalent (equal) while some others can be totally ordered. If there are some elements which are not comparable then the semi-order is partial otherwise it is a total semi-order relation.

Let $(T, S_{\sqsubseteq})$ be a semi-ordered tree. The set of semi-order relations $S_{\sqsubseteq}$ and the ancestor–descendant relation $\leq$ on the vertices of T induce an order relation denoted by $\sqsubseteq$ on the set of vertices of T defined as follows. Let us consider two vertices x and y. x will be smaller than y according to $\sqsubseteq$ if and only if x is an ancestor of y or there exists two ancestor s and t of respectively x and y, having the same parent u such that s is strictly smaller than t according to $\sqsubseteq_u$.

Formally, for any vertex u of T and $s, t \in C(u)$, the relation $s \sqsubseteq_u t$ and $t \not\sqsubseteq_u s$ is denoted by $s \sqsubset_u t$. The order relation $\sqsubseteq$ is such that for any pair (x, y) of vertices of T, x and y satisfy $x \sqsubseteq y$ if and only if x is an ancestor of y or there is a pair (s, t) of vertices in $C(x \wedge y)$ such that $s \leq x$ and $t \leq y$ and $s \sqsubset_{x \wedge y} t$:

$$\forall x, y \in V, \quad x \sqsubseteq y \Leftrightarrow \begin{cases} x \leq y \text{ or,} \\ \exists\, s, t \in C(x \wedge y) \mid s \leq x \text{ and } t \leq y \text{ and } s \sqsubset_{x \wedge y} t. \end{cases}$$

Note that if T is an unordered tree then $\sqsubseteq$ is simply the ancestor–descendant relation and if T is an ordered tree then $\sqsubseteq$ is a total order relation on V (namely the prefix order relation), in any other case $\sqsubseteq$ is a partial order relation.

Let V be a set and $\sqsubseteq$ a total semi-order relation on V. $\sqsubseteq$ induces an *equivalence relation* (*i.e.* reflexive, transitive and symmetric) on V denoted by $\equiv$ and defined by:

$$\forall x, y \in V, \quad x \equiv y \Leftrightarrow x \sqsubseteq y \quad \text{and} \quad y \sqsubseteq x.$$

The *equivalence class* of an element x in V is denoted by $[x] = \{y \in V \mid x \equiv y\}$ and the set of equivalence classes of the elements of V (defining a partition of V) is denoted by $V_{\equiv} = \{[x] \mid x \in V\}$. In the same way, the total semi-order relation $\sqsubseteq$ induces a *total order relation* $\preceq$ (*i.e.* reflexive, transitive and antisymmetric) on $V_{\equiv}$ defined by:

$$\forall [x], [y] \in V_{\equiv}, \quad [x] \preceq [y] \Leftrightarrow x \sqsubseteq y.$$

For any vertex x of T, the *partition* of $C(x)$ induced by $\sqsubseteq_x$ is denoted by $C(x)_{\equiv}$ (Fig. 1.b). For any set X in $C(x)_{\equiv}$, the *left-class* of X denoted by X^- is the set of all vertices x_i in $C(x)$ such that for any vertex x_j in X, $x_i \sqsubset x_j$ and the *augmented-left-class* of X denoted by X^+ is defined as $X^+ = X \cup X^-$. The maximal set X of $C(x)_{\equiv}$ for the order relation defined on $C(x)_{\equiv}$ is such that $C(x) = X^+$, then the subforest of x is the subforest of T consisting of subtrees rooted at vertices of X^+, $F[x] = F[X^+]$.

For example, in Fig. 1.b, the ordered partition of $C(x)$ is $C(x)_{\equiv} = \{\{x_1, x_2\}, \{x_3, x_4\}, \{x_5\}\}$. If the set X is $\{x_1, x_2\}$ (respectively, $\{x_3, x_4\}$; $\{x_5\}$) then the left-class of X is $X^- = \emptyset$ (respectively, $X^- = \{x_1, x_2\}$; $X^- = \{x_1, x_2, x_3, x_4\}$) and the augmented-left-class of X is $X^+ = \{x_1, x_2\}$ (respectively, $X^+ = \{x_1, x_2, x_3, x_4\}$; $X^+ = C(x)$).

A *semi-ordered forest* is a pair $(F, \sqsubseteq_F)$ where F is a set of semi-ordered trees and $\sqsubseteq_F$ is a total semi-order relation on roots(F). Note that, for any set X in $C(x)_{\equiv}$, $(F[X], \sqsubseteq_x)$, $(F[X^-], \sqsubseteq_x)$ and $(F[X^+], \sqsubseteq_x)$ are semi-ordered forests since $F[X]$, $F[X^-]$ and $F[X^+]$ are sets of semi-ordered trees and $\sqsubseteq_x$ is a total semi-order relation on roots$(F[X]) = X$, roots$(F[X^-]) = X^-$ and roots$(F[X^+]) = X^+$.

In the following, trees are labeled on an alphabet Σ and α is a function that associates a label of Σ to each vertex of a tree $T = (V, E)$, $\alpha : V \to \Sigma$. A distance d is assumed to be defined on the labels of Σ. A distance[1] between vertices is defined using d: $\gamma(x, y) = d(\alpha(x), \alpha(y))$. Let λ be a symbol not in Σ. d is extended by defining quantities $d(\alpha(x), \lambda)$ and $d(\lambda, \alpha(y))$ such that d is a distance on $\Sigma \cup \{\lambda\}$. The distance $d(\alpha(x), \lambda)$ between the label of a vertex x and the label λ is denoted by $\gamma(x, \lambda)$ by convention, and similarly for $\gamma(\lambda, y)$.

In the following, $(T_1, S_{\sqsubseteq_1})$ and $(T_2, S_{\sqsubseteq_2})$ are two semi-ordered trees. If there is no confusion, $(T_1, S_{\sqsubseteq_1})$ and $(T_2, S_{\sqsubseteq_2})$ are respectively denoted by T_1 and T_2, $\sqsubseteq_1$ and $\sqsubseteq_2$ are the partial order relations respectively defined on the set of vertices of T_1 and T_2. They are simply denoted by $\sqsubseteq$ if there is no confusion.

3. Edit distance

The computation of an edit distance between two trees consists in determining a sequence of edit operations of minimum cost which transforms an initial tree into a target tree. In order to characterize the effect of a sequence of edit operations between two trees, edit distance mappings between vertices of trees has been introduced in [4] for unordered and [2,3] for ordered trees.

3.1. Mapping between semi-ordered trees

An optimal sequence of edit operations (*i.e.* a sequence having a minimum cost) from T_1 to T_2 is such that the corresponding mapping associates a vertex of T_1 to at most one vertex of T_2 and reciprocally and preserves the relations defined on the vertices of T_1 and T_2, namely the ancestor–descendant relation and the semi-order relations. The preservation of the ancestor–descendant relationship $\leq$ and the semi-order relations $\sqsubseteq_x$ is equivalent to the preservation of $\leq$ and the order relation $\sqsubseteq$ induced on the vertices of T_1 and T_2.

A *valid mapping* from $T_1 = (V_1, E_1)$ to $T_2 = (V_2, E_2)$ is a set M of ordered pairs of vertices (s, s') with $s \in V_1$ and $s' \in V_2$ such that for any pairs (s, s') and (t, t') in M:

$s = t \Leftrightarrow s' = t'$ (one-to-one)
$s \leq t \Leftrightarrow s' \leq t'$ (ancestor–descendant preservation)
$s \sqsubseteq t \Leftrightarrow s' \sqsubseteq t'$ (semi-order preservation)

For any pair (s, s') in M, s and s' are called images of each other and are denoted by $M(s) = s'$ and $M(s') = s$. If there is no confusion, $(s, s') \in M$ is denoted by $s \in M$ and $s' \in M$. Let F_1 (respectively, F_2) be a subforest of T_1 (respectively, T_2). The

[1] A distance d on a set V is a function that associates to each pair of elements of V a non-negative real such that for any x, y, z in V, $d(x, y) = 0$ if and only if $x = y$, $d(x, y) = d(y, x)$ and $d(x, y) + d(y, z) \geq d(x, z)$.

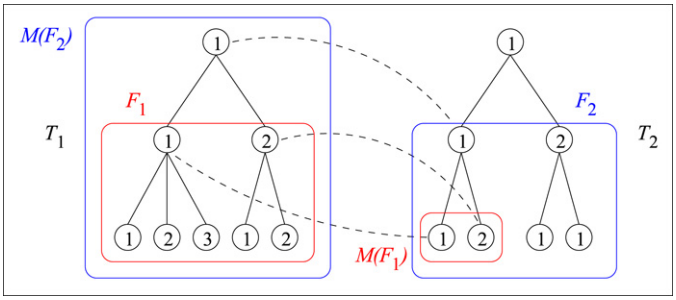

Fig. 2. A valid mapping from a semi-ordered tree T_1 to a semi-ordered tree T_2 and the images $M(F_1)$ and $M(F_2)$ of subforests F_1 and F_2 of T_1 and T_2 respectively. The vertices of T_1 and T_2 images of each other are linked by dotted lines.

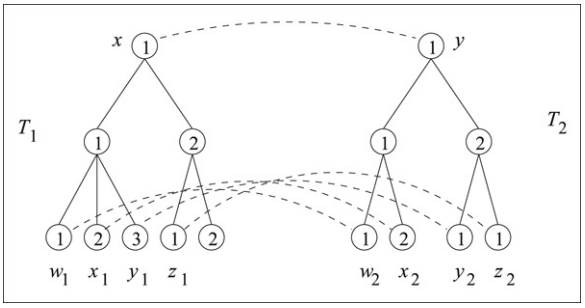

Fig. 3. Examples of mappings between two semi-ordered trees T_1 and T_2. $M = \{(x, y), (w_1, w_2), (x_1, x_2), (y_1, y_2), (z_1, z_2)\}$ is a not valid mapping from T_1 to T_2: $y_1 \sqsubseteq z_1$ whereas $y_2 \not\sqsubseteq z_2$. $M\backslash\{(z_1, z_2)\}$ is a valid but not constrained mapping: $(w_1 \wedge x_1) \leq y_1$ whereas $(w_2 \wedge x_2) \not\leq y_2$. $M\backslash\{(y_1, y_2), (z_1, z_2)\}$ is a constrained mapping from T_1 to T_2.

image of F_1 (respectively, F_2) denoted by $M(F_1)$ (respectively, $M(F_2)$) is the subforest of T_2 (respectively, T_1) containing all the images of vertices of F_1 (respectively, F_2) and their descendants. The concept of mapping is extended to forests and sets of vertices not necessarily connected.

Let M be a valid mapping from T_1 to T_2 (Fig. 2). Let I (respectively, J) be the set of vertices of T_1 (respectively, T_2) which do not appear in a pair of M. The cost of M is defined by:

$$\gamma(M) = \sum_{(s,s')\in M} \gamma(s, s') + \sum_{s\in I} \gamma(s, \lambda) + \sum_{s'\in J} \gamma(\lambda, s').$$

The *edit distance* between T_1 and T_2 is the minimum cost of a valid mapping from T_1 to T_2. Since unordered trees are particular cases of semi-ordered trees, computing the edit distance between two semi-ordered trees is at least as hard as the computation of the edit distance between unordered trees which is a MAX-SNP-hard problem [8].

3.2. Constrained edit distance

Since the computation of the edit distance between two semi-ordered trees is a MAX-SNP-hard problem, we consider a constrained version of the problem: the computation on semi-ordered trees of the *constrained edit distance* introduced by Zhang [4] to compare unordered trees.

A *constrained mapping* from T_1 to T_2 is a valid mapping (M, T_1, T_2) such that for any pairs (s, s'), (t, t') and (u, u') in M:

$$(s \wedge t) \leq u \Leftrightarrow (s' \wedge t') \leq u' \text{(structure preservation)}$$

Examples of mappings between semi-ordered trees are presented in Fig. 3. Let x (respectively, y) be a vertex of T_1 (respectively, T_2). Let $X = \{x_1, \ldots, x_n\}$ (respectively, $Y = \{y_1, \ldots, y_m\}$) be a subset of $C(x)$ (respectively, $C(y)$). The set of constrained mappings from $T_1[x]$ to $T_2[y]$ (respectively, $F_1[X]$ to $F_2[Y]$) is denoted by $\mathcal{M}_C(T_1[x], T_2[y])$ (respectively, $\mathcal{M}_C(F_1[X], F_2[Y])$).

Definition 2 (*Constrained Edit Distance*). The constrained edit distance between T_1 and T_2 is the minimum cost of a constrained mapping from T_1 to T_2:

$$D_C(T_1, T_2) = \min\{\gamma(M) \mid M \in \mathcal{M}_C(T_1, T_2)\}.$$

Zhang [4] has described a dynamic programming algorithm to compute the constrained edit distance between two unordered trees using reductions of some subproblems to *minimum cost maximum flow problems* [9]. Similarly, we propose in the following a reduction of the computation of the constrained edit distance between semi-ordered trees to minimum cost maximum flow problems. The following recurrence formulas form the basis of the algorithm for the computation of the constrained edit distance between two semi-ordered trees.

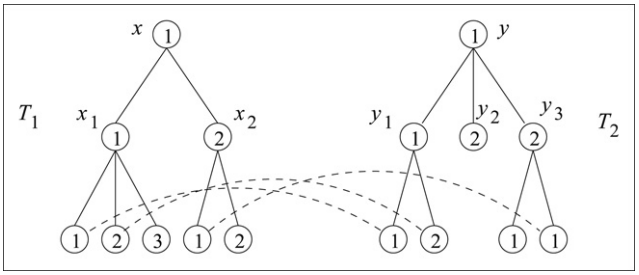

Fig. 4. A restricted mapping from $F_1[\mathrm{C}(x)]$ to $F_2[\mathrm{C}(y)]$ such that $\mathcal{P}(M) = \{(x_1, y_1), (x_2, y_3)\}$.

Lemma 3 (*Initialization*). *Let x, y be two vertices of T_1 and T_2 respectively. X and Y are two sets of vertices respectively in $C(x)_{\equiv}$ and $C(y)_{\equiv}$:*

$$\begin{cases} D_C(\emptyset, \emptyset) = 0 \\ D_C(T_1[x], \emptyset) = \gamma(x, \lambda) + D_C(F_1[x], \emptyset) \\ D_C(F_1[X], \emptyset) = \sum_{x_k \in X} D_C(T_1[x_k], \emptyset) \\ D_C(F_1[X^+], \emptyset) = D_C(F_1[X], \emptyset) + D_C(F_1[X^-], \emptyset) \\ D_C(\emptyset, T_2[y]) = \gamma(\lambda, y) + D_C(\emptyset, F_2[y]) \\ D_C(\emptyset, F_2[Y]) = \sum_{y_k \in Y} D_C(\emptyset, T_2[y_k]) \\ D_C(\emptyset, F_2[Y^+]) = D_C(\emptyset, F_2[Y]) + D_C(\emptyset, F_2[Y^-]). \end{cases}$$

Proof. The two first formulas are obvious. Since the forest $F_1[X]$ is composed of subtrees rooted at vertices of X, $D_C(F_1[X], \emptyset)$ is equal to the sum of the values $D_C(T_1[x_k], \emptyset)$ such that x_k is a vertex in X. Since $X^+ = X \cup X^-$, $D_C(F_1[X^+], \emptyset)$ is equal to $D_C(F_1[X], \emptyset)$ plus $D_C(F_1[X^-], \emptyset)$. The proofs of the last three formulas are symmetric to the previous proofs. □

Lemma 4 (*Constrained Edit Distance Between Subtrees*). *Let x and y be two vertices of T_1 and T_2 respectively. The constrained edit distance between $T_1[x]$ and $T_2[y]$ is:*

$$D_C(T_1[x], T_2[y]) = \min \begin{cases} D_C(F_1[x], F_2[y]) + \gamma(x, y) \\ \min_{y_k \in C(y)} \{D_C(T_1[x], T_2[y_k]) - D_C(\emptyset, T_2[y_k])\} + D_C(\emptyset, T_2[y]) \\ \min_{x_k \in C(x)} \{D_C(T_1[x_k], T_2[y]) - D_C(T_1[x_k], \emptyset)\} + D_C(T_1[x], \emptyset). \end{cases}$$

The proof of Lemma 4 (in Appendix) is similar to the proof of Lemma 6 in [4]. Indeed, only the preservation of ancestor–descendant relations is needed to compute $D_C(T_1[x], T_2[y])$.

However, to compute $D_C(T_1[x], T_2[y])$, we first need to compute $D_C(F_1[x], F_2[y])$. The computation of $D_C(F_1[x], F_2[y])$ uses the notion of *restricted mapping* introduced by Zhang [4] using a different formalism. Let $X = \{x_1, \ldots, x_n\}$ (respectively, $Y = \{y_1, \ldots, y_m\}$) be a subset of $\mathrm{C}(x)$ (respectively, $\mathrm{C}(y)$). A *restricted mapping* from $F_1[X]$ to $F_2[Y]$ is a constrained mapping M from $F_1[X]$ to $F_2[Y]$ such that for any pairs (s, s') and (t, t') in M (Fig. 4):

$$(s \wedge t) \in F_1[X] \;\Leftrightarrow\; (s' \wedge t') \in F_2[Y]$$

This means that each tree of $F_1[X]$ is mapped on at most one tree of $F_2[Y]$ and reciprocally. Thus, M induces a valid mapping from X to Y denoted by $\mathcal{P}(M)$ (Fig. 4) and defined as follows:

$$\mathcal{P}(M) = \{(x_i, y_j) \in X \times Y \mid M(T_1[x_i]) \subseteq M(T_2[y_j]) \quad \text{and} \quad M(T_2[y_j]) \subseteq M(T_1[x_i])\}.$$

The set of restricted mappings from $F_1[X]$ to $F_2[Y]$ is denoted by $\mathcal{R}(F_1[X], F_2[Y])$.

Lemma 5 (*Constrained Edit Distance Between Subforests*). *Let x and y be two vertices of T_1 and T_2 respectively. The constrained edit distance between $F_1[x]$ and $F_2[y]$ is:*

$$D_C(F_1[x], F_2[y]) = \min \begin{cases} \min\{\gamma(M), M \in \mathcal{R}(F_1[x], F_2[y])\} \\ \min_{y_k \in C(y)} \{D_C(F_1[x], F_2[y_k]) - D_C(\emptyset, F_2[y_k])\} + D_C(\emptyset, F_2[y]) \\ \min_{x_k \in C(x)} \{D_C(F_1[x_k], F_2[y]) - D_C(F_1[x_k], \emptyset)\} + D_C(F_1[x], \emptyset). \end{cases}$$

The proof of Lemma 5 (in Appendix) is only based on the preservation of the ancestor–descendant relation and the structure. It is then similar to the proof of Lemma 7 in [4].

Finally, the computation of $D_C(F_1[x], F_2[y])$ leads us to the computation of an optimal restricted mapping between $F_1[x]$ and $F_2[y]$. Let P be a valid mapping from $\mathrm{C}(x)$ to $\mathrm{C}(y)$. Let I (respectively, J) be the set of vertices of $\mathrm{C}(x)$ (respectively, $\mathrm{C}(y)$) which do not appear in a pair of P. We denote by $\gamma^*(P)$ the value:

$$\gamma^*(P) = \sum_{(x_i, y_j) \in P} D_C(T_1[x_i], T_2[y_j]) + \sum_{x_i \in I} D_C(T_1[x_i], \emptyset) + \sum_{y_j \in J} D_C(\emptyset, T_2[y_j]).$$

Let M be an optimal restricted mapping from $F_1[x]$ to $F_2[y]$. Since M has a minimum cost, its cost is $\gamma^*(\mathcal{P}(M))$. Thus, a way of computing the minimum cost of a restricted mapping in $\mathcal{R}(F_1[x], F_2[y])$ is finding a valid mapping P from the set of children of x (*i.e.* $C(x)$) to the set of children of y (*i.e.* $C(y)$) such that $\gamma^*(P)$ is minimum. In the case of unordered trees, the sets $C(x)$ and $C(y)$ are unordered therefore if $|C(x)| = |C(y)|$, finding a valid mapping P from $C(x)$ to $C(y)$ such that $\gamma^*(P)$ is minimum is equivalent to the computation of a bipartite matching of minimum weight in the weighted bipartite graph $(C(x) \cup C(y), C(x) \times C(y))$ such that the weight of an edge $(x_i, y_j) \in C(x) \times C(y)$ is $D_C(T_1[x_i], T_2[y_j])$. More generally, Zhang [4] reduces the computation of P to a minimum cost maximum flow problem [9]. In the current case, P is a mapping that conserves the semi-order relation defined on $C(x)$ and $C(y)$ and the problem cannot be immediately reduced to a minimum cost maximum flow problem. Nevertheless, P can be partitioned into mappings between forests whose sets of roots are unordered and thus, the computations of the costs of these mappings are reduced to minimum cost maximum flow problems. The minimum cost of a restricted mapping from a semi-ordered forest F_1 to a semi-ordered forest F_2 is denoted by $D_R(F_1, F_2)$:

$$D_R(F_1, F_2) = \min\{\gamma(M), M \in \mathcal{R}(F_1, F_2)\}.$$

For any vertices x and y of T_1 and T_2 respectively, since the set X (respectively, Y) of $C(x)_{\equiv}$ (respectively, $C(y)_{\equiv}$) which is maximal for the order relation defined on $C(x)_{\equiv}$ (respectively, $C(y)_{\equiv}$) is such that $F_1[x] = F_1[X^+]$ (respectively, $F_2[y] = F_2[Y^+]$), $D_R(F_1[x], F_2[y]) = D_R(F_1[X^+], F_2[Y^+])$.

Theorem 6 (*Minimum Cost of a Restricted Mapping*). *Let x, y be two vertices of T_1 and T_2 respectively. X and Y are two sets of vertices respectively in $C(x)_{\equiv}$ and $C(y)_{\equiv}$. The minimum cost of a restricted mapping from $F_1[X^+]$ to $F_2[Y^+]$ is:*

$$D_R(F_1[X^+], F_2[Y^+]) = \min \begin{cases} D_R(F_1[X], F_2[Y]) + D_R(F_1[X^-], F_2[Y^-]) \\ D_R(\emptyset, F_2[Y]) + D_R(F_1[X^+], F_2[Y^-]) \\ D_R(F_1[X], \emptyset) + D_R(F_1[X^-], F_2[Y^+]). \end{cases}$$

Proof. Let M be a restricted mapping from $F_1[X^+]$ to $F_2[Y^+]$. We consider 4 cases according to whether $M(F_1[X]) = \emptyset$ or not and $M(F_2[Y]) = \emptyset$ or not:

- If $M(F_1[X] \neq \emptyset)$ and $M(F_2[Y] \neq \emptyset)$, since M is a valid mapping (conservation of the partial order relation), for any vertex u_1 in $F_1[X]$ (respectively, u_2 in $F_2[Y]$) which appears in M, $M(u_1) \in F_2[Y]$ (respectively, $M(u_2) \in F_1[X]$). Then M is such that:
  - $M(F_1[X]) \subseteq F_2[Y]$ and $M(F_2[Y]) \subseteq F_1[X]$ and,
  - $M(F_1[X^-]) \subseteq F_2[Y^-]$ and $M(F_2[Y^-]) \subseteq F_1[X^-]$.

  The cost of M is equal to the cost of a restricted mapping from $F_1[X]$ to $F_2[Y]$ plus the cost of a restricted mapping from $F_1[X^-]$ to $F_2[Y^-]$ and since M has a minimum cost, its cost is:

  $$\gamma(M) = D_R(F_1[X], F_2[Y]) + D_R(F_1[X^-], F_2[Y^-]).$$

- If $M(F_1[X] \neq \emptyset)$ and $M(F_2[Y]) = \emptyset$ then since M is of minimum cost, its cost is:

  $$\gamma(M) = D_R(\emptyset, F_2[Y]) + D_R(F_1[X^+], F_2[Y^-]).$$

- If $M(F_1[X]) = \emptyset$ and $M(F_2[Y] \neq \emptyset)$ then this case is symmetric to the previous one and the cost of M is:

  $$\gamma(M) = D_R(F_1[X], \emptyset) + D_R(F_1[X^-], F_2[Y^+]).$$

- If $M(F_1[X]) = \emptyset$ and $M(F_2[Y]) = \emptyset$ then $M_2 = M \cup M_1$ where M_1 is a non-empty restricted mapping from $F_1[X]$ to $F_2[Y]$ is such that $M_2 \in \mathcal{R}(F_1[X^+], F_2[Y^+])$ and $\gamma(M_2) \leq \gamma(M)$ (since γ satisfies the triangle inequality), the cost of M is then greater than or equal to the minimum cost computed in the first case.

The minimum cost of a mapping in $\mathcal{R}(F_1[X^+], F_2[Y^+])$ is then the minimum of the minimum costs of mappings in the first three cases. □

In the formula for the computation of $D_R(F_1[X^+], F_2[Y^+])$ (Lemma 6), $D_R(F_1[X], F_2[Y])$ is the only part of the equation that is not of the form $D_R(F_1[Z^+], F_2[T^+])$ for some (Z, T) smaller than (X, Y). Then, to compute $D_R(F_1[X^+], F_2[Y^+])$, we first need to compute $D_R(F_1[X], F_2[Y])$ which can be reduced to a minimum cost maximum flow problem as for the minimum cost of a restricted mapping between unordered forests [4] using the notions of *network* and *flow*.

A *network* is a quintuple $G = (V, E, c, s, t)$ such that:

- (V, E) is a directed graph,
- c is a function that associates to each ordered pair (u, v) in $V \times V$ its *capacity*, a non-negative real $c(u, v)$ and such that $(u, v) \notin E \Rightarrow c(u, v) = 0$,
- s is a single vertex in V which has no parent and is called the source,
- t is a single vertex in V which has no child and is called the sink.

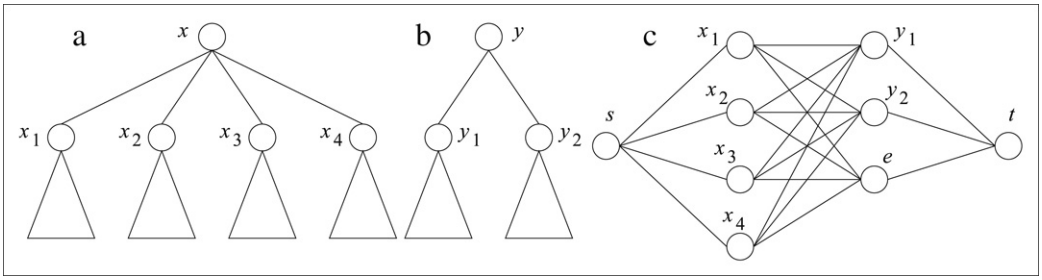

Fig. 5. (a) A tree $T_1[x]$, (b) a tree $T_2[y]$ and *(c)* the flow network $R(\mathrm{C}(x), \mathrm{C}(y))$: all the edges are directed from the left to the right and for any edge (u, v), $c(u, v) = 1$ except $c(e, t) = 2$.

A *flow* in a network $G = (V, E, c, s, t)$ is a function f that associates to each ordered pair (u, v) in $V \times V$ a real such that:

- $f(u, v) \leq c(u, v)$,
- $f(u, v) = -f(v, u)$,
- $\sum_{w \in V} f(u, w) = 0 \; \forall u \neq s, t$.

The *value of a flow* f in G denoted by $|f|$ is $|f| = \sum_{v \in V} f(s, v)$. If d is a function that associates to each (u, v) in E a real cost $d(u, v)$, the cost of the flow f denoted by $d(f)$ is $d(f) = \sum_{(u,v) \in E} d(u, v) \cdot f(u, v)$. Let $G = (V, E, c, s, t)$ be a flow network, d a cost function defined on E. The minimum cost maximum flow problem in G consists in computing a flow f in G such that $|f|$ is maximum and $d(f)$ is minimum.

For any $x \in T_1, X \subseteq \mathrm{C}(x), y \in T_2, Y \subseteq \mathrm{C}(y)$, $R(X, Y)$ denotes the flow network $G = (V, E, c, s, t)$ such that (Fig. 5):

- If $|X| = |Y|$
  - $V = \{s, t\} \cup X \cup Y$ with s and t the source and the sink of G respectively.
  - $E = (\{s\} \times X) \cup (X \times Y) \cup (Y \times \{t\})$.
  - for any (u, v) in E, $c(u, v) = 1$.
- If $|X| > |Y|$
  - $V = \{s, t, e\} \cup X \cup Y$ with s and t the source and the sink of G respectively.
  - $E = (\{s\} \times X) \cup (X \times (Y \cup \{e\})) \cup ((Y \cup \{e\}) \times \{t\})$.
  - for any (u, v) in $E \backslash \{(e, t)\}$, $c(u, v) = 1$ and $c(e, t) = |X| - |Y|$.
- If $|X| < |Y|$
  - $V = \{s, t, e\} \cup X \cup Y$ with s and t the source and the sink of G respectively.
  - $E = (\{s\} \times (X \cup \{e\})) \cup ((X \cup \{e\}) \times Y) \cup (Y \times \{t\})$.
  - for any (u, v) in $E \backslash \{(s, e)\}$, $c(u, v) = 1$ and $c(s, e) = |Y| - |X|$.

Lemma 7 (*Computation of D_R*). *Let x, y be two vertices of T_1 and T_2 respectively. X and Y are two sets of vertices respectively in $\mathrm{C}(x)_{\equiv}$ and $\mathrm{C}(y)_{\equiv}$. The minimum cost of a restricted mapping from $F_1[X]$ to $F_2[Y]$, $D_R(F_1[X], F_2[Y])$ is computed as the minimum cost of a maximum flow in the flow network $R(X, Y)$ with the costs:*

- *If $|X| = |Y|$ then $\forall (x_i, y_j) \in X \times Y,\ d(x_i, y_j) = D_C(T_1[x_i], T_2[y_j])$ and for any other edge (u, v), $d(u, v) = 0$,*
- *If $|X| > |Y|$ then $\forall (x_i, y_j) \in X \times Y,\ d(x_i, y_j) = D_C(T_1[x_i], T_2[y_j])$, $\forall x_i \in X,\ d(x_i, e) = D_C(T_1[x_i], \emptyset)$ and for any other edge (u, v), $d(u, v) = 0$,*
- *If $|X| < |Y|$ then $\forall (x_i, y_j) \in X \times Y,\ d(x_i, y_j) = D_C(T_1[x_i], T_2[y_j])$, $\forall y_j \in Y,\ d(e, y_j) = D_C(\emptyset, T_2[y_j])$ and for any other edge (u, v), $d(u, v) = 0$.*

Proof. The computation of $D_R(F_1[X], F_2[Y])$ is equivalent to find a valid mapping P from X to Y such that $\gamma^*(P)$ is minimum with:

$$\gamma^*(P) = \sum_{(x_i, y_j) \in P} D_C(T_1[x_i], T_2[y_j]) + \sum_{x_i \in I} D_C(T_1[x_i], \emptyset) + \sum_{y_j \in J} D_C(\emptyset, T_2[y_j])$$

where I (respectively, J) is the set of vertices of X (respectively, Y) which do not appear in a pair of P. If $I \neq \emptyset$ and $J \neq \emptyset$ then for any (x_i, y_j) in $I \times J$, $P_2 = P \cup \{(x_i, y_j)\}$ is such that $\gamma^*(P_2) \leq \gamma^*(P)$. Then the minimum cost of a restricted mapping from $F_1[X]$ to $F_2[Y]$ is the minimum value $\gamma^*(P)$ for a valid mapping P from X to Y such that $|I| = 0$ or $|J| = 0$. Thus if $|X| = |Y|$ (respectively, $|X| > |Y|$; $|X| < |Y|$) then $|I| = 0$ and $|J| = 0$ (respectively, $|I| = |X| - |Y|$ and $|J| = 0$; $|I| = 0$ and $|J| = |Y| - |X|$). Since X and Y are unordered sets and the maximum value of a flow in $R(X, Y)$ is $f^* = \max\{|X|, |Y|\}$, $\gamma^*(P)$ is exactly the minimum cost of a maximum flow in $R(X, Y)$ with the defined cost. □

Using Lemmas 3, 4, 5, 7 and Theorem 6, Algorithm 3.2 describes the computation of $D_C(T_1, T_2)$. In the algorithm, the vertices of T_1 (respectively, T_2) are numbered from 1 to $|T_1|$ (respectively, $|T_2|$) according to a postfix order (*i.e.* induced by a post-order traversal of each of the trees) and for any vertex $x \in T_1$ (respectively, $y \in T_2$), the sets of $\mathrm{C}(x)_{\equiv}$ (respectively, $\mathrm{C}(y)_{\equiv}$) are numbered from 1 to $|\mathrm{C}(x)_{\equiv}|$ (respectively, $|\mathrm{C}(y)_{\equiv}|$) according the order relation defined on $\mathrm{C}(x)_{\equiv}$ (respectively,

$C(y)_{\equiv}$). The validity of the algorithm 3.2 follows from six points:

(1) Lemmas 3, 4, 5, 7 and Theorem 6 are valid,
(2) The dynamic programming tables are first initialized,
(3) For any $(x, y) \in T_1 \times T_2$, $D_C(T_1[x], T_2[y])$ is computed after $D_C(F_1[x], F_2[y])$, all $D_C(F_1[x_k], F_2[y])$ such that $x_k \in C(x)$ and all $D_C(F_1[x], F_2[y_k])$ such that $y_k \in C(y)$,
(4) For any $(x, y) \in T_1 \times T_2$, $D_C(F_1[x], F_2[y])$ is computed after $D_R(F_1[x], F_2[y])$, all $D_C(F_1[x_k], F_2[y])$ such that $x_k \in C(x)$ and all $D_C(F_1[x], F_2[y_k])$ such that $y_k \in C(y)$,
(5) For any $(X, Y) \in C(x)_{\equiv} \times C(y)_{\equiv}$, $D_R(F_1[X^+], F_2[Y^+])$ is computed after distances $D_R(F_1[X], F_2[Y])$, $D_R(F_1[X^-], F_2[Y^-])$, $D_R(F_1[X^+], F_2[Y^-])$ and $D_R(F_1[X^-], F_2[Y^+])$,
(6) For any $(X, Y) \in C(x)_{\equiv} \times C(y)_{\equiv}$, $D_R(F_1[X], F_2[Y])$ is computed after all distances $D_C(T_1[x_i], T_2[y_j])$ such that $(x_i, y_j) \in C(x) \times C(y)$.

Algorithm 3.2.
Begin
initialize dynamic programming tables using Lemma 3
For $x = 1 \rightarrow |T_1|$ **Do**
  For $y = 1 \rightarrow |T_2|$ **Do**
    For $X = 1 \rightarrow |C(x)_{\equiv}|$ **Do**
      For $Y = 1 \rightarrow |C(y)_{\equiv}|$ **Do**
        compute $D_R(F_1[X], F_2[Y])$ using Lemma 7
        compute $D_R(F_1[X^+], F_2[Y^+])$ using Theorem 6
    compute $D_C(F_1[x], F_2[y])$ using Lemma 5
    compute $D_C(T_1[x], T_2[y])$ using Lemma 4
Output: $D_C(T_1[|T_1|], T_2[|T_2|])$
End

The complexity of the computation of $D_C(T_1, T_2)$ is due to the computations of $D_R(F_1[X], F_2[Y])$ for all $x \in T_1$, $X \in C(x)_{\equiv}$, $y \in T_2$, $Y \in C(y)_{\equiv}$, which are computed by solving minimum cost maximum flow problems [9]. For any vertices x and y of T_1 and T_2 respectively, X in $C(x)_{\equiv}$ and Y in $C(y)_{\equiv}$, an algorithm presented by Tarjan [9] allows computing the minimum cost of a maximum flow in $R(X, Y)$ in time $O(|X| \times |Y| \times (|X| + |Y|) \times \log_2(|X| + |Y|))$. For any i in $\{1, 2\}$, we set $deg(T_i) = \max_{x \in T_i}\{deg(x)\}$ with $deg(x) = |C(x)|$. The complexity in time of the computation of $D_C(T_1, T_2)$ using Algorithm 3.2 is then in:

$$
\begin{aligned}
&O\left(\sum_{x\in T_1}\sum_{y\in T_2}\sum_{X\in C(x)_{\equiv}}\sum_{Y\in C(y)_{\equiv}} |X| \times |Y| \times (|X| + |Y|) \times \log_2(|X| + |Y|)\right).\\
&\leq O\left(\sum_{x\in T_1}\sum_{y\in T_2}\sum_{X\in C(x)_{\equiv}} |X| \times \sum_{Y\in C(y)_{\equiv}} |Y| \times (deg(T_1) + deg(T_2)) \log_2(deg(T_1) + deg(T_2))\right).\\
&\leq O\left(\sum_{x\in T_1} |C(x)| \times \sum_{y\in T_2} |C(y)| \times (deg(T_1) + deg(T_2)) \times \log_2(deg(T_1) + deg(T_2))\right).\\
&\leq O(|T_1| \times |T_2| \times (deg(T_1) + deg(T_2)) \times \log_2(deg(T_1) + deg(T_2))).
\end{aligned}
$$

Finally, the computation of $D_C(T_1, T_2)$ has then a complexity in time bounded by $O(|T_1| \times |T_2| \times (deg(T_1) + deg(T_2)) \times \log_2(deg(T_1) + deg(T_2)))$.

Concerning the space complexity of Algorithm 3.2, for any vertices x and y of T_1 and T_2 respectively and any sets of vertices X and Y respectively in $C(x)_{\equiv}$ and $C(y)_{\equiv}$, the minimum costs of restricted mappings between subforests ($D_R(F_1[X], F_2[Y])$ and $D_R(F_1[X^+], F_2[Y^+])$) can be temporarily stored until the computation of $D_C(F_1[x], F_2[y])$. The temporary array used to store these values requires space $O(deg(T_1) \times deg(T_2))$. Then, the algorithm for the computation of $D_C(T_1, T_2)$ only requires a permanent array to memorize the intermediate computed distances between subforests ($D_C(F_1[x], F_2[y])$) and between subtrees ($D_C(T_1[x], T_2[y])$). This permanent array requires space $O(|T_1| \times |T_2|)$. The space complexity of the algorithm is then bounded by $O(|T_1| \times |T_2|)$.

If T_1 and T_2 are ordered trees (respectively, unordered trees), then Algorithm 3.2 has exactly the same steps (*i.e.* series of instructions) and results as the algorithm proposed by [3] (respectively, [4]) to compute the constrained edit distance between two ordered trees (respectively, unordered trees).

4. Conclusion

In this paper, we have formally defined semi-ordered trees and proposed an edit distance between such trees. The algorithm to compute the constrained edit distance between two semi-ordered trees uses the dynamic programming principle and works in polynomial time.

This work is part of a project to develop computer tools for comparing plant architectures [10] and the algorithm has been implemented in VPlants, a software dedicated to plant architecture analysis [11]. From an algorithmical point of view, the consideration of the constrained edit distance (using constrained and restricted mapping) for the comparison of plant architectures has allowed us to study a tractable problem (the computation of the constrained edit distance) but it is also relevant from a biological viewpoint. Indeed, in the plant a branching system is generated by a same meristem.[2] Then, it is normal for this branching system to be mapped on a ramified system generated by a single meristem.

This tool opens new perspectives for the comparison of plant architectures. Extensions of the algorithm to multiscale trees and local edition [12] are currently studied. The definition of distances between plant architectures highlights some general aspects concerning plant modeling. These methods are for instance useful and essential to improve plant simulation techniques.

Acknowledgement

This work was supported by the research project grant ANR-06-BLANC-0045 (BRASERO) from the national agency for research.

Appendix. Proofs of Lemmas 4 and 5

Proof of Lemma 4. We consider a partition of $\mathcal{M}_C(T_1[x], T_2[y])$ in 4 subsets according to whether $M(T_1[x]) \subseteq F_2[y]$ or not and $M(T_2[y]) \subseteq F_1[x]$ or not. Let M be a minimum cost mapping in $\mathcal{M}_C(T_1[x], T_2[y])$.

- If $M(T_1[x]) \subseteq F_2[y]$ and $M(T_2[y]) \not\subseteq F_1[x]$ then $x \in M$ and $y \notin M$, then there is a vertex $y_k \in \mathrm{C}(y)$ such that $M(x) \in T_2[y_k]$ and since M is a valid mapping (preservation of the ancestor–descendant relation) then $M(T_1[x]) \subseteq T_2[y_k]$. Since M is a minimum cost mapping, its cost is:
$$\gamma(M) = \min_{y_k \in \mathrm{C}(y)} \{D_C(T_1[x], T_2[y_k]) - D_C(\emptyset, T_2[y_k])\} + D_C(\emptyset, T_2[y]).$$
- If $M(T_1[x]) \not\subseteq F_2[y]$ and $M(T_2[y]) \subseteq F_1[x]$, then this case is symmetric to the previous one:
$$\gamma(M) = \min_{x_k \in \mathrm{C}(x)} \{D_C(T_1[x_k], T_2[y]) - D_C(T_1[x_k], \emptyset)\} + D_C(T_1[x], \emptyset).$$
- If $M(T_1[x]) \not\subseteq F_2[y]$ and $M(T_2[y]) \not\subseteq F_1[x]$ then $x \in M$ and $y \in M$, then $(x, y) \in M$ since M is a valid mapping (preservation of the ancestor–descendant relation). Since M is a minimum cost mapping, its cost is:
$$\gamma(M) = D_C(F_1[x], F_2[y]) + \gamma(x, y).$$
- If $M(T_1[x]) \subseteq F_2[y]$ and $M(T_2[y]) \subseteq F_1[x]$ then $x \notin M$ and $y \notin M$. Since M is a minimum cost mapping, its cost is:
$$\gamma(M) = D_C(F_1[x], F_2[y]) + \gamma(x, \lambda) + \gamma(\lambda, y).$$
Since γ verifies the triangle inequality, the minimum cost in this case is greater than or equal to the minimum cost in the previous case.

The minimum cost of a mapping in $\mathcal{M}_C(T_1[x], T_2[y])$ is then the minimum of the minimum costs of mappings in the first three cases. □

Proof of Lemma 5. We consider a partition of $\mathcal{M}_C(F_1[x], F_2[y])$ in 4 subsets according to whether there is a vertex $y_k \in \mathrm{C}(y)$ such that $M(F_1[x]) \subseteq F_2[y_k]$ or not and there is a vertex $x_k \in \mathrm{C}(x)$ such that $M(F_2[y]) \subseteq F_1[x_k]$ or not. Let M be a minimum cost mapping in $\mathcal{M}_C(F_1[x], F_2[y])$.

- If $\exists\, y_k \in \mathrm{C}(y) \mid M(F_1[x]) \subseteq F_2[y_k]$ and $\nexists\, x_k \in \mathrm{C}(x) \mid M(F_2[y]) \subseteq F_1[x_k]$ then since M is a minimum cost mapping, its cost is:
$$\gamma(M) = \min_{y_k \in \mathrm{C}(y)} \{D_C(F_1[x], F_2[y_k]) - D_C(\emptyset, F_2[y_k])\} + D_C(\emptyset, F_2[y]).$$
- If $\nexists\, y_k \in \mathrm{C}(y) \mid M(F_1[x]) \subseteq F_2[y_k]$ and $\exists\, x_k \in \mathrm{C}(x) \mid M(F_2[y]) \subseteq F_1[x_k]$, then this case is symmetric to the previous one:
$$\gamma(M) = \min_{x_k \in \mathrm{C}(x)} \{D_C(F_1[x_k], F_2[y]) - D_C(F_1[x_k], \emptyset)\} + D_C(F_1[x], \emptyset).$$
- If $\nexists\, y_k \in \mathrm{C}(y) \mid M(F_1[x]) \subseteq F_2[y_k]$ and $\nexists\, x_k \in \mathrm{C}(x) \mid M(F_2[y]) \subseteq F_1[x_k]$, since M is a constrained mapping, it is a restricted mapping from $F_1[x]$ to $F_2[y]$. Since M is a minimum cost mapping , its cost is the minimum cost of a restricted mapping from $F_1[x]$ to $F_2[y]$:
$$\gamma(M) = \min\{\gamma(M), M \in \mathcal{R}(F_1[x], F_2[y])\}.$$

[2] A meristem is an undifferentiated plant tissue from which new cells are formed, as that at the tip of a stem or root.

- If $\exists\, y_k \in C(y) \mid M(F_1[x]) \subseteq F_2[y_k]$ and $\exists\, x_k \in C(x) \mid M(F_2[y]) \subseteq F_1[x_k]$ then M is a particular case of restricted mapping such that $\mathcal{P}(M) = \{(x_k, y_k)\}$, its cost is then greater than or equal to the minimum cost in the previous case.

The minimum cost of a mapping in $\mathcal{M}_C(F_1[x], F_2[y])$ is then the minimum of the minimum costs of mappings in the first three cases. □

References

[1] Robert A. Wagner, Michael J. Fisher, The string-to-string correction problem, Journal of the association for computing machinery 21 (1974) 168–173.
[2] K. Zhang, D. Shasha, Simple fast algorithms for the editing distance between trees and related problems, SIAM Journal on Computing 18 (6) (1989) 1245–1262.
[3] Kaizhong Zhang, Algorithms for the constrained editing distance between ordered labeled trees and related problems, Pattern Recognition 28 (3) (1995) 463–474.
[4] Kaizhong Zhang, A constrained edit distance between unordered labeled trees, Algorithmica 15 (1996) 205–222.
[5] Christophe Godin, Yves Caraglio, A multiscale model of plant topological structures, Journal of Theoretical Biology 191 (1998) 1–46.
[6] Pascal Ferraro, Christophe Godin, A distance measure between plant architectures, Annals of Forest Science 57 (2000) 445–461.
[7] Franco P. Preparata, Raymond Tzuu-Yau Yeh, Introduction to Discrete Structures for Computer Science and Engineering, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1973.
[8] Kaizhong Zhang, Tao Jiang, Some max SNP-hard results concerning unordered labeled trees, Information Processing Letters 49 (1994) 249–254.
[9] Robert Endre Tarjan, Data Structures and Network Algorithms, in: CBMS-NFS — Regional Conference Series In Applied Mathematics, 1983.
[10] Vincent Segura, Aida Ouangraoua, Pascal Ferraro, Evelyne Costes, Comparison of tree architecture using tree edit distances: Application to two-year-old apple tree, Euphytica 161 (2007) 155–164.
[11] Christophe Godin, Evelyne Costes, Yves Caraglio, Exploring plant topological structure with the amapmod software: An outline, Silva Fennica 31 (1997) 355–366.
[12] Aida Ouangraoua, Pascal Ferraro, Laurent Tichit, Serge Dulucq, Local similarity between quotiented ordered trees, Journal of Discrete Algorithms 5 (1) (2007) 23–35.