Journal of Computational Biology
Volume 7, Number 5, 2000
Mary Ann Liebert, Inc.
Pp. 731-744

Comparison of additive trees using circular orders

Authors: Vladimir MAKARENKOV[1] and Bruno LECLERC[2]

Abstract

It has been postulated that existing species have been linked in the past in a way that can be described using an additive tree structure. Any such tree structure reflecting species relationships is associated with a matrix of distances between the species considered and called a distance matrix or a tree metric matrix. A circular order of elements of X corresponds to a circular (clockwise) scanning of the subset X of vertices of a tree drawn on a plane. This paper describes an optimal algorithm using circular orders to compare the topology of two trees given by their distance matrices. This algorithm allows us to compute the Robinson and Foulds topologic distance between two trees. It employs circular order tree reconstruction to compute an ordered bipartition table of the tree edges for both given distance matrices. These bipartition tables are then compared to determine the Robinson and Foulds topologic distance, known to be an important criterion of tree similarity. The described algorithm has optimal time complexity, requiring $O(n^2)$ time when performed on two $n \times n$ distance matrices. It can be generalized to get another optimal algorithm, which enables the strict consensus tree of k unrooted trees, given their distance matrices, to be constructed in $O(kn^2)$ time.

Key words: additive tree, circular order, Robinson and Foulds distance, strict consensus tree

[1] Département des Sciences Biologiques, Université de Montréal, C.P. 6128, Succ. Centre-Ville, Montréal, Québec H3C 3J7, CANADA.
e-mail address: makarenv@magellan.UMontreal.CA

[2] Centre d'Analyse et de Mathématique Sociales, École des Hautes Études en Sciences Sociales, 54 bd Raspail, F-75270 PARIS CEDEX 06, FRANCE, Bruno.Leclerc@ehess.fr

Introduction

A tree is a formal structure of the representation of the process of evolution. The leaves represent the species under study, the interior nodes represent virtual ancestors and the edges represent the evolutionary events. In biology this tree is called a *phylogenetic tree*, or an *additive tree* if tree edges have the valuations. The principal goal of phylogenetic reconstruction is to infer an additive tree from imperfect contemporary data, which do not correspond directly to any tree topology. Consequently, we should utilize an available fitting method to obtain the data corresponding to an additive tree.

It is important to be able to compare trees obtained from the observed real data because different fitting methods may provide different trees even for the same initial data set. When two data sets are considered, the knowledge of the comparative indices, like the Robinson and Foulds distance, for example, can give us some ideas about the similarity or dissimilarity of the evolutionary processes corresponding to these data.

Zaretskii (1965), Buneman (1971), Patrinos and Hakimi (1972) and Dobson (1974) have proved that a *dissimilarity d* corresponds to an additive tree if and only if the following statement holds: for each four elements i, j, k, l of X, we have $d(i,j) + d(k,l) \leq \max \{d(i,k) + d(j,l); d(i,l) + d(j,k)\}$. Moreover, this additive tree is unique considering a strict definition of additive trees (see the next section). A dissimilarity d fulfilling the above inequality is called a *tree metric*. Several optimal algorithms, with time complexity $O(n^2)$ for an $n \times n$ matrix, for reconstructing an additive tree corresponding to a given tree metric matrix, have been proposed in the literature. Among the works addressing this problem, the paper of Waterman *et al*. (1977) introducing the first optimal reconstruction algorithm as well as its modified version, presented by Hein (1989) in the case of binary trees, which takes $O(n\log n)$ time, should be mentioned.

Another type of tree reconstruction procedure consists of algorithms based on *circular* (*diagonal*) *orders*. A circular order of elements of X corresponds to a circular (clockwise) scanning of

the subset X of vertices of a tree drawn on a plane. The first tree reconstruction algorithms using diagonal and circular orders were independently proposed by Chaiken *et al.* (1983) and Yushmanov (1984) respectively. In Makarenkov and Leclerc (1997), we proved that the diagonal and circular orders introduced in the two above-mentioned articles are indeed the same and also suggested a manner in which they may be applied to the problems of tree metric recognition and fitting of a tree metric to a given dissimilarity.

The *Robinson and Foulds topologic distance* is an important and frequently used tool to compare additive (phylogenetic) tree structures (see for instance Robinson and Foulds (1981), Saitou and Nei (1987), Gascuel and Lévy (1996), Gascuel (1997), Makarenkov and Leclerc (1997)). This distance is equal to the minimum number of elementary operations, consisting of merging or splitting nodes, necessary to transform one tree into the other. As proved in Robinson and Foulds (1981), it is also the number of bipartitions, or Buneman's splits (1971), which belong to exactly one of the two trees. If we deal with two unrooted trees having no internal vertices labeled according to the elements of X, the Robinson and Foulds distance on the set X of n elements varies between 0 (when the trees are isomorphic) and $2n$-6 (when all non-trivial bipartitions in two trees are different; a trivial bipartition corresponds to an edge incident to a leaf), whereas, the maximum value of the Robinson and Foulds topologic distance between two unrooted trees allowing internal vertices labeled according to elements of X, is equal to $3n$-6 (for the demonstration see Robinson and Foulds (1981)).

An optimal $O(n)$ time complexity algorithm for computing the Robinson and Foulds topologic distance between two trees, given by their postordered tree representation (PSW) introduced in Standish (1980), has been proposed by Day (1985). In the same paper this algorithm was generalized for computing the *strict consensus tree* of k trees in $O(kn)$ time. Day's algorithms first transform a PSW table representations into a special cluster representation from which the Robinson and Foulds topologic distance and the strict consensus tree can be computed by means of a sufficiently complicated but optimal, regarding the time required, procedure.

Obviously, only the topology of an additive tree is meaningful when computing comparative indices, as for example the Robinson and Foulds distance or the strict consensus tree, and the values of edge lengths are not of importance. But often, for instance after performing a heuristic fitting algorithm or bootstrap and jackknife validation procedures, we have to deal with tree metric matrices and not with tree structures encoded by their PSW representations, nor bipartition matrices, nor lists of vertices and edges.

In fact, there are *two ways* for computing the Robinson and Foulds distance between two trees given by their tree metric matrices on the set X of n elements. The *first* proceeds by transforming each of two matrices into bipartition tables which can be compared line by line in order to detect matching clusters. It is without doubt the easiest and the most popular approach to compute the Robinson and Foulds distance and the strict consensus tree, as well as other tree comparative indices known in the literature. However, such an approach is not optimal because comparing two bipartition matrices (not specially ordered) takes $O(n^3)$ time. The *second way* to compute the distance should be optimal for time complexity, but too complicated for implementation due to the three different algorithms involved. First, starting from two given distance matrices we can infer the two corresponding trees under the form of their edge lists. It can be done using Waterman's *et al.* algorithm, for instance. Second, an algorithm for transforming an edge list into a PSW table or directly into Day's cluster table should be implemented. Such an algorithm is not likely to exist in the literature and should be sophisticated. Third, Day's linear time complexity algorithm can be applied to these transformed data to compute the Robinson and Foulds distance. Such an approach would lead to the optimal $O(n^2)$ time complexity procedure but is very difficult to implement in practice.

In this paper we propose another way to compute the Robinson and Foulds topologic distance between two trees given by their tree metric matrices as well as the strict consensus tree of k trees and other tree comparative indices. Thus, we introduce an optimal time complexity algorithm, taking $O(n^2)$ time to compute the Robinson and Foulds distance between two trees and $O(kn^2)$ time to compute the strict consensus tree of k trees, when applied to $n \times n$ tree metric matrices. This algorithm using the combinatorial properties of circular orders, allows for the construction of specially ordered

bipartition tables for each tree metric considered. An interesting feature of such tables is that any two of them can be compared in $O(n^2)$ time to detect matching clusters.

We begin this paper by recalling some necessary definitions followed by the presentation of the algorithm (Algorithm 1) providing a circular order of elements of X given a tree metric matrix. This algorithm is discussed in greater detail in Yushmanov (1984) and Makarenkov and Leclerc (1997). We also recall Yushmanov's result establishing that an additive tree can be encoded by a sequence, defined through a circular order of elements of X, of $2n$-3 distances between its leaves. Furthermore, the entire additive tree can be inferred from such a sequence in $O(n)$ time (see Makarenkov and Leclerc (1997) for a demonstration). Algorithm 1 is then used as a base for the design of Algorithm 2, which computes the Robinson and Foulds distance between two trees in optimal time. This algorithm is then generalized for computation, still in optimal time, of some other useful tree comparative indices as well as the strict consensus tree.

Circular orders and their application for encoding and reconstructing of additive trees

Let X be a set with n elements. A *tree metric* on X is a non-negative real function d on $X \times X$ satisfying the following conditions: for all $x, y, z, w \in X$, $d(x,y) = d(y,x)$, $d(x,y) = 0$ if and only if $x = y$, and $d(x,y) + d(z,w) \leq \max\{ d(x,z) + d(y,w) , d(x,w) + d(y,z) \}$.

A *graph* G on a finite *vertex set* $V(G)$ is a pair $G = (V(G),E(G))$, where $E(G)$, the set of the *edges* of G, is a subset of the set $V(G)^{(2)}$ of all unordered pairs of distinct elements of G; an edge is denoted here vv'. The *degree* of a vertex v is the number of edges $e \in E(G)$ such that $v \in e$. A *leaf* is a vertex of degree one. The elements of $V(T)$-X are the *interior vertices*. In a graph G, a *path* P between two vertices v and v' is a sequence of edges $vv_1, v_1v_2,\ldots, v_{k-1}v_k, v_kv'$. If, in the previous

definition, $v = v'$, then P is a *circuit* of G. The graph G is *connected* if there exists a path between v and v' for any pair v, v' of distinct vertices of G. The graph G is a *tree* if it is connected and has no circuits. A tree $T = (V(T),E(T))$ has exactly $|V(T)|$-1 edges; there is a unique path, denoted $T(vv')$, or (vv') when there is no risk of ambiguity, between any two distinct vertices v and v' of a tree.

An *additive tree* (*valued tree*) is an ordered pair $T_\ell = (T,\ell)$, where T is a tree and ℓ is a non-negative real *length* function on the edge set $E(T)$ of T. The distance $dist(v,v')$ between two vertices v and v' of T is equal to $\sum_{e\in T(vv')} \ell(e)$; it defines a tree metric. According to a well-known result recalled in the introduction, any tree metric on $X \subseteq V(T)$ corresponds to a positively valued tree.

Let U_X denote the set of unrooted additive trees, some of whose vertices are labeled by the integers in the set $X = \{1,...,n\}$; moreover, all leaves and all interior vertices of degree two are always labeled by the elements of X. Each element of U_X is associated with an unique distance matrix d of dimensions $n\times n$. In such a matrix every labeled vertex is associated with one row (column) and the value $d(i,j)$ corresponding to the pair i, j of elements of X is equal to the distance between vertices i and j in the tree.

Two options are retained throughout the paper for the choice of a class U_X of additive trees providing unicity of the tree representation of any tree metric d. Let $L(T)$ be the set of the leaves of T:

Option 1: a tree T of U_X has no edges of null length.

Option 2: no interior vertex of a tree T of U_X has degree 2 and $X = L(T)$.

To obtain a tree T' of Option 2 type from a tree T of Option 1 type, one can merely replace each interior vertex $x \in X$ of degree 2 (if any) by a new interior vertex u and an edge ux of null length.

In both cases, a tree of U_X has at most $2n$-2 vertices and $2n$-3 edges, these numbers corresponding to the non-degenerate case (a *binary tree*) where all the elements of X are leaves and all the interior vertices have degree 3. The *articulation point* $a(x)$ of $x \in X$ is the vertex v adjacent to x

(that is $vx \in E(T)$) if x is a leaf, while (under Option 1) $a(x) = x$ if not; the articulation point of a vertex indexed as x_i is denoted as a_i.

We now recall two equivalent definitions of some classes of linear orders on X. First, such a class is associated with a given non-valued tree T by geometric considerations; it has been proposed by Chaiken *et al.* (1983) and generalized by Barthélemy and Guénoche (1988, 1991). Consider a graphic planar representation of T (where two edges have no common points other than a common vertex) and an ordering obtained as follows: first, the leaf x_1 is arbitrarily chosen; then, the leaves are indexed as $x_1, x_2,\ldots, x_n$ according to a circular (clockwise) scanning of the subset X of vertices of T. Such an order, frequently called a *diagonal plane order or a circular order* in the literature. For instance, the leaf order 1, 2, 3, 4 in the three trees in Figure 1 is circular.

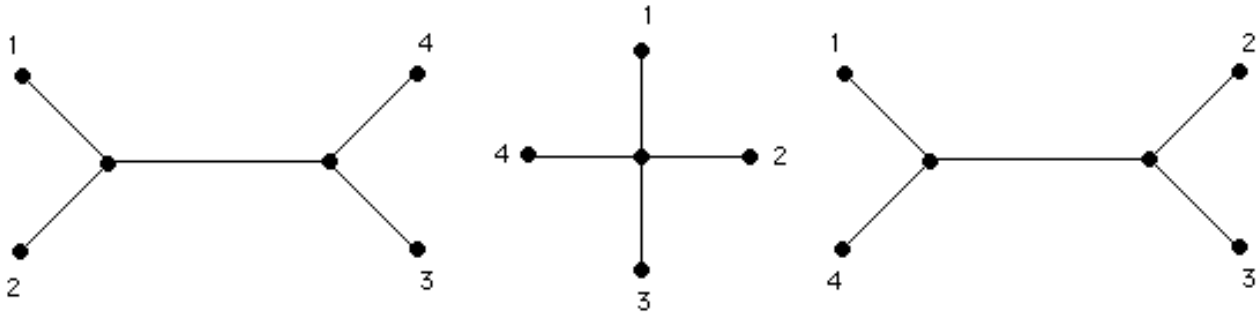

Figure 1: Three tree topologies for which 1, 2, 3, 4 is a circular leaf order

The second definition is more combinatorial and relies on one of Yushmanov's (1984) ideas; he proposed to construct, from the tree metric d on X, an ordering $x_1, x_2,\ldots, x_n$ of X such that: (i) x_1 and x_n are arbitrarily chosen; (ii) for $k = 0, 1,\ldots, n\text{-}3$, $d(x_{n-k},x_{n-k-1}) - d(x_1,x_{n-k-1}) = min_{i\in\{2,\ldots,n-k-1\}} d(x_{n-k},x_i) - d(x_1,x_i)$. This property ensures that the tree corresponding to d can be obtained by sequential grafting of the edge $a_{n-k}x_{n-k}$ (or the vertex x_{n-k}) on the path (x_1x_{n-k-1}). Such orders have been studied in more detail in Makarenkov and Leclerc (1997), where the equivalence of the Yushmanov orders of a tree metric matrix d and the circular orders of the additive tree corresponding

to *d* has been proved. In this paper we recall Yushmanov's algorithm (Table 1) for computation of a circular order of elements of X (circular order of leaves of the corresponding additive tree) given a tree metric matrix.

Algorithm 1: Construction of a circular (Yushmanov) order $x_1, x_2,\ldots,x_n$ *of the set* X

Input: a finite set X with n elements; a dissimilarity d on X.

Output: a circular (Yushmanov) order $(x_1, x_2,\ldots, x_n)$ on X associated with d.

Initialization Choose arbitrarily two leaves x_1 and x_n; $W := X-\{x_1,x_n\}$; $k = 0$

Repeat

Find x_{n-k-1} in W such that:

$d(x_{n-k},x_{n-k-1})-d(x_1,x_{n-k-1}) = min_{w\in W}\, d(x_{n-k},w)-d(x_1,w)$;

$W := W-\{x_{n-k}\}$;

$k = k+1$

Until $W = \Delta$

Table 1: Algorithm 1 (Construction of a circular order of the set X)

b	28					
c	5	25				
d	48	56	45			
e	54	62	51	8		
f	59	67	56	55	61	
g	41	49	38	37	43	30
	a	b	c	d	e	f

Table 2: Tree matrix *d* on the set of elements a, b, c, d, e, f, g

Here we demonstrate the implementation of Algorithm 1 on the tree metric *d* from Table 2. So, we are looking for a circular order of elements of X ={a, b, c, d, e, f, g}.

Let us set x_1 = a and x_7 = b ; Algorithm 1 computes a quantity: d(b,w) - d(a,w) for w = c, d, e , f and g , which gives :

d(b,c)-d(a,c)=25-5=20 d(b,d)-d(a,d)=56-48=8 d(b,e)-d(a,e)=62-54=8

d(b,f)-d(a,f)=67-59=8 d(b,g)-d(a,g)=49-41=8

Thus x_6 may be chosen from d , e , f and g providing the minimal value of 8; let us set x_6 = d and compute :

d(d,c)-d(a,c)=45-5=40 d(d,e)-d(a,e)=8-54=-46

d(d,f)-d(a,f)=55-59=-4 d(d,g)-d(a,g)=37-41=-4

Which implies that x_5 = e and, at the next step, we have :

d(e,c)-d(a,c)=51-5=46 d(e,f)-d(a,f)=61-59=2 d(e,g)-d(a,g)=43-41=2

Let us set x_4 = f and compute :

d(f,g)-d(a,g)=30-41=-11 d(f,c)-d(a,c)=56-5=51,

that implies x_3 = g and x_2 = c.

Therefore, the circular order of elements of X thus obtained is: x_1 = a ; x_2 = c ; x_3 = g ; x_4 = f ; x_5 = e ; x_6 = d ; x_7 = b.

Once a circular order (x_1, x_2,…, x_n) of elements of X is determined, the corresponding additive tree (the list of edges with their lengths) can be reconstructed from the following sequences of 2n-3 entries of the given tree metric matrix: $d(x_1,x_2)$, $d(x_1,x_3)$, $d(x_2,x_3)$, ..., $d(x_1,x_i)$, $d(x_{i-1},x_i)$,..., $d(x_{n-1},x_n)$. Moreover, this reconstruction can be performed in $O(n)$ time using an appropriate algorithm (see Yushmanov (1984) or Chaiken *et al*. (1983) for the case of unvalued trees and Makarenkov and Leclerc (1997) for the case of additive (valued) trees). This linear time complexity algorithm reconstructs an additive tree from a sequence of 2n-3 values, adding a new leaf (a new element) at a time to the growing tree and calculating the lengths of new edges. The reconstructing algorithm starts from a tree containing only the edge x_1x_2 of the length $d(x_1,x_2)$. At each step k (k = 2, ..., n-1) a new edge $a_{k+1}x_{k+1}$ of the length ($d(x_1,x_{k+1})$ + $d(x_k,x_{k+1})$ - $d(x_1,x_k)$)/2 is added to the tree, where the

articulation point a_{k+1} of the new leaf x_{k+1} is located on the path (x_1,x_k) at the distance $(d(x_1,x_k) + d(x_k,x_{k+1}) - d(x_1,x_{k+1}))/2$ from the leaf x_k.

The six consecutive additive trees (the number in parenthesis at each edge corresponds to its length) obtained by this fast reconstructing algorithm from the sequence of $2n$-3=11 entries {5, 41, 38, 59, 30, 54, 61, 48, 8, 28, 35} of the tree metric of Table 2 chosen according to the circular order a c g f e d b determined above, are presented in Figure 2.

So, any such a sequence of $2n$-3 entries of the tree matrix d constitutes a linear encoding of the minimal length of the additive tree T. This sequence may also be used to recover the entire matrix d using the following recurrence formula found by Leclerc (1995) for all $i < j$-1, $d(x_i,x_j) = \max\{ d(x_1,x_j) + d(x_i,x_{j-1}) ; d(x_1,x_i) + d(x_{j-1},x_j) \} - d(x_1,x_{j-1})$.

Calculation of the topologic distance between two additive trees given by their distance matrices

In this section we present an algorithm which computes the value of the Robinson and Foulds topologic distance between two additive trees. The algorithm described here runs in $O(n^2)$ time when starting with two $n \times n$ tree metric matrices as the input. In fact, tree valuations have no importance when calculating the Robinson and Foulds topologic distance, but, often we have to deal with the situation where the only available information is a tree metric matrix. It is assumed that the elements of X are labeled as 1,..., n, according to an arbitrary *fixed order*. Algorithm 2 (Table 10) is given in the form corresponding to Option 2 (no interior vertex of any tree T of U_X has degree 2 and $X = L(T)$), whereas Option 1 can be treated in a similar way by introducing slight modifications in the second part of the algorithm.

We start with an outline of Algorithm 2, which includes three basic parts. The first part consists of carrying out Algorithm 1 on the tree metric matrices d_1 and d_2 to determine their corresponding circular orders on X. We should perform the first part of Algorithm 2 with the same choice of the

starting element x_1 in both circular orders of d_1 and d_2. The second part of the algorithm involves consecutive performing of the following computation procedure on d_1 and d_2: a modified version of the reconstruction algorithm mentioned at the end of previous section (a linear time complexity reconstruction algorithm providing the list of tree edges with their lengths from the sequence of $2n$-3 entries of a given tree metric matrix, chosen with respect to a circular order) is applied to get at step k a reduced tree T_{k+1} with k+1 leaves corresponding to the first k+1 elements of a considered circular order. The ordered bipartition matrix of T_k, built during the previous steps, is updated into the one of T_{k+1} after taking the new edges into account. The third part of Algorithm 2 proceeds by matching edges, providing the same bipartitions in two ordered bipartition tables and thus, by calculating the topologic distance.

Here we mainly detail the working principles of the second part of this new algorithm, that is the construction of two bipartition tables. As these tables are obtained by two independent executions of the same procedure, we explain how it works on an arbitrary tree metric matrix. The bipartition table B is a binary matrix, containing only 0 and 1 values. It is composed of n columns and m rows, where each column corresponds to an element of X (a leaf of the tree) and each row corresponds to an edge of the additive tree under construction. Let us recall that the number of edges m of an additive tree with n leaves is comprised between n and $2n$-3.

The column order of B is the above-mentioned fixed order on X: the first column of B corresponds to Element 1, the second to Element 2, and so on. This fixed order should not be confused with a circular order $x_1, x_2,\ldots, x_n$ on X, obtained in the first part of Algorithm 2 and stored as a permutation of the fixed order.

The matrix $B = (B_{ij})_{i=1,\ldots|V(T)|,\, j=1,\ldots,n}$, obtained after the first k steps, is organized as follows: given a row i and a column j, the value of B_{ij} is 0 if the vertex j has not been added to T_{k+1} yet, i.e. $j \notin \{x_1,\ldots,x_{k+1}\}$; otherwise, $B(i,j) = 0$ if the vertices x_1 and j belong to the same part of the bipartition created by the edge i in T_{k+1}; and $B(i,j) = 1$, if not.

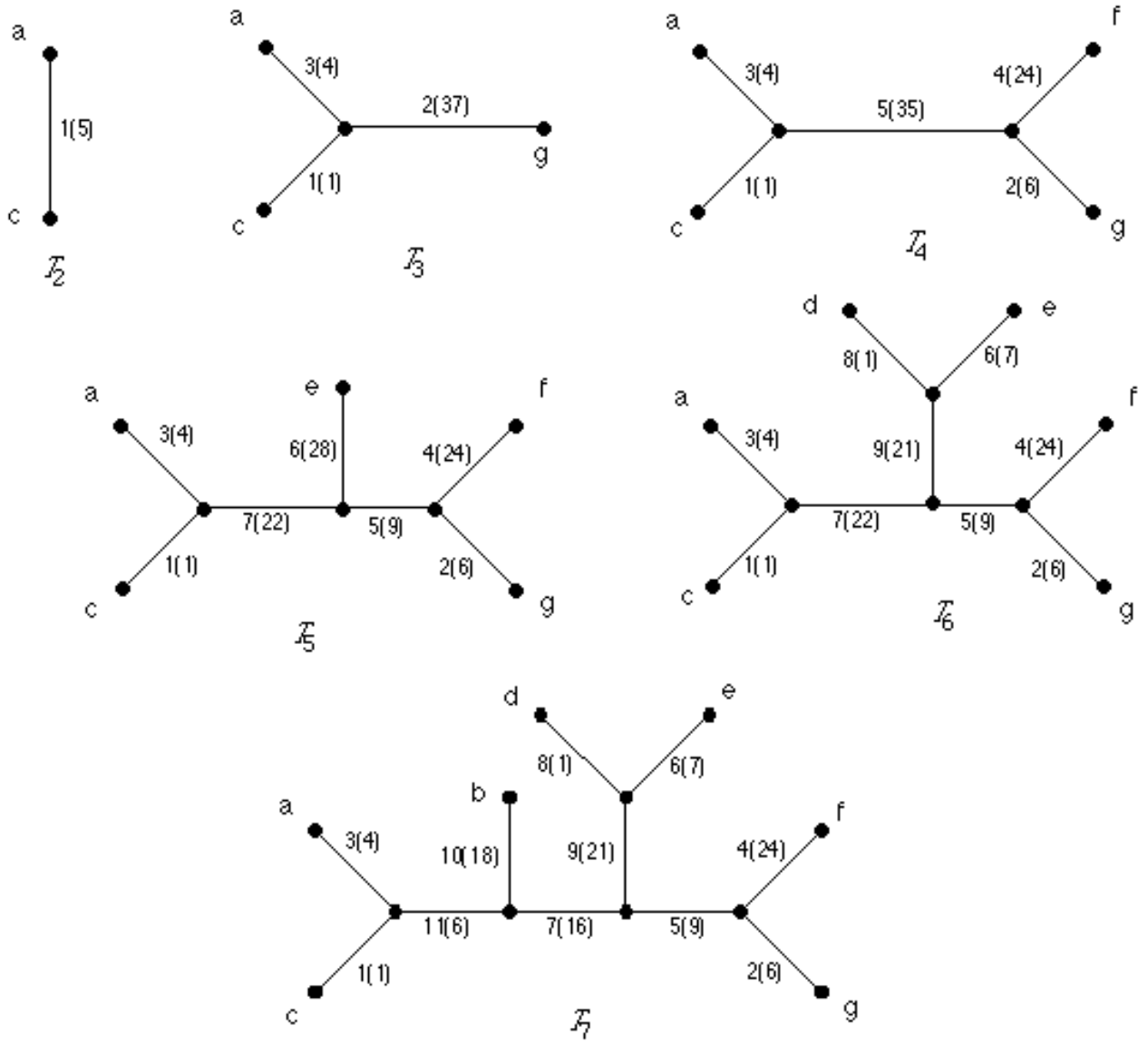

Figure 2 : Valued trees obtained by the reconstruction algorithm on the tree metric of Table 1 and then used to build an ordered bipartition table $\mathcal{B}$ in Part 2 of Algorithm 2 (the numbers in parentheses correspond to edge lengths; edges are numbered according to their positions in the bipartition table $\mathcal{B}$ building by Algorithm 2)

Let us consider the classical lexicographic linear order on the rows of B (edges of T): the row i is said greater than the row i' if there is a column j such that $B_{ij} = 1$, $B_{i'j} = 0$, and $B_{ij'} = B_{i'j'}$

for all $j' > j$. For instance, the vector 0010001 is greater than 0000001 which is, in its turn, greater than 0010000, in Table 4 corresponding to the tree T_3 from Figure 2.

Edge	a	b	c	d	e	f	g
1	0	0	1	0	0	0	0

PLACE	Path
1	1

Table 3: Bipartition table B of T_2

Edge	a	b	c	d	e	f	g
1	0	0	1	0	0	0	0
2	0	0	0	0	0	0	1
3	0	0	1	0	0	0	1

PLACE	Path
1	3
2	2
3	

Table 4: Bipartition table B of T_3

Edge	a	b	c	d	e	f	g
1	0	0	1	0	0	0	0
2	0	0	0	0	0	0	1
3	0	0	1	0	0	1	1
4	0	0	0	0	0	1	0
5	0	0	0	0	0	1	1

PLACE	Path
1	3
4	5
2	4
5	
3	

Table 5: Bipartition table B of T_4

Edge	a	b	c	d	e	f	g
1	0	0	1	0	0	0	0
2	0	0	0	0	0	0	1
3	0	0	1	0	1	1	1
4	0	0	0	0	0	1	0
5	0	0	0	0	0	1	1
6	0	0	0	0	1	0	0
7	0	0	0	0	1	1	1

PLACE	Path
1	3
6	7
4	6
2	
5	
7	
3	

Table 6: Bipartition table B of T_5

Edge	a	b	c	d	e	f	g
1	0	0	1	0	0	0	0
2	0	0	0	0	0	0	1
3	0	0	1	1	1	1	1
4	0	0	0	0	0	1	0
5	0	0	0	0	0	1	1
6	0	0	0	0	1	0	0
7	0	0	0	1	1	1	1
8	0	0	0	1	0	0	0
9	0	0	0	1	1	0	0

PLACE	Path
1	3
8	7
6	9
9	8
4	
2	
5	
7	
3	

Table 7: Bipartition table B of T_6

Edge	a	b	c	d	e	f	g
1	0	0	1	0	0	0	0
2	0	0	0	0	0	0	1
3	0	1	1	1	1	1	1
4	0	0	0	0	0	1	0
5	0	0	0	0	0	1	1
6	0	0	0	0	1	0	0
7	0	0	0	1	1	1	1
8	0	0	0	1	0	0	0
9	0	0	0	1	1	0	0
10	0	1	0	0	0	0	0
11	0	1	0	1	1	1	1

PLACE	Path
10	3
1	11
8	10
6	
9	
4	
2	
5	
7	
11	
3	

Table 8: Bipartition table B of T_7

Edge	a	b	c	d	e	f	g
10	0	1	0	0	0	0	0
1	0	0	1	0	0	0	0
8	0	0	0	1	0	0	0
6	0	0	0	0	1	0	0
9	0	0	0	1	1	0	0
4	0	0	0	0	0	1	0
2	0	0	0	0	0	0	1
5	0	0	0	0	0	1	1
7	0	0	0	1	1	1	1
11	0	1	0	1	1	1	1
3	0	1	1	1	1	1	1

Table 9: Ordered bipartition table B for the final tree T_7 Tables 3-9 completely describe the second part of Algorithm 2 (Stages 2.0 to 2.6 below), which starts with the *circular order* a c g f e d b as above and the initial *fixed order* a b c d e f g. In these tables, the first column gives edge identifications, whereas the next columns "a" to "g" are presented according to the fixed order. The meanings of the last two columns, denoted *PLACE* and *Path* (see Tables 3-9), are explained in the description below. This description discusses the full algorithmic diagram presented in Table 10.

We illustrate the steps of Algorithm 2 on the tree associated with the tree metric of Table 2. Figure 2 shows the six successive additive trees obtained at the respective six steps during the linear time complexity reconstruction procedure, given the circular order a c g f e d b of elements on X and the sequence of the corresponding 2n-3 tree metric entries from the example of previous section. Here we will show how six consecutive ordered bipartition tables corresponding to the six trees from Figure 2 can be built using some combinatorial properties of circular orders.

Let us set $\Delta_{i,j}^{k} = (d(x_i,x_k) + d(x_j,x_k) - d(x_i,x_j))/2$. It is worth noting that in an additive tree the quantity $\Delta_{i,j}^{k}$ is equal to the distance between the vertex x_k and the path (x_i,x_j).

Algorithm 2 Calculation of Robinson and Foulds topologic distance

Input: two tree metrics d_1 and d_2 on the same finite set $X = \{1,2,...,n\}$.

Output: Robinson and Foulds topologic distance between d_1 and d_2.

1. Perform **Algorithm 1** on d_1 and d_2 in order to obtain a circular order of elements of X for each tree metric, choosing the same element of X as a starting element $x_1$of both circular orders.

2. Execute independently on d_1 and d_2 to obtain their bipartition matrices B_1 and B_2 and the corresponding linked lists $PLACE_1$ and $PLACE_2$, which provide their edge orders.

2.0 **Initialization**

$B(i,j) = 0$; ($i = 1,\ldots, 2n\text{-}3$, $j = 1,\ldots, n$); $MaxCol(i) = 0$, ($i = 1,\ldots, 2n\text{-}3$); $k = 2$;

$B(1,x_2) = 1$; $MaxCol(1) = x_2$; $Path(1) = 1$; $PLACE(1) = 1$;

2.1. Run each edge of the path (x_1,x_k) starting from x_k until finding an edge uv such that:

$dist(x_k,v) \leq \Delta^k_{1,k+1} \leq dist(x_k,u)$;

Remove edge number of each passed edge, including uv, from the array $Path$;

2.2. **If** $dist(x_k,v) < \Delta^k_{1,k+1} < dist(x_k,u)$, **then**

Add the number of the edge ua_{k+1} in B equal to $m(T_k)$+2 to $Path$;

If $dist(x_k,v) < \Delta^k_{1,k+1} < dist(x_k,u)$, **then do** $j = 1, n$

$B(m(T_k)+2,j) = B(uv,j)$;

Add the number of the edge $a_{k+1}x_{k+1}$ in B equal to $m(T_k)$+1 to $Path$;

2.3. **Do** $i = 1, |Path|$

$B(Path(i), x_{k+1}) = 1$;

2.4. $MaxCol(m(T_k)+2) = MaxCol(uv)$;

Do $i = 1, |Path|$

if $MaxCol(Path(i)) < x_{k+1}$ **then** $MaxCol(Path(i)) = x_{k+1}$;

Table 10: Algorithm 2 (Calculation of Robinson and Foulds topologic distance)

Table 10 (Continued)

2.5. Insert $m(T_k)+1$ in $PLACE(1)$;

if $(m(T_k)+2 = m(T_{k+1}))$ **then do**

$i = 2$;

while $(PLACE(i) \neq uv)$ **do**

$i = i+1$;

Insert $m(T_k)+2$ in $PLACE(i+1)$;

$i = m(T_{k+1})-1$;

$edge = 2$;

repeat

if $(PLACE(i) = Path(edge))$ **then do**

$edge = edge+1$;

$j = i+1$;

while $(x_{k+1} > MaxCol(PLACE(j))$**do**

$j = j+1$;

if $(j > i+1)$ **then**

Move up $PLACE(i)$, putting it just before $PLACE(j-1)$;

$i = i-1$;

until $(i = 0)$

2.6. **If** $k < n-1$, **then** $k = k+1$ and **go to** Stage 2.1.,

else if $k = n$, **then** sort the bipartition table B subject to the linked list $PLACE$.

3. Compare the ordered bipartition tables B_1 and B_2 associated with d_1 and d_2 to compute the Robinson and Folds distance.

2.0. The tree T_2 consists of the edge x_1x_2. The matrix B (Table 3) is initialized with a unique row containing value 1 in the case associated with the leaf c = x_2 and zero values elsewhere.

A number is assigned to each edge of the current tree (column "Edge" in Tables 3-9); this column is initialized by assigning number 1 to the unique edge x_1x_2. Another array listing the edges of the path (x_1,x_k), according to their numbers, is denoted here *Path*. An ordered linked list, denoted *PLACE*, provides the lexicographic order on the edges; it is also initialized with the rank number 1 assigned to the unique edge x_1x_2. This linked list gives edge ranks according to lexicographic growing order. The statement *PLACE*(i) corresponds to the i-th element of the list. To update the array *PLACE* in optimal time we also need an auxiliary array, denoted *MaxCol* in Table 10, which contains maximum column values of each edge of a tree under consideration. For example, in Table 4 associated with the tree T_3 we have: *MaxCol* (2) = *MaxCol* (3) = g > c = *MaxCol*(1).

2.1. The element x_{k+1} has to be added to the growing tree T_k at the current step k. In the array *Path*, the i-th entry consists of the number of the i-th edge of the path (x_1,x_k) starting from x_1. We traverse this path starting from x_k until finding an edge uv such that: $dist(x_k,v) \leq \Delta^k_{1,k+1} \leq dist(x_k,u)$. This is accomplished by summing the edge lengths of all the passed edges starting from a_kx_k. According to our tree representation agreement (Option 2), one or three new edges should be added to the tree T_k and none or one should be removed to obtain the tree T_{k+1}. Not all the edges on the path (x_1,x_k) belong to the path (x_1,x_{k+1}) and those that do not are removed from the array *Path*.

2.2. At this stage new edges are introduced in the bipartition table B. If there are three new edges to add in T_k and one to remove to obtain the tree T_{k+1}, then we complete the matrix B with two new rows $m(T_k)$+1 and $m(T_k)$+2.

If the edge ua_{k+1} is added to the tree, then its binary vector is located in the row $m(T_k)$+2 of B; it holds the same values as the row corresponding to the edge uv shared at the current step of the algorithm. This edge uv is replaced in B by the edge $a_{k+1}v$. This last operation does not involve any change in B.

2.3. The value 1 is assigned to each entry (i,x_{k+1}) of B such that i is a row associated with an edge of the path (x_1,x_{k+1}) in the new tree T_{k+1}.

2.4. This is an updating stage for the array of maximum column values *MaxCol*. The maximum column value of the row i associated with an edge of the path (x_1,x_{k+1}) becomes x_{k+1} if $MaxCol(i)$ is smaller than x_{k+1}.

2.5. Let us explain in detail the important and a bit sophisticated procedure used to update the linked list *PLACE* providing the lexicographic order of edges.

To incorporate the three new edges ua_{k+1}, $a_{k+1}v$ and $a_{k+1}x_{k+1}$ into the linked list *PLACE* we should proceed as follows: the edge $a_{k+1}x_{k+1}$ will be located at the bottom of *PLACE*, the edge $a_{k+1}v$ will replace the shared edge uv and the edge ua_{k+1} will follow $a_{k+1}v$. If there is only one new added edge $a_{k+1}x_{k+1}$, it should be located at the bottom of *PLACE*.

The addition of the new leaf x_{k+1} may involve some other modifications in the linked list *PLACE*. Let us show how to take them into account. The rank in *PLACE* of an edge of the path (x_1,x_k) of T_k can only increase if this edge belongs to the path (x_1,x_{k+1}) of T_{k+1}. The edge x_1a_1 always has the greatest rank in *PLACE* and is located at the top of this list, because all the leaves of T_{k+1}, except x_1, are located on the side of the vertex a_1. It consists of values of 1 in all entries associated with leaves x_2 to x_{k+1}. So, the rank assigned to x_1a_1 is always equal to the number $m(T_{k+1})$ of edges in the tree T_{k+1}.

Let us consider an edge uv of the path (x_1,x_{k+1}), where u is located on the path (x_1,v); its rank in *PLACE* is always greater than the rank of any edge located on the side of the new leaf x_{k+1}, i.e. any edge belonging to the subtree rooted by v and not including the edge uv. In the same way, the rank of uv is inferior to the rank of any edge of the path (x_1,u). As for other edges located on the same side as the vertex x_1 with respect to the edge uv and not belonging to (x_1,u), they may have ranks superior or inferior than that of uv and, when adding the new leaf x_{k+1}, the rank of the edge uv may become greater than the rank of some of these edges.

Thus, at the current step k, we have to update the linked list *PLACE* by checking each edge of the path (a_1,x_{k+1}) against the possibility of increasing its rank in *PLACE*. We begin this updating procedure with the second edge, called here a_1u, of the path (x_1,x_{k+1}) encoded in the array *Path*. We then compare, for the fixed order on X, the leaf x_{k+1} with the maximum leaf number, available in the array *MaxCol*, of each edge located between a_1u and x_1a_1 in *PLACE* (if such an edge exists). Once the new rank of a_1u in *PLACE* is found, we check the next edge uv of the path (a_1,x_{k+1}) in the same way. A similar checking procedure is performed on each edge of the path (a_1,x_{k+1}) to get the new linked list of edge numbers *PLACE*, corresponding to the tree T_{k+1}.

2.6. If $k < n$, then we continue the turn of the loop with an incremented value of k and go to Stage 2.1. If $k = n$, then we sort the bipartition table B subject to the linked list *PLACE* in order to provide the ordered bipartition table corresponding to the final tree with n leaves.

To show that the time complexity of Part 2 of Algorithm 2 is $O(n^2)$, we notice that the number of different edges appearing in the array *Path* does not exceed $2n$-3 and also that the rank of any of these edges belonging to the growing tree at each step of the algorithm cannot rise in the linked list *PLACE* more than $3n$-6 times.

In fact, *PLACE* can be organized as an array and not a linked list without increasing the order of complexity of Part 2 of Algorithm 4, which remains $O(n^2)$, but in this case we should carry out at each step a supplementary recording procedure that makes the algorithm less elegant.

The third part of Algorithm 2 proceeds by comparing ordered bipartition tables B_1 and B_2, corresponding to d_1 and d_2. The minimal rows of B_1 and B_2 are compared first and so on. The Robinson and Foulds topologic distance is then equal to $2n$-6 minus twice the number of common non-trivial bipartitions in the matrices B_1 and B_2. For the estimation of the total complexity of Algorithm 2, we notice that each of its three basic parts can be performed in $O(n^2)$ time. That allows us to formulate the following theorem:

Theorem *Algorithm* 2 *requires* $O(n^2)$ *operations to compute the Robinson and Foulds topologic distance between two trees given by their distance matrices of dimensions* $n \times n$.

Computing the strict consensus tree and some consensus indices

For any k-tuple $T^* = (T_1,\ldots,T_k)$ of trees in U_X, the *strict consensus tree* $C(T^*)$ is that tree in U_X containing exactly those bipartitions common to all the trees of T^*. Here we are interested in reconstructing the strict consensus tree of a tuple of trees given as a tuple $d^* = (d_1,\ldots,d_k)$ of distance matrices and we also denote it $C(d^*)$. It is not difficult to see that the consecutive carrying out of the first two parts of Algorithm 2, i.e. the determination of circular orders and ordered bipartition tables of k distance matrices, takes $O(kn^2)$ time. We can compare the ordered bipartition tables B_1 and B_2 corresponding to d_1 and d_2 to build an ordered bipartition table of $C(d_1,d_2)$. This table, denoted here B_{12}, includes only those bipartitions which appear in both B_1 and B_2 tables. Therefore, the number of rows of B_{12} is equal to the number of common clusters in B_1 and B_2.

In the same way, we compare the ordered bipartition table B_{12} with the ordered bipartition table B_3 of d_3 to build the ordered bipartition table of $C(d_1,d_2,d_3)$, and so on.

Since the comparison of two ordered bipartition tables can be carried out in $O(n^2)$, the whole algorithm computes $C(d^*)$ in $O(kn^2)$ time. Figure 3 exhibits the strict consensus tree $C(T_1,T_2)$ of two trees T_1 and T_2. In this example there are two common non-trivial bipartitions in T_1 and T_2, {ac, bdefghi} and {ach, bdefgi}.

McMorris, Meronk and Neumann (1983) defined a family of consensus functions M_ℓ on the set of the k-tuples of elements of U_X on the following way: let ℓ be an integer such that $\lfloor k/2 \rfloor + 1 \leq \ell \leq k$ (where $\lfloor k/2 \rfloor$ denotes the greatest integer not exceeding $k/2$): a bipartition of X is a bipartition of $M_\ell(T^*)$ if and only if it appears in at least ℓ of the T_i's. It may be shown that such a set of bipartitions

corresponds to a tree. Indeed, these authors considered the case of rooted trees (*hierarchies*, or *n*-trees), which are closely related to additive trees by the following one-to-one correspondence (see, e.g., Day (1985)).

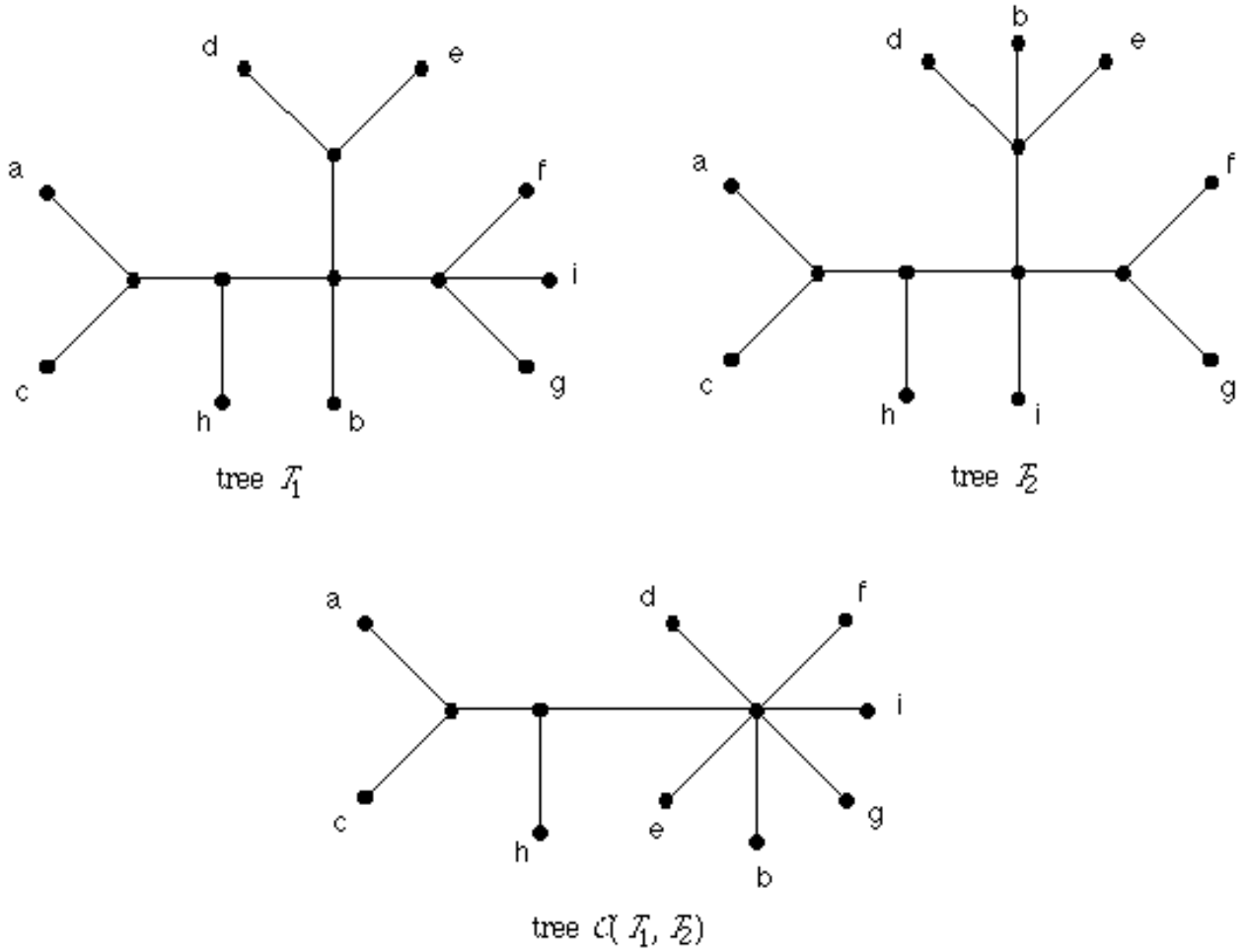

Figure 3 : The trees $\mathcal{T}_1$ and $\mathcal{T}_2$ and their strict consensus tree $\mathcal{C}(\mathcal{T}_1, \mathcal{T}_2)$

Let $x_0 \in X$; a hierarchy H on X-$\{x_0\}$ is obtained by deleting the edge a_0x_0 from an additive tree T (in the Option 2 form) and rooting the obtained tree in a_0. A cluster Y of H comes from the bipartition (Y,X-Y), where $x_0 \in X$-Y. The strict consensus C corresponds to $\ell = k$, and the *majority rule* to $\ell = \lfloor k/2 \rfloor + 1$. The latter consensus tree method is considered in Margush and McMorris (1981) for

rooted trees and by Barthélemy *et al.* (1986) in the unrooted case. Other consensus functions M_ℓ are studied in McMorris and Neumann (1983) and Barthélemy (1988) for hierarchies, and in Barthélemy and Janowitz (1991) for both kinds of trees (see Leclerc (1998) for a recent review on the consensus of classification trees). It is easy to see that the consensus tree $M_\ell(d^*)$ of k trees given by a k-tuple d^* of distance matrices can be computed in $O(k(k-l+1)n^2)$ time using a variant of Algorithm 2.

Let $S(T)$ be the set of bipartitions of a tree $T \in U_X$. Several comparison indices measuring a degree of agreement among two trees T_1 and T_2 in U_X have been proposed in the literature. They are often based on the bipartition sets $D(T_1,T_2) = (S(T_1)-S(T_2))\cup(S(T_2)-S(T_1))$ (distance indices) or $S(C(T_1,T_2))$ (consensus indices); for another paradigm, see Goddard *et al.* (1995). For instance, as noted above, the Robinson and Foulds distance between T_1 and T_2 is the cardinality of $D(T_1,T_2)$, and the CI_C index of Nelson (1979) is a normalization of the cardinality of $S(C(T_1,T_2))$; other indices take in consideration some parameters related to the blocks of the bipartitions of $D(T_1,T_2)$ or $S(C(T_1,T_2))$, for instance their cardinalities. Day (1985) surveys eight such indices, sometimes adapted from comparison indices on hierarchies; other indices are proposed in Barthélemy *et al.* (1986), following a study about hierarchies of Leclerc (1985). Shao (1983) and Day (1985) investigate interrelationships among many of these measures as well as among some other consensus indices available only for rooted trees. Since Algorithm 2 allows us to determine efficiently the sets $D(T_1,T_2)$ and $S(C(T_1,T_2))$, it is still an efficient tool for the calculation of almost all these comparison indices.

Discussion

We described an optimal algorithm (Algorithm 2) designed to compute in $O(n^2)$ time the Robinson and Foulds topologic distance between two trees given by their distance (tree metric) matrices. This algorithm can be adapted to compute in optimal ($O(kn^2)$) time the strict consensus tree

of k trees as well as most of the well-known consensus indices between two trees in $O(n^2)$. Our algorithm consists of three basic parts: the first part proceeds by obtaining a circular order of the elements of X for both given distance matrices; the second part, performed independently on both distance matrices, uses this circular order to build an ordered bipartition table of a unique additive tree associated with distance matrix; the third part proceeds by matching two ordered bipartition tables and, subsequently computing the Robinson and Foulds distance.

Since optimal procedures for inferring an additive tree in U_X from its distance matrix require $O(n^2)$ time to solve this problem, the new algorithm presented in this paper allows us to compare trees given by their distance matrices without increasing the order of complexity of the tree inferring procedure.

In this work we emphasize the usefulness of circular orders in the study of additive trees. These orders may be employed not only for inferring, fitting or drawing additive trees in optimal time, as in Chaiken *et al*. (1983), Yushmanov (1984) and Makarenkov and Leclerc (1997), but also for fast comparisons of two or more tree structures represented by their distance matrices. As has been shown in the latter work, a circular order determination algorithm (Algorithm 1 of this paper) can be implemented not only with a given tree metric matrix as the input, but also with any dissimilarity matrix. So, some new similarity measures between two or more dissimilarities can be defined by the consecutive performing Algorithms 1 and 2. Such new dissimilarity measures as well as development of other properties of introduced ordered bipartition tables might be interesting problems for further investigation.

A computer program (distributed as freeware, for Windows 32-bit, Macintosh and various versions of UNIX) which performs the computation of the Robinson and Foulds topological distance between two or more additive tree distance matrices according the Algorithm 2 described here, as well as its C source code, are available on the World Wide Web at URL <http://www.fas.umontreal.ca/BIOL/legendre/index.html>. Algorithms 1 and 2 are also part of the computer package T-REX (Macintosh and Windows versions available at the above-mentioned URL), which also includes some popular methods of tree reconstruction, such as ADDTREE by

Sattath and Tversky (1977), the Neighbor Joining method by Saitou and Nei (1987), the Unweighted Neighbor Joining method by Gascuel (1997), the Method of Weights by Makarenkov and Leclerc (1999), and others.

Acknowledgments

This research was supported by NSERC grant no. OGP7738 to P. Legendre. The authors thank Ph. D. student Matthew B. Norton for his numerous revising suggestions.

References

Barthélemy, J.P. 1988. Thresholded consensus for n-trees, *J. of Classification* 5 (2), 229-236.

Barthélemy, J.P., and Janowitz, M.F. 1991. A formal theory of consensus, *SIAM J. Discr. Math.*, 4, 305-322.

Barthélemy, J.P., Leclerc, B, and Monjardet, B. 1986. On the use of ordered sets in problems of comparison and consensus of classifications, *J. of Classification* 3, 187-224.

Barthélemy, J.P., and Guénoche, A. 1988. *Les arbres et les représentations des proximités*, Paris: Masson, transl. 1991, *Trees and proximity representations*, New York: Wiley.

Buneman, P. 1971. The Recovery of Trees from Measures of Dissimilarity, 387-395. In *Mathematics in Archaeological and Historical Sciences*, eds. F.R. Hodson, D.G. Kendall and P. Tautu, Edinburgh : Edinburgh University Press.

Chaiken, S., Dewdney, A.K., and Slater, P.J. 1983. An optimal diagonal tree-code, *SIAM Journal on Algebraic and Discrete Methods* 4 (1), 42-49.

Day W.H.E. 1985. Optimal Algorithms for Comparing Trees with Labelled Leaves, *J. of Classification* 2, 7-28.

Dobson A.J. 1974. Unrooted trees for numerical taxonomy, *J. Appl. Prob.* 11, 32-42.

Gascuel O. 1997. Concerning the NJ algorithm and its unweighted version UNJ, 149-170 In *Mathematical hierarchies and Biology* (B. Mirkin *et al*., eds.), *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* 37, Amer. Math. Soc., Providence, R.I.,.

Gascuel O., and Lévy D. 1996. A reduction algorithm for approximating a (nonmetric) dissimilarity by a tree distance, *J. of Classification* 13, 129-155.

Goddard W., Kubicka E., Kubicki G., and McMorris F.R. 1995. Agreement subtrees, metrics and consensus for labeled binary trees, 97-104. In *Partitioning Data Sets* (I. Cox *et al*., eds.), *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* 19, Amer. Math. Soc., Providence, R.I.

Hein J.J. 1989. An optimal algorithm to reconstruct trees from additive distance data, *Bull. Math. Biol.* 51 (5), 597-603.

Leclerc B. 1985. La comparaison des hiérarchies : indices et métriques, *Math. Sci. hum.* 92, 5-40.

Leclerc B. 1995. Minimum spanning trees for tree metrics: abridgements and adjustments, *J. of Classification* 12, 207-241.

Leclerc B. 1998. Consensus of classifications: the case of trees, the Proceedings of IFCS-98, Berlin, Springer-Verlag.

Makarenkov V., and Leclerc B. 1997. Tree metrics and their circular orders: some uses for the reconstruction and fitting of phylogenetic trees, 183-208. In *Mathematical hierarchies and Biology* (B. Mirkin *et al*., eds.), *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* 37, Amer. Math. Soc., Providence, R.I.

Makarenkov V., and Leclerc B. 1999. The fitting of a tree metric to a given dissimilarity with the weighted least squares criterion, *Journal of Classification*, 16, 3-26.

Margush T., and Mc Morris F.R. 1981. Consensus n-trees, *Bull. Mathematical Biology* 43 (2), 239-244.

McMorris F.R., and Neumann D.A. 1983. Consensus functions defined on trees, *Math. Soc. Sci.* 4, 131-136.

McMorris F.R., Meronk D.B., and Neumann D.A. 1983. A view of some consensus methods for trees, 122-126. In J. Felsenstein, ed., *Numerical Taxonomy*, Berlin, Springer-Verlag.

Nelson G. 1979. Cladistic Analysis and Synthesis: principles and definitions, with a historical note on Adanson's Familles des plantes (1763-1764), *Systematic Zoology* 28, 1-21.

Patrinos A.N., and Hakimi S.L. 1972. The distance matrix of a graph and its tree realization, *Quart. Appl. Math.* 30, 255-269.

Robinson D.R., and Foulds L.R. 1981. Comparison of phylogenetic trees, *Mathematical Biosciences* 53, 131-147.

Saitou N., and Nei M. 1987. The neighbor-joining method: a new method for reconstructing phylogenetic trees, *Molecular Biology Evolution* 4, 406-425.

Sattath S., and Tversky A. 1977. Additive similarity trees, *Psychometrika* 42, 319-345.

Shao K. 1983. *Consensus methods in numerical taxonomy*, Ph.D. dissertation, State University of New York, Stony Brook, New York, U.S.A.

Standish T.A. 1980, *Data structures techniques*, Addison-Wesley, Reading, Mass.

Waterman M.S., Smith T.F., Singh M., and Beyer W.A. 1977. Additive evolutionary trees, *J. Theor. Biol.* 64, 199-213.

Yushmanov S.V. 1984. Construction of a tree with p leaves from $2p$-3 elements of its distance matrix (russian), *Matematicheskie Zametki* 35, 877-887.

Zaretskii K. 1965. Construction of a tree on the basis of a set of distances between its leaves (russian), *Uspekhi Mat. Nauk.* 20, 90-92.