The triangles method to build X-trees from incomplete distance matrices.

Alain GUÉNOCHE[1], Bruno LECLERC[2]

1- Institut Mathématique de Luminy, CNRS
163 Av. de Luminy
F-13009 Marseille, France
guenoche@iml.univ-mrs.fr

2- Centre d'Analyse et de Mathématiques Sociales
École des Hautes Études en Sciences Sociales
54 boulevard Raspail
F-75270 Paris cedex 06, France
leclerc@ehess.fr

Keywords: *X*-tree; partial distances; 2-trees

Abstract.

A method to infer *X*-trees (valued trees having *X* as set of leaves) from incomplete distance arrays (where some entries are uncertain or unknown) is described. It allows us to build an unrooted tree using only $2n$-3 distance values between the n elements of *X*, if they fulfill some explicit conditions. This construction is based on the mapping between *X*-tree and a weighted generalized 2-tree spanning *X*.

Résumé.

On décrit une méthode pour la reconstruction des *X*-arbres (arbres valués admettant *X* comme ensemble de feuilles) à partir de tableaux de distances incomplets (où certaines valeurs sont incertaines ou inconnues). Elle permet de construire un arbre non orienté à partir de $2n$-3 valeurs de distance entre les n éléments de *X*, sous des conditions qui sont explicitées. Cette construction est basée sur une relation entre *X*-arbres et 2-arbres valués généralisés d'ensemble de sommets *X*.

Acknowledgements

We are deeply grateful to Laurent Duret (L.G.B.P., Lyon) who is at the origin of this work and who helped us to improve the algorithm, providing partial distance matrices from his HOVERGEN database. We also thank an anonymous referee for his comments and suggestions.

1. Introduction and notations

In molecular phylogeny and more generally in tree reconstruction from distance arrays, we are often confronted to partial distances, that correspond to matrices with unknown values. In such cases, the classical distance methods (e.g. NJ, Saitou and Nei 1987) are no longer appropriate to compute a tree because they need all distance values. A possible approach to complete the distance matrix consists in estimating missing values by techniques based on the four point condition (Lapointe and Kirsch 1996, Guénoche and Grandcolas 1999). But such methods fail when there is no quadruple with only one missing value, whereas the quality of the prediction decreases with the number of known values. The purpose of this paper is to develop a method for the reconstruction of a valued X-tree from a tree metric, that only uses a few distance values, and to adapt it to fit a tree metric to a given partial metric (or dissimilarity).

The method we propose is sequential. At each step, a new element z is placed according to two previously examined elements; let $a(z)$ be the connecting point of the corresponding new edge $\{z,a(z)\}$. We have to choose a pair $\{x,y\}$ of leaves in the current tree, to determine the position of the vertex $a(z)$ on the path between x and y, and to compute the length of the edge $\{z,a(z)\}$. At each step, a valued tree on a subset of X is obtained, that will not be modified in the sequel. The algorithm stops when the last element is placed.

The above algorithmic principle was first proposed in phylogeny by Farris (1972) when he extended the reconstruction of Wagner trees, defined for taxa described by observed characters, to distances between amino acids sequences. At each step he tries to graft a new taxon z by placing first a new internal node $a(z)$ onto one of the edges of the current tree A and then adding the edge $\{z,a(z)\}$. He retains, as we do, the edge of A which is the closest to z. For that, distances between z and the two ends of each edge have to be known. Consequently, at each step, when a new node $a(z)$ is added, distances between unplaced taxa and $a(z)$ are estimated using a formula which is only correct for tree distances. In our method, the positioning of $a(z)$ is made in relation to paths between leaves, and we do not need to expand the distance matrix, by calculating all distance values between leaves and internal vertices as in the Farris method.

Later, Waterman et al. (1977) and Hein (1989) have used again this idea to reconstruct a tree minimizing the number of necessary distance values when they are evaluated comparing biological sequences using a quadratic algorithm. In both papers, several paths are tried until obtaining the correct position for z. At the beginning, an arbitrary first path $[x,y]$ between taxa is tested, and retained if $a(z)$ is a new vertex.

Otherwise, it is a node that is the grafting point of a subtree; this point and a new leaf y is chosen in that subtree to iterate the process. This algorithm, based on the coincidence of nodes, works only with tree metrics that fulfills the famous four point condition (Buneman, 1971). When it is not the case, Waterman et al. adopt a heavy mathematical programming method. The location of a new edge is determined by minimizing a linear function of the edge lengths, with some linear constraints. So they use a linear programming method depending on the order in which the taxa are added. Moreover, these constructions make no use of a fundamental property of the correct path, that corresponds to the minimum length for the edge $\{z,a(z)\}$. An alternative strategy, avoiding these drawbacks, is proposed in Section 2 below.

Most of the following notations are taken from Barthélemy and Guénoche (1991). Let X be the set of elements, of cardinality $| X | = n$, and D be the given, possibly incomplete, distance matrix. The positive or null values of D correspond to known distances, while negative values, generally normalized to -1, account for unknown distances.

An X-tree (or phylogenetic tree) is a tree $A_X = (X \cup X', E, L)$, where:
- X is the set of leaves (or external vertices),
- X' is the set of latent vertices (or internal vertices),
- $E \subset \{(X \times X') \cup (X' \times X')\}$ is the set of edges, and
- L: $E - \mathrm{R}_+$ is the edge length function on E.

The unique path between two vertices x and y of the current tree A is denoted as $[x,y]$, or $A[x,y]$ in case of ambiguity about the tree. Edge lengths extend to path lengths by denoting as $L[x,y]$ the length of the path $[x,y]$, that is the sum of the lengths of the edges of the path of the tree A between x and y.

A *dissimilarity* d on X is a non-negative real function on $X \times X$ satisfying, for all $x, y \in X$, $d(x,y) = d(y,x)$, and $d(x,y) = 0$ if and only if $x = y$. The dissimilarity d is a *distance*, or a *metric*, if, moreover, it satisfies the triangular inequality $d(x,z) \leq d(x,y) + d(y,z)$ for all $x, y, z \in X$. It is a *tree metric* if it satisfies, for all $x, y, z, w \in X$, $d(x,y) + d(z,w) \leq \max\{ d(x,z) + d(y,w) , d(x,w) + d(y,z) \}$ (the four point condition). It is well known that, if d is a tree metric on X, then there exists a unique positively valued X-tree, called the *support tree* of d, such that, for any $x, y \in X$, $L[x,y] = d(x,y)$. If this tree has n leaves, then it has at most $2n$-2 vertices and $2n$-3 edges. When these maximum numbers are obtained, all the latent vertices have degree 3.

In the sequel, it is generally assumed that the given, possibly incomplete, dissimilarity D is extracted from a distance on X. Otherwise, it will be precised when it

is allowed to come from a dissimilarity or when it is assumed to come from a tree metric.

Given a subset Y of X and an X-tree A, the subtree A_Y is the Y-tree obtained by the following operations. First, each vertex of $X\backslash Y$ is deleted with its adjacent edge. Then, it may appear internal vertices of degree 2; they are suppressed by replacing their two adjacent edges with a unique one, whose length is the sum of the lengths of these edges.

The paper is organized as follows: definitions about twotrees that generalize 2-trees used by Leclerc and Makarenkov (1998) are given in section 2, together with a method allowing to recover an X-tree from a reduced set of pairwise distances between taxa. A greedy algorithm for the construction of a twotree providing a good summary of a tree metric is given in Section 3. In Section 4, the conditions under which a partial distance defines an entire tree metric array are detailed. Section 5 is devoted to an algorithm for building, when possible, a phylogenetic tree from an incomplete distance matrix.

2 The triangles method

The triangles method described all along this paper allows us to reconstruct the support tree of a tree metric. A first, incomplete, version is underlying in Leclerc (1995) and formalized in Makarenkov (1997) and Leclerc and Makarenkov (1998). The particularity of the method lies in an intermediate step between the distance array and the estimated X-tree. This step consists in the construction of a graph of the twotree type. So, the algorithm involves two parts: first, the construction of a "greedy" twotree graph from the distance values; then, the construction of an X-tree from this twotree. The first part will be presented in Section 3, and the second one in this section, after definitions about twotrees.

2.1. *What is a twotree ?*

Let $G = (X,E)$ be a finite unoriented simple graph (where n is the cardinality of X). So, E is a subset of the set $X^{(2)}$ of unordered pairs of distinct elements of X. A vertex $x \in X$ and an edge $e \in E$ are *incident* if $x \in e$; two vertices x and y (resp. two edges a and b) are *adjacent* if $\{x,y\} \in E$ (resp. if $a \cap b \neq \emptyset$). Let $A \subseteq E$ be a set of edges of G, X_A the set of vertices incident to an edge of A, and consider the subgraph $G_A = (X_A,A)$ of G.

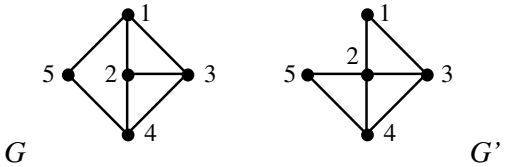
Definition. A set $C \subseteq E$ is a kd-*cycle* (d for degree) of G if all the vertices of X_C have degree at least k+1 in G_C and C is minimal for inclusion with this property. Clearly, a 1d-cycle is a cycle. Here we consider the case $k = 2$. If C is a 2d-cycle and $\{x,y\} \in C$, at least one of the vertices x and y has degree exactly 3 in G_C .

Examples. If G_C is isomorphic to the complete graph K_4 or to the complete bipartite graph $K_{3,3}$, then C is a 2d-cycle. If G_C is a wheel, then C is a 2d-cycle.

A graph with no kd-cycles is said kd-*acyclic*. The maximal kd-acyclic graphs are called here kd-*trees*, and *twotrees* for $k = 2$. Clearly, 1d-trees are trees. kd-trees have been characterized in a recursive way by Todd (1989):

- the complete graph K_k with k vertices is a kd-tree ;
- if $G = (X,E)$ is a kd-tree then, for any subset $Y \subseteq X$ of cardinality k and new vertex $z \notin X$, the graph $G' = (X \cup \{z\}, E \cup \{\{z,y\}: y \in Y\})$ is a kd-tree.

As for trees and vertices of degree 1, twotrees admit some so-called *elimination orders* defined on their vertex sets. They are orders of removal, at each step, of a vertex of degree 2 (at least one such vertex exists in a twotree) with its two adjacent edges. A twotree with n vertices is connected and has $2n$-3 edges. It is obviously a 2-connected graph. Both graphs G and G' of Figure 1 are twotrees.



G G'

Figure 1: two types of twotrees (with the natural label ordering)

In this figure, the graph G' belongs to a class introduced by Harary and Palmer (1968) as a two-dimensional generalization of trees (multidimensional generalizations were also immediately proposed in the graph theory literature). Their main properties have been established by Pippert and Beineke (1969), Rose (1974) and Proskurowski (1984). All these authors call "2-trees" these graphs. A 2-tree on X is any graph Θ with vertex set X that can be obtained by the following algorithm: start with an arbitrary pair $\{x,y\}$, which initializes the edge set E of Θ. At each step, an edge $\{x,y\}$ of the current 2-tree is chosen and a new vertex z is added, together with the edges $\{z,x\}$ and $\{z,y\}$, all the choices being done arbitrarily. Clearly, such a graph is a twotree, but the same procedure extends to twotrees relaxing condition $\{x,y\}$ being an edge. The twotree G in Figure 1 is not a 2-tree, and G' belongs to both types. More formally:

Let Y be the current vertex set of the twotree, NbPl = $| Y |$, the number of already placed vertices and E the current edge set.
$E := \{(x,y)\}$; $Y := \{x,y\}$; NbPl := 2
While NbPl < n
    Choose $z \in X \backslash Y$ and $\{x,y\} \subseteq Y$, $x \neq y$
    NbPl := NbPl + 1 ; $Y := Y \cup \{z\}$
    $E := E \cup \{(z,x), (z,y)\}$
End of While

The edges of a twotree Θ may be weighted according to the values of a given distance D on X. It will be explained below how the $2n$-3 values of the edges of a

weighted twotree provide an exhaustive representation of the $\frac{1}{2}\ n(n\text{-}1)$ values of a tree metric. Such representations have been used in Leclerc (1995), Makarenkov (1997) and Leclerc and Makarenkov (1998), but only with 2-trees. So, our presentation is more general.

2.2. *From twotrees to X-trees*

In the second part of the method, an X-tree is built from a weighted twotree Θ. In this construction, the vertices of Θ are considered in an arbitrary *reversed* elimination order $x_1, x_2, \ldots, x_n$. In such an order, for every x_i, there exist exactly two elements $y, z \in \{x_1, \ldots, x_{i-1}\}$ such that both $\{x_i,y\}$ and $\{x_i,z\}$ are edges of Θ. The basic operation is the representation of a triangle $\{x_i,y,z\}$, weighted according to the given metric D, as a 3-star, that is an $\{x_i,y,z\}$-tree with a unique internal vertex, denoted as u.

Proposition 1. *Let D be a metric on $X = \{x,y,z\}$. Then, there exists a unique X-tree of the 3-star type, whose edges $\{x,u\}$, $\{y,u\}$ and $\{z,u\}$ have positive or null lengths, respectively denoted as $L(x)$, $L(y)$ and $L(z)$ and given by:*

$$L(x) = \frac{1}{2}\ [D(x,y) + D(x,z) - D(y,z)],$$

$$L(y) = \frac{1}{2}\ [D(y,x) + D(y,z) - D(x,z)] \text{ and}$$

$$L(z) = \frac{1}{2}\ [D(z,x) + D(z,y) - D(x,y)].$$

Proof. These edge lengths actually correspond to a tree metric, and the equalities $D(x,y) = L(x) + L(y)$, $D(x,z) = L(x) + L(z)$ and $D(y,z) = L(y) + L(z)$ hold. The metric triangular inequalities imply that the edge lengths are not negative.

In the algorithm for the construction of an X-tree from a valued twotree, the basic operation above is applied to successive triangles, each with two edges of the twotree, and the obtained 3-stars are "glued". So, we start from the triple of elements $\{x_1,x_2,x_3\}$, which is represented as a 3-star accordingly with Proposition 1. Then, the same operation is made on a triangle $\{x_4,y,z\}$, adjacent to the first one (that is, $y, z \in \{x_1,x_2,x_3\}$). The second 3-star having the path $[y,z]$ in common with the first one, we merely glue the two 3-stars on this path to obtain an $\{x_1,x_2,x_3,x_4\}$-tree. Similarly, a new triangle with the vertices x_5, y, z such that $y, z \in \{x_1,x_2,x_3,x_4\}$ and $\{x_5,y\}, \{x_5,z\} \in E(\Theta)$ is considered at each step; the existence of such a triangle is guaranteed by the properties of elimination orders. If $\{y,z\}$ is not already an edge of Θ, its weight is fixed as the length of the path $[y,z]$ in the current $\{x_1,x_2,x_3,x_4\}$-tree. So, the 3-star corresponding to the triangle $\{x_5,y,z\}$ provides a grafting of the new vertex x_5 onto this tree, and so on.

Example 1. Consider the twotree of Figure 2. Figure 3 shows the 3-stars associated to its triangles and their successive incorporation, until the final X-tree is obtained.

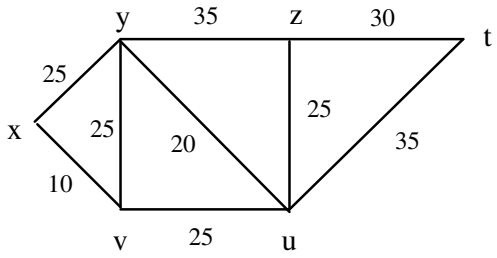


Figure 2: a twotree endowed with an incomplete distance *D*

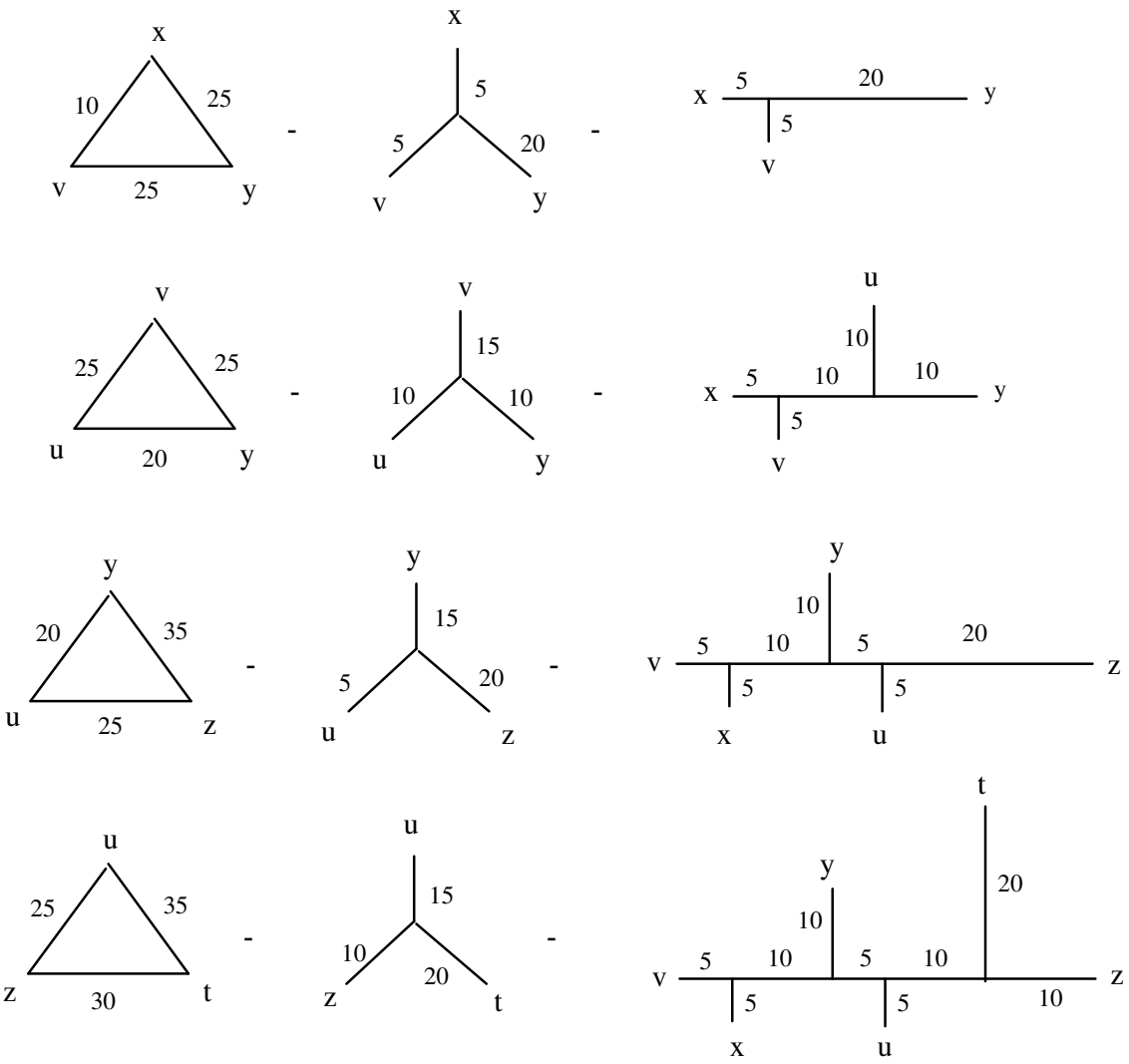


Figure 3: construction of an *X*-tree from the valued twotree of Figure 2

Remark. There may be several *X*-trees corresponding to the same 2-tree. When two adjacent triangles realize a quadruple, for instance {x,y,z} and {y,z,t} as in Figure 4 (left), there is necessarily one missing edge, here {x,t}. For the quadruple {x,y,z,t}, only two of the three sums of the four point condition can be evaluated and, if they are

different, the lenght of path [x,t] is defined. But if they are equal, many *X*-trees, formally an infinite number, correspond to a possible value for *D*(x,t).

Example 2. For the 2-tree in Figure 4 (left), we have $2 \leq D(x,t) \leq 10$, the lower bound being imposed by metricity of {x,y,t}. In the tree of Figure 4 (right), we have represented four positions for t and the rightmost is the one determined by the triangle method, since t is placed according to path [y,z]. But all positions respect the given lengths for paths [y,t] and [z,t].

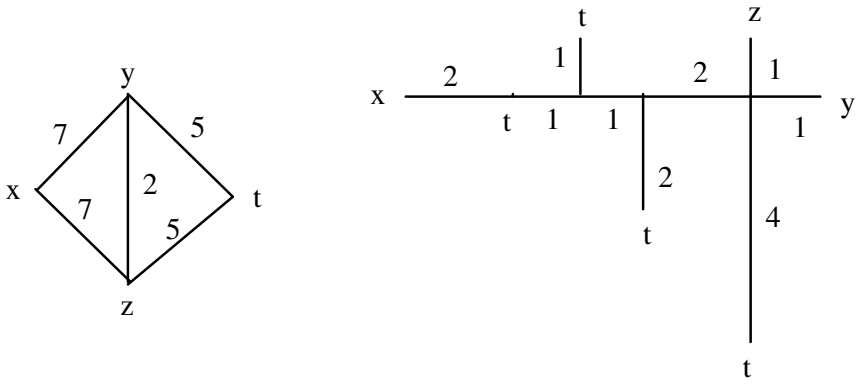


Figure 4 : A 2-tree on {x,y,z,t} and four possible positions for t in a corresponding *X*-tree

Definition A *resolved twotree* is a twotree such that, for any cycle on four elements, the two sums of the edge lengths of the opposite sides are not equal. For instance, if (*x*,*y*,*z*,*t*) is a cycle of a resolved twotree, then $D(x,y) + D(z,t)$ and $D(x,t) + D(y,z)$) are different, whatever $D(x,z)$ or $D(y,t)$ are. This terminology comes from the phylogenist community that qualify as *resolved* a quadruplet having a tree representation with a strictly positive internal edge length. In that case, the three sums of the four point condition are not equal, and for resolved twotrees, the two computable sums guarantee this property.

Proposition 2. *The weights of the edges of a resolved twotree are exactly represented as path lengths in the corresponding X-tree.*

Proof. The weights of the edges of the twotree are exactly represented as path lengths in the 3-stars, and the possibility of gluing them is due to the fact that common paths have the same length in all the 3-stars to which they belong. These lengths do not change along the iterations. This reason suffices for 2-trees but not necessarily for other twotrees. Suppose we add a new element u to the 2-tree of Figure 4, with two edges (x,u) and (t,u) having weight 3. For some positions of t, those for which path [x,t] is longer than 6, the 3-star on {x,t,u} cannot be drawn and consequently u could not be linked to the expanding *X*-tree with two paths of length 3. This difficulty is avoided by restricting Proposition 2 to resolved twotrees.

2.3 *The tree expanding procedure*

Here we describe in further details how the algorithm works. At each step, a new external vertex $x = x_i$ is added to the tree by grafting it on a "best" path $[y, z]$ to be determined. The tree structure is defined by a list of edges, all directed towards a root of the tree which is the node of the first 3-star. To reconstruct the path, the tree is scanned from y to the root and, similarly, from z to the root. These two scans join in a first common vertex r_{yz} which allows us to determine the path $[y,z]$ as the union of $[y,r_{yz}]$ and $[z,r_{yz}]$. Now, we run the path $[y,z]$, starting from y, on a length $L(y)$; if the vertex r_{yz} is overtaken, then we start from z on a $L(z)$ distance, which excludes to overtake r_{yz} again.

This procedure allows us to determine an edge $\{u,v\}$ where the new internal vertex $a(x)$ is placed. The edge $\{x,a(x)\}$ of length $L(x)$ is added to the tree. If the edge $\{u,v\}$ was obtained when starting from y, then the edge $\{u,a(x)\}$ has length $L(y) - L[y,u]$ and the edge $\{a(x),v\}$ has length $L[y,a(x)] - L(y)$; otherwise, replace y with z in these formulas.

Proposition 3. *The construction of the X-tree associated with a valued twotree has* $O(n^2)$ *time complexity.*

Proof. Since the path $[y,z]$ has at most $2n$-3 edges and is scanned at most once to place $a(x)$, the procedure for expanding the tree is linear in the number of vertices of the tree. This expanding procedure is called n-2 times and the whole construction is in $O(n^2)$.

3. Greedy twotree

The previous triangles method allows us to recover the X-tree corresponding to a twotree having X as vertex set and weighted accordingly to D, using only $2n$-3 values. Consequently, the problem is to select, in a (partial) distance matrix, the $2n$-3 pairs of a weighted twotree satisfying the following basic property of a fitting method: if D is a tree metric, then the X-tree provided by the triangles method is the support tree of D.

Three solutions to this problem, all leading to a 2-tree, are described in Leclerc (1995). In Makarenkov (1997) (see also Leclerc and Makarenkov 1998), several characteristic properties of the 2-trees that provide good representations of a tree metric D (that is, allowing to recover all the values of D) are given, with several other particular solutions. They essentially consist in sequential constructions where, at each step, a new vertex x is joined to two previously adjacent vertices y and z. We tested several methods for the construction of such twotrees computing on noised tree metrics and also partial distances coming from the HOVERGEN data base (Duret et al. 1994). Since we mainly focus on the fitness of tree structure, assumed to exist, we evaluate topological criteria as the percentage of quadruples that are correctly represented and the Robinson and Foulds (1981) distance between trees. For more details on this kind of

criteria to compare X-trees, we refer to Guénoche (1997). After many simulations, we have retained, among several alternatives, a greedy algorithm for the twotree construction.

Processing triangle $\{x,y,z\}$, for which $\{y,z\}$ are two vertices in the current twotree, allows us to place a new vertex x. The question is to choose an unplaced taxon x and a path of the current tree A_Y on which the edge $\{x,a(x)\}$ will be grafted. Remark that, for $x \in X\backslash Y$, there is, starting from x, a first node $a(x)$ of the support tree A_X of D belonging to A_Y.

Proposition 4. *Let D be a tree metric, A_X its support tree, Y a subset of X and A_Y the corresponding subtree of A_X. The length $L(x)$ of the path $A_X[x,a(x)]$ is given by*

$$L(x) = \frac{1}{2} \text{ Min}_{y,z \in Y,\, y \neq z} [D(x,y) + D(x,z) - D(y,z)].$$

More, for any $y, z \in Y$ giving this minimum, $a(x)$ belongs to the path $[y,z]$.

Proof. It is well-known that the quantity $L(x) = \frac{1}{2} [D(x,y) + D(x,z) - D(y,z)]$ is the distance $L[x,m]$ in the tree A_X from the node x to the path $[y,z]$, that is the tree distance from x to the unique vertex m common to the three paths $A_X[x,y]$, $A_X[x,z]$ and $A_X[y,z]$ (see, e.g., Leclerc and Makarenkov 1998). Let $\{u,v\}$ be the edge in A_Y containing the node $a(x)$. If the path between y and z goes through this edge, then $a(x) = m$ and the quantity $L(x) = L[x,m]$ is minimum. Otherwise $[x,a(x)] \subset [x,m]$ and this quantity is necessarily greater than $L[x,a(x)]$.

Consequently, the path on which $a(x)$ has to be placed minimizes, over the pairs y, z of distinct elements of the current twotree, the function $L(x)$ of Proposition 4. Let $\{y,z\}$ be a pair (generally not unique when D is a tree metric) of elements of X that realizes this minimum. We have $L(x) = \frac{1}{2} [D(x,y) + D(x,z) - D(y,z)]$ and $a(x) \in A_Y[y,z]$, with

$$L(A[y,a(x)]) = L(y) = \frac{1}{2} [D(y,z) + D(x,y) - D(x,z)] \text{ and}$$
$$L(A[z,a(x)]) = L(z) = \frac{1}{2} [D(x,y) + D(x,z) - D(y,z)].$$

The greedy algorithm to build a twotree is a straightforward derivation from the above result: let E be the set of the twotree edges. For each vertex s, except the first two, we use arrays Sd and Sf to record the two ends of a best path in Y to add s. We initialize the procedure taking the closest pair of elements, in agreement with the Farris (1972) choice.

Let (y,z) be a pair of elements such that $D(y,z)$ is minimum.
$E := \{\{y,z\}\}$; NbPl := 2 ; $Y := \{y,z\}$
For all $s \notin Y$
    $Sd(s) := y$; $Sf(s) = z$; $L(s) := [D(s,y) + D(s,z) - D(y,z)] / 2$
End of For all s

While NbPl < n

Let $x \notin Y$ be such that $L(x)$ is minimum

NbPl := NbPl + 1 ; $Y := Y \cup \{x\}$

$y := Sd(x)$; $z := Sf(x)$; $E := E \cup \{\{x,y\}, \{x,z\}\}$

For all $s \notin Y$

NewL := $[D(s,x) + D(s,y) - D(x,y)] / 2$

If NewL < $L(s)$ Then

$L(s) :=$ NewL ; $Sd(s) := y$; $Sf(s) := x$

End of For all s

End of While

This very simple algorithm suggests some remarks.

- It is an adaptation to twotrees of the famous Prim's algorithm (1957) for the minimum spanning tree over a valued graph. At each step we select, according to Proposition 4, an element x at minimum distance D from Y; then we compare, for every unplaced element s, its distance from Y, stored as $L(s)$, to the new edge in the Y-tree coming from x. To do that, we use the length of the path $[x,y]$ in the current tree, that is the weight of the edge $\{x,y\}$ in the twotree.
- If D is a tree metric, it is the same to use y or z to evaluate NewL. Because if element s gets $[x,y]$ as best path, it has to link the tree on the edge $\{x,a(x)\}$; but this edge is also on the path $[x,y]$. But if D is not a tree metric, some difference may appear and we use:

$$\text{NewL} := \frac{1}{2} \text{ Min } \{ [D(s,x) + D(s,y) - D(x,y)], [D(s,x) + D(s,z) - D(x,z)] \}.$$

If the distance from s to A_Y is lower than $L(s)$, then $Sd(s)$ will become y or z according to the path $[x,y]$ or $[x,z]$ giving the minimum.

Proposition 5. *Time complexity to build the X-tree corresponding to distance D is* $O(n^2)$.

Proof. Because of the begining with a closest pair, the twotree initialization is in $O(n^2)$. At each step, we select the element x at minimum distance from the current Y-tree, then we update the distances from the $Y \cup \{x\}$-tree. These two operations are in $O(n)$, so the twotree building is in $O(n^2)$. Since the construction of the X-tree associated to a given twotree has the same complexity, the whole procedure has $O(n^2)$ time complexity.

Of course, we can perform these two phases simultaneously, without increasing time complexity. It suffices to activate the expanding procedure just after having chosen a new element. Consequently, it is not necessary to memorize the twotree any more; this implementation is the one given in Section 5.

4. Application to partial distances

4.1. *Recovering an X-tree from a partial distance*

When only a partial distance is available, one can apply the same algorithm as above: a shortest known distance value $D(y,z)$ provides the initial $\{y,z\}$-tree. All the other taxa will be first linked to this path, with edge lengths given by the $L(x)$ formula. If the distance value between x and y or z is unknown, we set $L(x) := +\infty$. It is clear that a taxon can be grafted on a path if and only if its distances to both ends are known. It is the same when updating $L(s)$; to calculate NewL, we use $D(s,x)$ and at least one of the two $D(s,y)$ or $D(s,z)$. This remark allows us to formulate, in the partial tree metric case, an obvious necessary and sufficient condition to get the correct tree.

Proposition 6. *Let D be a partial tree metric having A as support tree and* $\{x,a(x)\}$ *be the edge connecting x to* $A_{X-\{x\}}$. *The tree given by the triangles method is correct (equal to A), iff, for any element x, the vertex a(x) has degree 3 and there exist on both sides apart a(x) - the side of u and the side of v if* $a(x) \in \{u,v\}$ *- two elements having known distances to x.*

As claimed above, the triangles method gives an X-tree from $2n$-3 distance values, provided the corresponding pairs define a twotree. Expanding an X-tree is possible, when, for an element x, only two distance values $D(x,y)$ and $D(x,z)$ are evaluated, even if $D(y,z)$ is unknown, or if $\{y,z\}$ is not an edge kept in the twotree.

Proposition 7. *An X-tree can be computed from a partial distance D iff there exists a linear order on X such that, for any element x, there are at least two other elements, having known distances from x, that are ranked before z.*

Proof. If $D(x,y)$ and $D(x,z)$ are evaluated, we can link x to the path $[y,z]$ if and only if $D(y,z)$ is fixed. If $D(y,z)$ is an unknown value, or if (y,z) is not a twotree edge, the value of $D(y,z)$ is provided by the length of the path from x to y in the current tree. This new evaluation of $D(y,z)$ is necessary to avoid incompatibilities when gluing the $\{x,y,z\}$-star.

Example 3. Let us consider the partial distance given in Figure 5 with its associated weighted twotree. Applying the triangle method to {x,y,u} and {y,u,z} we get the {x,y,z,u}-tree drawn in Figure 6. We calculate the length of the path between x and z in this tree and we get D(x,z):= 50. Now we can treat the triple {x,z,t} and connect t as it is represented in Figure 7.

	x	y	z	t
y	50			
z	?	40		
t	55	?	25	
u	20	50	50	?

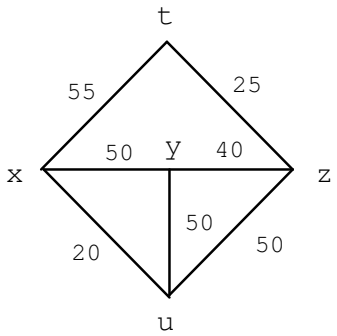


Figure 5: a partial distance and the graph of the evaluated pairs

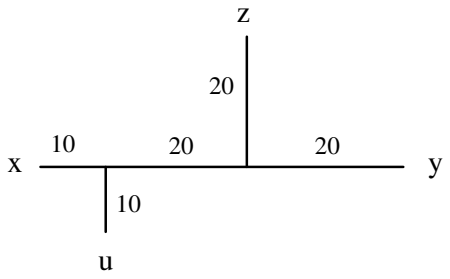


Figure 6: the valued tree representing the partial distance of Figure 5
restricted to {x,y,z,u}

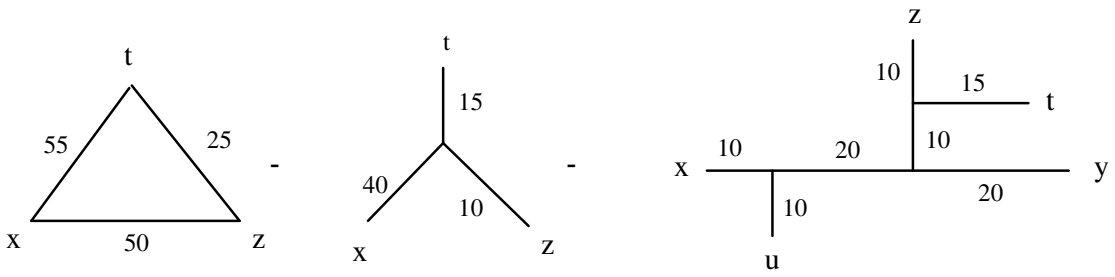


Figure 7: adding of the z element to the valued tree of Figure 6

Consequently, each time we add a new element x, the distance value $D(x,y)$ between x and any external vertex y in the current tree has to be replaced by the tree metric value $D_A(x,y)$. This operation is of course necessary if $D(x,y)$ is unknown to place a next element u for which only $D(y,u)$ and $D(z,u)$ are known. This operation is also necessary if $D(x,y)$ is known, because if $\{x,y\}$ is not a twotree edge, it may not be exactly represented, i.e., the tree path between x and y may not have the length $D(x,y)$. In such case, if $D(x,y)$ was not replaced by $D_A(x,y)$, the position of $a(u)$ on this path would depend on the starting point, x or y.

So, the distance array can be completed from $2n$-3 values, corresponding to pairs of elements of X which define a twotree. Those of the 2-tree type are more adequate for such a purpose because the used distance values are provided by direct observations.

4.3. *The case of non metric triangles*

Let us consider again the situation where the vertex x is placed on the path $[y,z]$. Sometimes, the given partial dissimilarity D does not satisfy the metric triangular inequality for all triples of elements of X. If it is the case for the triple (x,y,z), then Proposition 2 remains valid only with the introduction of some negatively valued edges in the obtained X-tree (as in Leclerc and Makarenkov 1998, where it is emphasized that such edges are external). To avoid such an unfamiliar representation of dissimilarities, we adopt an alternative option, considering two possible situations:

- Either $D(x,y) + D(x,z) < D(y,z)$, that is $L(x) < 0$. We set $L(x) := 0$, which is equivalent to place the vertex z directly on the path $[y,z]$; nevertheless, to preserve the assumption that X is a set of leaves, an edge $\{x,a(x)\}$ of null length is created. In the obtained tree, the length $L[x,y]$ is now equal to $L(y)$ and becomes superior to $D(x,y)$. Similarly, $L[x,z] = L(z) > D(x,z)$.
- Or $D(x,z) + D(y,z) < D(x,y)$ and the quantity $L(z)$ is negative (the negative $L(y)$ case is similar). This would turn out to consider two parts in the path $[y,z]$. Starting from y, the distance to z decreases first while the distance to y increases, before to find the converse situation when reaching the positively valued part of the path at the point $a(x)$. In order to avoid such an absurd effect, one sets $L(z) := 0$ and $L(y) := D(y,z)$. Then, $L[y,z] = D(y,z)$ does not change and the triangles may be glued again. In the obtained tree, $L[x,y] = L(x)$ becomes superior to $D(x,y)$, whereas $L[x,z] = D(y,z) + L(x) > D(x,z)$.

5. An algorithm for tree reconstruction from partial distances

Now we can detail our algorithm that is mainly an extension of the triangles method to partial distances.

5.1. *Ranking the set X*

We first determine a subset $Y \subset X$ of elements having known pairwise distance values. Getting Y as large as possible is obviously a clique problem (find a maximum complete subgraph in a given graph) and, for complexity reasons, we will not search a maximum cardinality subset. We begin by arranging the elements of X in the decreasing order of the number of known distance values. Let Nc be the number of elements having all the possible values. Then we examine this list from the (Nc+1)-th. For each element x, we erase those that are ranked after it and for which the distance to x is not evaluated. At the end, the remaining elements have the searched property and constitute the set Y. Let m be the cardinality of Y.

To extend this list, we introduce, one after another, the n-m erased elements, counting for each one how many distance values from the listed elements it has. At each step, we add, at the end of the list, the one having the largest number of values and we update these scores for the remaining unlisted elements.

5.2. *Building the Y-tree using the triangles algorithm*

We apply the greedy algorithm that selects, at each iteration, the closest element to the current tree.

Let $D(y,z)$ be a smallest distance value
NbPl := 2 ; $Pl := \{y,z\}$
For all $s \in Y \backslash Pl$
    $Sd(s) := y$; $Sf(s) = z$; $L(s) := [D(s,y) + D(s,z) - D(y,z)] / 2$
End of For all s
While NbPl < n

Let $x \in Y \backslash Pl$ be such that $L(x)$ is minimum
$y := Sd(x)$; $z := Sf(x)$
GrowTree(x,y,z) ; /* we expand the current tree */
UpDist(x) ; /* we calculate tree distances between x and the placed elements */
NbPl := NbPl + 1 ; $Pl := Pl \cup \{x\}$
For all $s \in Y \backslash Pl$
NewL := [$D(s,x) + D(s,z) - D(x,z)$] / 2
If NewL < $L(s)$ Then
$L(s) :=$ NewL ; $Sd(s) := x$; $Sf(s) := z$
NewL := [$D(s,x) + D(s,y) - D(x,y)$] / 2
If NewL < $L(s)$ Then
$L(s) :=$ NewL ; $Sd(s) := x$; $Sf(s) := y$
End of For all s
End of While

If the distance is complete, then $Y = X$ and the tree is achieved. If it is partial, we add the remaining elements in the order defined in section 5.1.

5.3. *Adding the elements of X\Y*

To add an element x, we test all the pairs (u,v) of placed elements such that $D(x,u)$ and $D(x,v)$ are evaluated.

For all $x \in X \backslash Pl$
For all (u,v) such that $D(x,u)$ and $D(x,v)$ are defined
$Luv := D(x,u) + D(x,v) - D(u,v)$
End of For all (u,v)
Let $\{y,z\}$ be such that $L(x) = D(x,y)+D(x,z)-D(y,z) = \mathrm{Min}_{(u,v)}\ Luv$
$Pl := Pl \cup \{x\}$; NbPl := NbPl+1
GrowTree (x,y,z)
UpDist(x)
End of For all x

Because of the last part, the final algorithm has a $O(n^3)$ time complexity, since we test an $O(n^2)$ number of paths. At the end of the process, the X-tree is given by its edge list: $\{\{x,\mathrm{Pred}(x)\}, \mathrm{Long}(x)\}$ for $x = 1,\ldots, 2n\text{-}3$. We have used procedures Path, GrowTree, UpDist which are detailed in the Annex below.

6. Discussion

Like most other distance methods, the triangles method described here reconstructs the exact support tree if the given distance corresponds to a complete tree distance or a resolved partial tree distance. Conditions for recovering the true tree, even when some distance values are missing, have been explicited in Proposition 6. Since our method

uses only a subset of the distance matrix, it is certainly less robust to deviations from tree metrics than other methods, such as NJ or BIONJ (Gascuel 1997), that take all distances into account. However, as discussed in the introduction, there are some cases where it is not possible to get all the distance values.

The triangles method is appropriate for such situations where the distance matrix contains unreliable or unknown values. A new element can be added to the tree as far as it has at least two known distances to elements present in the tree. The triangles method is simpler, and easier to implement than the ones of Waterman et al. (1977) or Hein (1989). The time complexity of the triangles method is only $O(n^2)$ with complete distances, and $O(n^3)$ with partial distances. This time complexity, for recovering an *X*-tree from a distance matrix is optimal, as it has been proved by Hein (1989). It allows to practice bootstrapping strategies on a faster way than with usual algorithms for phylogeny, which complexity functions vary from $O(n^3)$ to $O(n^5)$ (Gascuel, 1994).

A possibility to improve the reliability of *X*-trees inferred from partial distances would consist in computing first with a classical method the subtree for which all pairwise distances are available, and then using the triangles algorithm to progressively introduce in the tree the sequences for which there are some non-reliable distances. For phylogenetic studies, it also provides a strategy to introduce, at the end of the procedure, an out-group (external element to decide where the root can be) in the *X*-tree computed from an homogenous set of taxa.

References

J.P. BARTHÉLEMY, A. GUÉNOCHE (1991), *Trees and proximities representations*, J. Wiley, Chichester (UK).

P. BUNEMAN (1971), The recovery of trees from measures of dissimilarity, *in* F.R. HODSON, D.G. KENDALL, P. TAUTU, eds., *Mathematics in Archaeological and Historical Sciences*, Edinburg, Edinburg University Press, pp. 387-395.

L.DURET, D. MOUCHIROUD, M. GOUY (1994), HOVERGEN: a database of homologous vertebrate genes, *Nucleic Acids Res.* 22, 2360-2365.

J.S. FARRIS (1972), Estimating phylogenetic trees from distance matrices, *Am. Nat.* 106, 645-668.

O. GASCUEL (1994), A note on Sattah and Tversky's, Saitou and Nei's and Studier and Keppler's algorithms for inferring phylogenies from evolutionary distances, *Mol. Biol. Evol.* 11, 961-963.

O. GASCUEL (1997), BIONJ: an improved version of the NJ algorithm based on a simple model of sequence data, *Mol. Biol. Evol.* 14, 685-695.

A. GUÉNOCHE (1997), Order distances in tree reconstruction, *in* B. MIRKIN et al. eds., *Mathematical Hierarchies and Biology, DIMACS Series in Discrete Mathematics and Theoretical Computer Science 37*, Providence, RI, American Mathematical Society, pp. 171-182.

A. GUÉNOCHE, S. GRANDCOLAS (1999), Approximation par arbre d'une distance partielle, *Math. Inf. et Sc. hum.*, 146, 51-64.

F. HARARY, E.M. PALMER (1968), On acyclical simplicial complexes, *Mathematika* 15, 115-122.

J.J. HEIN (1989), An optimal algorithm to reconstruct trees from additive distance data, *Bulletin of Mathematical Biology* 51 (5), 597-603.

F.J. LAPOINTE, J.A.W. KIRSCH (1996), Estimating phylogenies from lacunose distance matrices: additive is superior to ultrametric estimation. *Mol. Biol. Evol.* 13 (6), 266-284.

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

B. LECLERC, V. MAKARENKOV (1998), On some relations between 2-trees and tree metrics, *Discrete Math.* 192 (1998), 223-249.

V. MAKARENKOV (1997), *Propriétés combinatoires des distances d'arbre : Algorithmes et applications*, Thèse de l'EHESS, Paris.

R.E. PIPPERT, L.W. BEINEKE (1969), Characterisation of 2-dimentional trees, *in* G. Chatrand, SF. Kapoor, eds., *The Many Facets of Graph Theory, Lecture Notes in Mathematics 110*, Berlin, Springer-Verlag, pp. 263-270.

R.C. PRIM (1957), Shortest connection network and some generalizations, *Bell System Tech. J.* 26, 1389-1401.

A. PROSKUROWSKI (1984), Separating subgraphs in *k*-trees: cables and caterpillars, *Discrete Math.* 49, 275-295.

D.R. ROBINSON, L.R. FOULDS (1981), Comparison of phylogenetic trees, *Mathematical Biosciences* 53, 131-147.

D.J. ROSE (1974), On simple characterizations of *k*-trees, *Discrete Math.* 7, 317-322.

N. SAITOU, M. NEI (1987), The neighbor-joining method: a new method for reconstructing phylogenetic trees, *Mol. Biol. Evol.* 4, 406-425.

P. TODD (1989), A *k*-tree generalization that characterizes consistency of dimensioned engineering drawings, *SIAM J. Disc. Math.* 2 (2), 255-261.

M.S. WATERMAN, T.F. SMITH, M. SINGH, W.A. BEYER (1977), Additive Evolutionary Trees, *J. Theor. Biol.* 64, 199-213.

Annex

We detail all the procedures used in the algorithm of Section 5.

Procedure **Path(x,y)**
/* Gives a characteristic function of the edges between x and y. An integer array Ch, indexed on all the tree vertices is required */
For k :=1 à 2N-2 ; Ch(k) := 0 ; End For k
k := x : Ch(k) := 1 /* We start from x */
While Pred(k) > 0 Do
    k := Pred(k) ; Ch(k) := 1
End of While
k := y : Ch(k) := 1 /* We start from y */
While Pred(k) > 0 Do
    k := Pred(k) ; Ch(k) := Ch(k)+1
End of while
/* Edges (k,Pred(k)) such that Ch(k) = 1 constitute the path */
Return

Procedure **GrowTree(z,x,y)**
/* We expand the Pl-tree, placing z in connection with x and y */
Lx := (D(x,y) + D(x,z) - D(y,z))/2
Ly := (D(x,y) + D(y,z) - D(x,z))/2
If Lx < 0 Then Lx := 0 And Ly := D(x,y)
If Ly < 0 Then Ly := 0 And Lx := D(x,y)
Lz := (D(x,z) + D(y,z) - D(x,y))/2 : If Lz < 0 Then Lz := 0
/* We link z on the path [x,y] at distance Lx from x and distance Ly from y */
Ns := Ns-1 ; /* it is the label of the added taxon */
Pred(z) := Ns ; Long(z) := Lz
Path(x,y)
/* We try starting from x */
Long := Lx
Lab1: If Long(x) < Long Goto Lab2
    xx=Pred(x) : Lrest := Long(x) - Long
    Pred(Ns) := xx ; Long(Ns) := Lrest
    Pred(x) := Ns ; Long(x) := Long ; Return
Lab2: Long := Long - Long(x) ; x := Pred(x) ; If x > 0 and Ch(x) = 1 Goto Lab1
/* We start from y */
Long := Ly
Lab3: If Long(y) < Long Goto Lab4
    yy=Pred(y) : Lrest := Long(y) - Long
    Pred(Ns) := yy ; Long(Ns) := Lrest
    Pred(y) := Ns ; Long(y) := Long ; Return
Lab4: Long := Long - Long(y) ; y := Pred(y) ; If y > 0 And Ch(y) = 1 Goto Lab3

Procedure **UpDist(z)**
/* The values of distance from z are replaced by tree distance values */
For all $x \in Pl$
    Path(x,z) ; Dx := 0
    For k=1 To 2N-2

```
            If Ch(k) = 1 Then Dx := Dx + Long(Ch(k))
        End For k
        D(x,z) := Dx ; D(z,x) := Dx
End For x
Return
```