An Optimal Algorithm To Recognize Robinsonian Dissimilarities

Pascal Préa, Dominique Fortin

An Optimal Algorithm To Recognize Robinsonian Dissimilarities

Pascal Préa*†
LIF, Laboratoire d'Informatique Fondamentale de Marseille, CNRS UMR 7279
École Centrale Marseille
Technopôle de Château-Gombert, 38, rue Joliot Curie, 13451 Marseille Cédex 20, France
pascal.prea@lif.univ-mrs.fr

Dominique Fortin
INRIA
Domaine de Voluceau, Rocquencourt, B.P. 105, 78153 Le Chesnay Cédex, France
dominique.fortin@inria.fr

Abstract

A dissimilarity D on a finite set S is said to be *Robinsonian* if S can be totally ordered in such a way that, for every $i < j < k$, $D(i,j) \leq D(i,k)$ and $D(j,k) \leq D(i,k)$. Intuitively, D is Robinsonian if S can be represented by points on a line. Recognizing Robinsonian dissimilarities has many applications in seriation and classification. In this paper, we present an optimal $O(n^2)$ algorithm to recognize Robinsonian dissimilarities, where n is the cardinal of S. Our result improves the already known algorithms.

Key Words

Robinsonian Dissimilarities; Classification, Seriation; Interval Graphs, PQ-Trees, Consecutive One's Property, Partition Refinement.

1 Introduction

A *dissimilarity* on a finite set S is a symmetric function $D : S \times S \to \mathbb{R}^+$ such that $D(i,j) = 0$ if $i = j$. Dissimilarities can be seen as a generalization of distances. A dissimilarity on an n-set S can be represented by a symmetric $n \times n$ matrix on $\mathbb{R}^+$ with a null diagonal, and we shall identify dissimilarities and symmetric non negative matrices with null diagonal. An $n \times n$ matrix (or dissimilarity) D on a set S is said to be *Robinsonian* if S can be linearly sorted in such a way that, for every $i < j < k$, $D(i,j) \leq D(i,k)$ and $D(j,k) \leq D(i,k)$ (Robinson 1951). Such an order is called a *compatible order* or a *compatible permutation*. If D is Robinsonian and S is sorted along a compatible order, then all the rows and columns of D (considered as a matrix) are non-decreasing when going from the diagonal.

Robinsonian dissimilarities first appeared as a tool to solve problems in seriation of archeological deposits (Robinson 1951). Given a set of archeological deposits, archeologists often make the following hypothesis: *the further in time are the deposits, the farther they are from a stylistic point of view*; that is to say that, when we measure the stylistic difference between deposits, we get a Robinsonian dissimilarity, and the chronological ordering (which is looked for) is one of the compatible orderings (see Petrie 1899, Kendall 1969). Similar problems arise in musicology (see Halperin 1994), matrix visualisation (see Caraux and Pinloche 2005, Chen and all 2004), philology (see Boneva 1980), hypertext analysis (see Berry and all 1996), ecology (see Miklós and all 2005), DNA sequencing (see Benzer 1962, Mirkin and Rodin 1984) ... Actually, Robinsonian

*Corresponding author.
†Supported in part by ANR grant BLAN06-1-138894 (projet OPTICOMB).

dissimilarities appear when data has a underlying linear total ordering and they play a prominent role in seriation.

In addition, Robinsonian dissimilarities are a generalization of ultrametrics (a distance D on a set S is an *ultrametric* if $\forall x, y, z \in S, D(x, y) \geq \max(D(x, z), D(y, z))$) which are in one-to-one correspondence with hierarchies (see Bartélemy and Guénoche 1991, Critchley and Fichet 1994). Similarly, Robinson dissimilarities are in one-to-one correspondence with a generalization of hierarchies, namely the indexed pseudo hierarchies, also called pyramids (see Durand and Fichet 1988 and Diday 1986). A *pseudo-hierarchy* on a set S is a subset H of $\mathcal{P}(S)$, whose elements are called *classes* and such that:

- $S \in H$.
- $\forall x \in S, \{x\} \in H$.
- $\forall h, h' \in H$, $(h \cap h' = \emptyset)$ or $(h \cap h' \in H)$.
- There exists an order $\leq_H$ on S such that, when S is sorted along $\leq_H$, every class is an interval.

A *(weak) index* on a pseudo hierarchy H is a function $i : H \to \mathbb{R}^+$ such that:

- $\forall x \in S, i(\{x\}) = 0$.
- $\forall h, h' \in H, h \subset h' \implies i(h) \leq i(h')$.
- $\forall h, h' \in H$, $h \subsetneq h', i(h) = i(h') \implies h' = \bigcap\{h" \in H : h \subsetneq h"\}$.

Given an indexed pseudo hierarchy (H, i), the dissimilarity D defined by $D(x, y)$ is the index of the smallest class containing x and y is Robinsonian. Conversely, given a Robinsonian dissimilarity D, it is possible to construct a pseudo-hierarchy from the set $\mathcal{B}_D = \{B(x, \lambda), x \in S, \lambda \in \mathbb{R}^+\}$, where $B(x, \lambda) = \{y \in S, D(x, y) \leq \lambda\}$. More precisely, if D is a Robinsonian dissimilarity on a finite set S, then $\{B(x, D(x, y)) \cap B(y, D(x, y)) : x, y \in S\}$ is a pseudo hierarchy on S. This one-to-one correspondence makes Robinsonian dissimilarities have many applications in classification and data analysis (see Durand and Fichet 1988 ,Critchley and Fichet 1994, Strehl and Ghosh 2003).

From a graph theoretical point of view, these dissimilarities are in relation with interval hypergraphs (see Fulkerson and Gross 1965). An *interval hypergraph* is an hypergraph (i.e. a vertex set V with a hyperedge set E, each hyperedge being a subset of V) such that the vertex set can be ordered in such a way that every hyperedge is an interval; given a dissimilarity D on a set S, $(S, \mathcal{B}_D)$ is an interval hypergraph if and only if D is Robinsonian.

The first polynomial algorithm to recognize Robinsonian dissimilarities was given in Mirkin and Rodin (1984). This algorithm is based on the relation between interval hypergraphs and Robinsonian dissimilarities; it runs in $O(n^4)$ time and $O(n^3)$ space. Chepoi and Fichet (1997) and Seston (2008a) gave $O(n^3)$ time and $O(n^2)$ space algorithms to recognize Robinsonian dissimilarities; these algorithms consider the values of D in decreasing order. The first of these algorithms uses the divide-and-conquer paradigm, and the second one constructs paths in threshold graphs of D. Seston (2008b) improved the second algorithm to $O(n^2 \log n)$.

Optimal L_∞-fitting of a dissimilarity by a Robinsonian dissimilarity is a NP-hard problem (see Barthélemy and Brucker 2003 and Chepoi and all 2009). So, there is no efficient exact algorithms for this problem, and one has to use heuristics like Brusco (2002) and Hubert (1974), or an approximation algorithm like Chepoi and Seston (2011).

In this paper, we present a new algorithm to recognize Robinsonian dissimilarities, based on interval graphs (see Golumbic 1980). Our algorithm runs in $O(n^2)$ time. Since the recognition of Robinsonian matrices needs at least to check all the elements of the matrix, our algorithm is optimal.

In Section 2, we establish some links between Robinsonian dissimilarities and interval graphs and give a sketch of our algorithm. In Sections 3, we give some sub-routines which are needed by our algorithm. In Section 4, we give a precise description of our algorithm. In Section 5, we show, on an example, how our algorithm runs. In Section 6, we mention some other applications of our algorithm.

The crucial idea of our algorithm is to consider the values $D(x, y)$ in increasing order. So, we first consider small values of D from which we get local necessary conditions on the permutations compatible with Robinson property. Although local, these informations are quite accurate: we get subsets of S such that, if D is Robinsonian, then for every compatible permutation σ, these subsets are intervals of S when S

is sorted along σ.

Then we retrieve the relative locations of these intervals thanks to large values of D (opposite to increasing threshold); however, we do not have to consider all the points of S: since the informations on the intervals are accurate, we just have to keep one or two representatives for each interval and recursively apply our algorithm on the set of representatives. This set of representatives remains small enough to guarantee that the recursion does not change the overall complexity. The major achievement over previous algorithms relies on this parsimonious shrinking of points of S.

For the sake of simplicity, our algorithm does not systematically test if D is Robinsonian or not. So, after the recursive calls, we are in one of the following two cases:

- If D is Robinsonian, then we have the set of the compatible permutations.
- If D is not Robinsonian, then we have an arbitrary set of permutations.

In order to distinguish between these two cases, we just have to take one (arbitrary) permutation of the returned set and test if it is a compatible permutation or not.

2 Robinsonian dissimilarities and interval graphs

The main goal of this section is to give some links between Robinsonian dissimilarities and interval graphs. This is a kind of folklore and, for instance, such results can be found in Mirkin and Rodin (1984). All sets and graphs considered in this paper are finite.

A graph $G = (V, E)$ is an *interval graph* if G is the intersection graph of a family of intervals on the real line, i.e. if there exists a family $\mathcal{I}$ of intervals such that $V = \{x_i, i \in \mathcal{I}\}$ and $\{x_i, x_j\} \in E \iff i \cap j \neq \emptyset$ (see Golumbic 1980). It is well known (see Gilmore and Hoffman 1964) that graph G is an interval graph if and only if its maximal cliques can be linearly ordered in such a way that if $c_1 < c_2 < c_3$, $i \in c_1$ and $i \in c_3$, then $i \in c_2$. Equivalently, the vertex-clique incidence matrix of G has the *Consecutive One's Property*: if the rows are indexed by the vertices and the columns are indexed by the maximal cliques, it is possible to sort the columns in such a way that, on every row, the 1's appear consecutively. The set of all the compatible orders for a dissimilarity can be represented by a PQ-tree (see Booth 1975, Lueker 1975 and Booth and Lueker 1976). A *PQ-tree* $\mathcal{T}$ on a set S is a tree that represents a set of permutations on S. The leaves of $\mathcal{T}$ are the elements of S, and the nodes of $\mathcal{T}$ are of two types : the *P-nodes* and the *Q-nodes*. We use the general convention of writing that is to represent P-nodes by circles, and Q-nodes by rectangles. On a P-node, one can apply any permutation of its children (equivalently, its children are not ordered). The children of a Q-node are ordered, and the only permutation we can apply on them is to reverse the order. For instance, the PQ-tree of figure 1 represents the set of permutations {(1,2,3,4,5), (1,3,2,4,5), (2,1,3,4,5), (2,3,1,4,5), (3,1,2,4,5), (3,2,1,4,5), (5,4,1,2,3), (5,4,1,3,2), (5,4,2,1,3), (5,4,2,3,1), (5,4,3,1,2), (5,4,3,2,1)}.

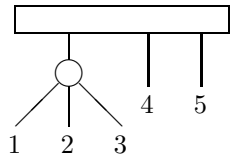


Figure 1: A PQ-tree

For a node α of a PQ-tree $\mathcal{T}$, we denote by $\mathcal{T}(\alpha)$ the subtree of $\mathcal{T}$ with root α and by S_α the set of the leaves of $\mathcal{T}(\alpha)$. We will say that a node is *basic* if all its children are leaves.

Claim 1. *For any permutation represented by $\mathcal{T}$, S_α is an interval of S.*

Let D be a dissimilarity on an n-set S. For every x in S and k in $[0, \ldots, n]$, we define the sets $\Gamma_k(x)$ by setting:

- $\Gamma_0(x) = \{x\}$,

- $\Gamma_k(x) = \{y \in S, \forall z \in S \setminus \Gamma_{k-1}(x), D(x,y) \leq D(x,z)\}$.

Equivalently,

- $\Gamma_0(x) = \Delta_0(x) = \{x\}$,
- $\Delta_k(x) = \{y \in S \setminus \Gamma_{k-1}(x), \forall z \in S \setminus \Gamma_{k-1}(x), D(x,y) \leq D(x,z)\}$,
- $\Gamma_k(x) = \Gamma_{k-1}(x) \cup \Delta_k(x)$.

The sets $\Delta_k(x)$ are spheres with center x, and the sets $\Gamma_k(x)$ are balls with center x. Intuitively speaking, $\Delta_1(x)$ contains the elements of S closest from x, the sphere made of its "inner circle" of neighbors, $\Delta_2(x)$ contains the "second circle" of neighbors,... For all i, $\Gamma_k(x)$ is the unions of the k first $\Delta_i(x)$.

Claim 2. *If D is Robinsonian and S is sorted along a compatible order, then the sets $\Gamma_k(x)$ are intervals. We will call these sets Γ-intervals.*

For every l in $[0, \ldots, n]$, we define the intersection graph $G_l = (V_l, E_l)$, where:

- $V_l = \{\Gamma_k(x), x \in S, 0 \leq k \leq l\}$
- $E_l = \{\{v_i, v_j\}, v_i \cap v_j \neq \emptyset\}$

On Figure 2, one can see that two different dissimilarities on three points a, b, c yield two different graphs G_1, although, for these four dissimilarities, the intersection graphs of $\{\Gamma_1(x)\}$ or $\{\Gamma_2(x)\}$ are K_3, and the intersection graph of $\{\Gamma_1(x)\} \cup \{\Gamma_2(x)\}$ is K_6.

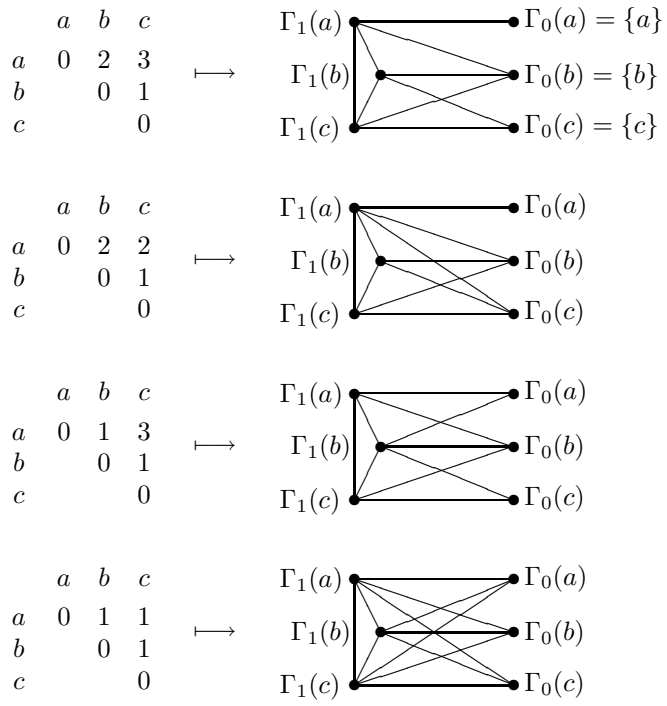


Figure 2: The graphs G_1 for the four dissimilarities on three points

The graphs G_l do not contain all the information on D. Let x and y be two elements of S. We say that $D(x,y)$ is *l-known* (or *known*) if $x \in \Gamma_l(y)$ or $y \in \Gamma_l(x)$. Otherwise, we say that $D(x,y)$ is *l-unknown* (or *unknown*) The graphs G_l have the following properties:

Property 1. *If D is Robinsonian, then, for any l in [0,...,n], G_l is an interval graph. In addition, the set of the permutations compatible with D is a subset of the set represented by the PQ-tree of G_l.*

Proof. Follows from Claim 2. □

It is clear that, if $\mathcal{I}$ is a set of horizontal intervals, then the maximal cliques of the intersection graph of $\mathcal{I}$ corresponds to vertical lines (see Figure 3). For the graphs G_l, we have the more precise property 2.

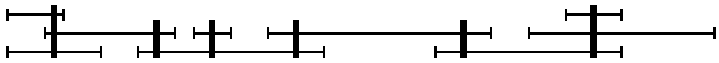


Figure 3: Maximal cliques of an interval graph

Property 2. *If D is Robinsonian, then for any l in $[0,\ldots,n]$, the maximal cliques of G_l are in one-to-one correspondence with the points of S.*

Proof. Let $C = \{v_1, \ldots v_p\}$ be a maximal clique of G_l. By Claim 2, each v_i is an interval, so $v_1 \cap \ldots \cap v_p \neq \emptyset$. Suppose that $v_1 \cap \ldots \cap v_p = \{x_1, \ldots, x_q\}$, with $q > 1$. The set $\Gamma_0(x_1)$ is not in C (it does not contain x_q), but it intersects all the elements of C. So $C \cup \{\Gamma_0(x_1)\}$ is also a clique, which is in contradiction with the maximality of C. So each maximal clique contains one (and only one) set $\Gamma_0(x)$.

Conversely, each element x of S corresponds to one maximal clique, namely the clique which contains $\Gamma_0(x)$. □

Property 3. *For any k in $[0,\ldots,n-1]$ and any x in S, if $\Gamma_k(x) = \Gamma_{k+1}(x)$, then $\Gamma_k(x) = S$.*

Proof. If $\Gamma_k(x) = \Gamma_{k+1}(x)$, then $\Delta_{k+1}(x) = \emptyset$, and so $S \setminus \Gamma_k(x) = \emptyset$ □

Corollary 1. *For any k in $[0,\ldots,n-1]$ and any x in S, $|\Gamma_k(x)| > k$.*

Proof. Since $|\Gamma_0(x)| = 1$, if for every $i < k$, $|\Gamma_{i+1}(x)| > |\Gamma_i(x)|$, then $|\Gamma_k(x)| > k$. Otherwise, by Property 3, $|\Gamma_k(x)| = n > k$. □

Theorem 1. (Mirkin and Rodin 1984 p.62) *D is Robinsonian if and only if G_n is an interval graph. In this case, the PQ-tree of G_n represents the set of the permutations compatible with D.*

Proof. By Property 1, if D is Robinsonian, then G_n is an interval graph and the set of the compatible permutations is a subset of the set represented by the PQ-tree of G_n.
Conversely, let us suppose that G_n is an interval graph and that S is ordered according to a permutation represented by the PQ-tree of G_n. For any $x < y < z$, if an interval contains x and z, it contains y. Suppose that, for some $x < y < z$, we have $D(x,z) < D(x,y)$, then there exists $k \in [1,\ldots,n-1]$ such that $z \in \Gamma_k(x)$ and $y \notin \Gamma_k(x)$, which is impossible. □

Given $l \in [0,\ldots,n-1]$, let α be a P-node of the PQ-tree of G_l, and β be one of its children. If $\Gamma_l(x) \subset S_\beta$ for some x in S_β, then we say that β is *l-free* or *free*. Otherwise, β is *l-closed* or *closed*. We say that a node α is *l-big* if $|S_\alpha| > l$. Otherwise, we say that α is *l-small*.

Property 4. *Each l-free child is l-big.*

Proof. Let α be a P-node of the PQ-tree of G_l, and β a l-free child of α, S_β contains a set $\Gamma_l(x)$ which has, by Corollary 1, more than l elements. □

Property 5. *Let α be a P-node of the PQ-tree of G_l, β a closed child of it and x an element of S_β. Then there exists $l' \leq l$ such that $\Gamma_{l'-1}(x) \subset S_\beta$ and $S_\alpha \subset \Gamma_{l'}(x)$. So if y and z are in $S_\alpha \setminus S_\beta$, then $D(x,y) = D(x,z)$.*

Proof. Let us suppose that there exist $x \in S_\beta$ and l' such that $S_\beta \subsetneq \Gamma_{l'}(x) \subsetneq S_\alpha$. Let $y \in \Gamma_{l'}(x) \setminus S_\beta$ and $z \in S_\alpha \setminus \Gamma_{l'}(x)$. Then y is in a S_{γ_1} and z is in a S_{γ_2}, where γ_1 and γ_2 are children of α. For any permutation compatible with the PQ-tree of G_l, z cannot be between x and y. But this is impossible since if $\gamma_1 = \gamma_2$, we can reverse the order of S_{γ_1}, and if $\gamma_1 \neq \gamma_2$, as α is a P-node, we can "put" γ_2 between γ_1 and β. □

Property 6. *Let D be a Robinsonian dissimilarity and $\mathcal{T}$ be the PQ-tree which represents the permutations compatible with D. If α is a node of $\mathcal{T}$, $x, y \in S_\alpha$ and $z \in S \setminus S_\alpha$,then $D(x, y) \leq D(x, z) = D(y, z)$.*

Proof. We can reverse the order of the children of any node of $\mathcal{T}$ (included the root), and S_α is an interval of S. So there exists a compatible order for which $x < y < z$, and another one for which $y < x < z$. □

Our algorithm can be briefly described in the following way (K is a fixed integer which does not depend on S):

step 1 Initialization
Check if G_K is an interval graph (actually, the algorithm checks if the vertex-clique incidence matrix of G_K has the consecutive one's property).
If this is the case, then compute the PQ-tree $\mathcal{T}$ of G_K.
Otherwise, D is not Robinsonian and the algorithm halts.

step 2 Treatment of P-Nodes
For each P-node α of $\mathcal{T}$, check if, for every x in $S \setminus S_\alpha$, $D(x, y)$ has a constant value for $y \in S_\alpha$. If this is not the case, then transform α into a Q-node.

step 3 Treatment of Q-Nodes
For each Q-node α, determine if α satisfies the conditions of Property 6. If this is not the case, and if α is the child of Q-node, then delete α (the children of α become children of its parent). If α is the child of a P-node, it will be deleted at step 4.

These three first steps constitute the "local part" of our algorithm. After Step 3, if D is Robinsonian, we have subsets of S which are intervals for any compatible permutation. These intervals correspond to some nodes of $\mathcal{T}$.

step 4 Recursive Calls and Merging
Construct a subset S' of S on which the values of D which have not been used to build G_K may have influence on the structure of $\mathcal{T}$ (these values will determine the relative locations of the intervals built during the three first steps). The set S' will be precisely defined in Subsection 4.4.
Recursively determine if D restricted to S' is Robinsonian and construct its PQ-tree $\mathcal{T}'$. The recursion stops when $S' < 3$.
Merge $\mathcal{T}$ and $\mathcal{T}'$.

step 5 Verification
Choose one (arbitrary) permutation represented by $\mathcal{T}$ and verify if it corresponds to a linear ordering. If this is the case, then D is Robinsonian and $\mathcal{T}$ represents the set of its compatible permutations. Otherwise, D is not Robinsonian.

During Step 1, the algorithm considers some values of D (the ones which define the graph G_K) and constructs, if it does not halt, the PQ-tree $\mathcal{T}$ which represents all the permutations compatible with these values. Since $\mathcal{T}$ corresponds to a subset of the $n \times n$ values of D, it represents a set of permutations which contains all the permutations compatible with D (Property 1). During Steps 2—4, the algorithm considers more and more values of D and maintains $\mathcal{T}$ such that $\mathcal{T}$ always represents the set of the permutations compatible with the considered values of D. So, at every moment, $\mathcal{T}$ represents a superset of the permutations compatible with D. At the end of Step 4, if D is Robinsonian, the non-considered values of D do not have any influence on $\mathcal{T}$, the set represented by $\mathcal{T}$ is the set of the permutations compatible with D.

3 Auxiliary Sub-routines

In this Section, we give some auxiliary sub-routines which are needed by our algorithm and are independent of the PQ-tree structure.

3.1 Basic procedures on sequences

We will use the following basic procedures, each running in linear time in the size of the input:

- REVERSE(X), whose input is a sequence $X = (x_1, x_2, \ldots, x_p)$, and output the sequence $(x_p, x_{p-1}, \ldots, x_1)$.
- INSERT(X, x_i, Y), where $X = (x_1, \ldots, x_i, \ldots, x_p)$ and $Y = (y_1, \ldots, y_q)$ are sequences, whose output is the sequence $(x_1, \ldots, x_{i-1}, y_1, \ldots, y_q, x_{i+1}, \ldots, x_p)$.
- CONCAT($X_1, \ldots, X_p$), whose input are the sequences $(X_1 = (x_1^1, \ldots, x_{i_1}^1), X_2 = (x_1^2, \ldots, x_{i_2}^2), \ldots, X_p = (x_1^p, \ldots, x_{i_p}^p))$, and whose output is the sequence $(x_1^1, \ldots, x_{i_1}^1, x_1^2, \ldots, x_{i_2}^2, \ldots, x_1^p, \ldots, x_{i_p}^p)$.

3.2 Construction of the vertex-clique incidence matrix of G_l

Let D be a dissimilarity on an n-set S. For any point x and any l in [1,...,n], it is possible to construct the sets $(\Gamma_k(x))_{0 \leq k \leq l}$ in $O(n \cdot l)$ with the following pseudo-code:

Procedure Γ-CONSTRUCTION(x, l)
Input $x \in S$, l is an integer $\leq n$.
Output the sets $\Gamma_k(x)$, for $k \leq l$, with their cardinality $Nb(k, x)$.
begin
  $PreviousMin \leftarrow -1$;
  $\Gamma_0(x) \leftarrow \{x\}$;
  For $k \leftarrow 1$ **To** l **Do**
    $Val(x)[k] \leftarrow \infty$;
    For $y \leftarrow 1$ **To** n **Do**
      If $(PreviousMin < D(x,y) < Val(x)[k])$ **and** $(x \neq y)$ **Then**
        $Val(x)[k] \leftarrow D(x,y)$;
    $PreviousMin \leftarrow Val(x)[k]$;
  For $k \leftarrow 1$ **To** l **Do**
    $\Gamma_k(x) \leftarrow \emptyset$; $Nb(k,x) \leftarrow 0$;
    For $y \leftarrow 1$ **To** n **Do**
      If $(D(x,y) \leq Val(x)[k])$ **Then**
        $\Gamma_k(x) \leftarrow \Gamma_k(x) \cup \{y\}$;
        $Nb(k,x) \leftarrow Nb(k,x) + 1$;
end

This procedure consists of two loops. In the first loop, the value $Val(x)[k]$ such that $y \in \Gamma_k(x)$ if and only if $D(x,y) \leq Val(x)[k]$ is computed. In the second loop, the procedure computes the sets $\Gamma_k(x)$.

The vertex-clique incidence matrix M of G_l has n rows and $n(l+1)$ columns. For x in $[1, \ldots, n]$ the x^{th} row corresponds to the point x of S, which is a clique of G_l. The column $(n \cdot k + x)$, where $0 \leq k < l$ and $1 \leq x \leq n$, corresponds to the set $\Gamma_k(x)$. We can construct M in $O(n^2 l)$ time with the following algorithm:

Procedure VERTEX-CLIQUE-MATRIX-CONSTRUCTION(l)
Input l is an integer $\leq n$.
The sets $\Gamma_k(x)$, for $x \in S$ and $k \leq l$, with their cardinality $Nb(k, x)$.
Output The vertex-clique-incidence matrix M of G_l.
begin
  ForAll $(x, j) \in [1, \ldots, n] \times [1, \ldots, n(l+1)]$ **Do**
    $M[x, j] \leftarrow 0$;
  ForAll $(x, i) \in [1, \ldots, n] \times [1, \ldots, l]$ **Do**
    $N[i, x] \leftarrow 1$;
  ForAll $(x, y) \in [1, \ldots, n] \times [1, \ldots, n]$ **Do**
    For $i \leftarrow 1$ **To** l **Do**
      If $(N[i, y] \leq Nb(i, y))$ **and** $(x = \Gamma_i(y)[N[i, y]])$ **Then**
        $N[i, y] \leftarrow N[i, y] + 1$;
        $M[x, n \cdot i + y] \leftarrow 1$;
end

3.3 Refinement of partition

Given two disjoint subsets X and Y of S, *refining* X (with respect to Y) consists in partitioning X into $X_1 \cup X_2 \cup \ldots, X_k$ in such a way that there exists a subset Y' of Y such that:

- $\forall i,\ 0 < i \leq k$, $\forall x, y \in X_i, z \in Y$, $D(x, z) = D(y, z)$.
- $\forall i, j,\ 0 < i < j \leq k$, $\forall x \in X_i, y \in X_j, z \in Y$, if $z \in Y'$, $D(x, z) \leq D(y, z)$, and if $z \in Y \setminus Y'$, $D(x, z) \geq D(y, z)$.

We remark that this operation is not always possible, for example with $X = \{x_1, x_2, x_3\}, Y = \{y_1, y_2\}$ and $D(x_1, y_1) = D(x_1, y_2) < D(x_2, y_1) = D(x_3, y_2) < D(x_3, y_1) = D(x_2, y_2)$. But if D is Robinsonian, and if, when $X \cup Y$ is sorted along a compatible order, X is an interval of $X \cup Y$, then it is possible to refine X with respect to Y. In this case, Y' is the set of the points on the left of X and $Y \setminus Y'$ is the set of the points on the right of X (or vice-versa); in addition, each X_i is an interval of X when $X \cup Y$ is sorted along a compatible order.

If Y has only one element, it is possible, by using a binary search tree, to refine X with respect to Y in $O(m \cdot k)$ time, where m is the cardinality of X and k the number of classes in the partition. If Y is empty or X has only one element, there is nothing to do. If Y and X have more than one element, the following pseudo-code refines X with respect to Y.

Procedure REFINE$((X_1, \ldots, X_p), Y)$
Input X and $Y = \{y_1, \ldots\}$ are two disjoint subsets of S.
$(X_1, \ldots, X_p)$ is a partition of X.
Output A partition $(X'_1, \ldots, X'_k)$ of X.
begin
  For $i \leftarrow 1$ **To** p **Do**
    $\Xi_i \leftarrow$ REFINE$(X_i, \{y_1\})$;
    # Ξ_i is a partition $(X_i^1, \ldots, X_i^{r_i})$ of X_i such that
    # $\forall j \in \{1, \ldots r_i\}, x, x' \in X_1^j, D(x, y_1) = D(x', y_1)$ and
    # $\forall (x_1, \ldots x_{r_i}) \in X_1^1 \times \ldots \times X_1^{r_i}, D(x_1, y_1) < D(x_2, y_1) < \ldots < D(x_{r_i}, y_1)$ or
    # $D(x_1, y_1) > D(x_2, y_1) > \ldots > D(x_{r_i}, y_1)$
    If $(p > 1)$ **and** $(r_i > 1)$ **Then**
      # This part tests and treats the case $y_1 \in Y'$
      Take x_1 in X_i^1 , x_2 in $X_i^{r_i}$, x_3 in X_{i-1}, x_4 in X_{i+1} ;
      # At least one of x_3, x_4 exists
      If $(D(x_2, y_1) < D(x_3, y_1))$ **or** $(D(x_1, y_1) > D(x_4, y_1))$ **Then**
        # In this case, $y_1 \in Y'$
        REVERSE$(X_i^1, \ldots, X_i^{r_i})$;
  $(X'_1, \ldots, X'_q) \leftarrow$ CONCAT$(\Xi_1, \ldots, \Xi_p)$;
  # $(X'_1, \ldots, X'_q)$ is a partition of X
  # $(X'_1, \ldots, X'_q) = (X_1^1, X_1^2, \ldots X_1^{r_1}, X_2^1, \ldots X_2^{r_2}, \ldots X_{p-1}^{r_{p-1}}, X_p^1, \ldots X_p^{r_p})$
  If $Y \setminus \{y_1\} = \emptyset$ **Then**
    RETURN$(X'_1, \ldots, X'_q)$
  Else
    REFINE$((X'_1, \ldots, X'_q), Y \setminus \{y_1\})$;
end

Although the definition of refining deals with two sets, the procedure REFINE takes as arguments a sequence of sets $(X_1, \ldots, X_p)$ and a set Y. This procedure is recursive and its initial call takes as arguments the (initial) sets X and Y. The recursive calls take as arguments a partition $(X_1, \ldots, X_p)$ of X and a subset of Y.

If refining X with respect to Y is not possible, this procedure returns an arbitrary partition of X. If D is Robinsonian, every time our algorithm uses the REFINE procedure, refining X with respect to Y is possible. This is not the case if D is not Robinsonian, but because of Step 5 (VERIFICATION), this does not matter.

Theorem 2. *The procedure* REFINE *runs in* $O(m^2)$ *time, where* $m = \max(|X|, |Y|)$.

Proof. Set $n = |X|$, $\overline{n} = |Y|$ and $m = \max(n, \overline{n})$. Let $X = \{x_1, x_2, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots, y_{\overline{n}}\}$.

We first consider the y_i's such that, if $(X_1, \ldots X_p)$ is the partition obtained by refining X with respect to $\{y_1, \ldots y_{i-1}\}$, for every $j \in \{1 \ldots p\}$, $\{D(y_i, x) : x \in X_j\}$ has only one value. Refining X with respect to $\{y_i\}$ takes $O(n)$ time. So treating all such y_i's takes $O(m^2)$ time.

We now consider the other y_i's, i.e. the y_i's such that, when refining X with respect to $\{y_i\}$, at least one set X_j is partitioned into many subsets. There cannot be more such y_i's than subsets in the (final) partition of X and thus there cannot be more such y_i's than elements in X. So we can suppose that $m = n$. We now prove by induction on m that the time $T(m)$ used by REFINE to treat these y_i's is $O(m^2)$, in a similar way to the one in Heun (2008).

By the induction hypothesis, there exists a constant c such that $T(r) \leq 2c \cdot r^2$ for every $r < m$. We can take c such that REVERSE, CONCAT and partitioning X into $X^1 \cup X^2 \cup \ldots X^k$ (i.e. REFINE$(X, \{y_1\})$) takes less than $c \cdot k \cdot m$ time (partitioning X takes $O(km)$ time, REVERSE and CONCAT take $O(m)$ time). So, we have:

$$T(m) \leq ckm + \sum_{i=1}^{k} T(m_i)$$

With $k > 1$, $\sum_{i=1}^{k} m_i \leq m$ (m_i is the cardinality of X^i), and thus $m_i < m$ for every i. So:

$$T(m) \leq ckm + \sum_{i=1}^{k} 2cm_i^2 \leq ckm + 2c\sum_{i=1}^{k} m_i(m - m + m_i)$$
$$\leq ckm + 2c\sum_{i=1}^{k} m_i m - 2c\sum_{i=1}^{k} m_i(m - m_i).$$

Since $x(m-x) \geq m-1$ for all x in $\{1, \ldots, m-1\}$:

$$T(m) \leq ckm + 2cm^2 - 2c\sum_{i=1}^{k}(m-1)$$
$$\leq ckm + 2cm^2 - 2ck(m-1) \leq 2cm^2.$$

□

Remark: This partition refinement technique has already been used in similar problems, such as ultrametric recognition in Heun (2008), interval graph recognition in Habib and all (2000) or doubly lexical ordering in Paige and Tarjan (1987).

4 The algorithm

In this Section, we present our algorithm in a more precise way. Let D be a dissimilarity on an n-set S and K be a fixed integer. The value of K has no incidence on the structure or the correctness of the algorithm; in Subsection 4.4, we will fix $K > 12$ for ease of proving Step 4 runs in $O(n^2)$.

4.1 Initialization (Step 1)

The aim of this step is to construct the PQ-tree $\mathcal{T}_K$ of G_K. By property 1, if D is Robinsonian, then this PQ-tree represents a set of permutations which contains all the compatible permutations.

step 1: INITIALIZATION
begin
  ForAll $x \in S$ **Do**
    Γ-CONSTRUCTION(x, K) ;
  $M \leftarrow$ VERTEX-CLIQUE-MATRIX-CONSTRUCTION(K) ;
  BOOTH-AND-LUEKER(M) ;
  If G_K is not an interval graph **Then**
    STOP ; # D is not Robinsonian
  Else
    # The algorithm has built a PQ-tree $\mathcal{T}_K$
    ADD-INFORMATION-ON-NODES$(\mathcal{T}_K)$;
end

We have used, as a sub-routine called BOOTH-AND-LUEKER, the algorithm of Booth and Lueker (1976). This algorithm takes as input the vertex-clique incidence matrix of a graph G and determines if G is an interval graph or not. In addition, if G is an interval graph, this algorithm computes the PQ-tree which represents the sets of the possible orders of the maximal clique set. We recall that, for a graph G_l, the maximal clique set is in one-to-one correspondence with S. The algorithm of Booth and Lueker runs in linear time in the size of the matrix; more precisely, it runs in $O(c + r + o)$, where c (resp. r, resp. o) is the number of columns (resp. rows, resp. ones) in the vertex-clique incidence matrix.

ADD-INFORMATION-ON-NODES associates, with each node α, S_α and $|S_\alpha|$ so that in the following steps, it will be possible to get these informations in $O(1)$. ADD-INFORMATION-ON-NODES runs in $O(n^2)$.

The loop **forall** $x \in S$ **do** Γ-CONSTRUCTION runs in $O(n^2K)$, as VERTEX-CLIQUE-MATRIX-CONSTRUCTION. The graph G_K has $n \cdot K$ vertices and n maximal cliques. So M is an $n \times (nK)$ matrix, and BOOTH-AND-LUEKER also runs in $O(n^2K)$. Since K is a constant, this step runs in $O(n^2)$.

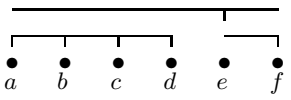
4.2 Treatment of P-nodes (Step 2)

For any P-node α, different from the root of $\mathcal{T}_K$, let $\overline{\alpha}$ be the nearest ancestor of α which is a P-node. If α has no P-node as ancestor, we take $\overline{\alpha}$ equal to the root of $\mathcal{T}_K$. We define $\overline{S_\alpha}$ as $S_{\overline{\alpha}} \setminus S_\alpha$.

step 2: TREATMENT OF P-NODES
begin
  ForAll P-node α ($\mathcal{T}_K$ is traversed in postorder) **Do**
1   REFINE($S_\alpha, \overline{S_\alpha}$) ; # We get a partition $S_\alpha = S_\alpha^1 \cup S_\alpha^2 \cup \ldots S_\alpha^{k_\alpha}$
    If ($k_\alpha > 1$) **Then**
2   Create a Q-node α' with children $(\nu_1, \nu_2, \ldots, \nu_{k_\alpha})$, each ν_i being a basic P-node with $S_{\nu_i} = S_\alpha^i$;
      ForAll child β of α such that $|S_\beta| > 1$ **Do**
        SUBTREE-INSERTION($\beta, \alpha', False$) ;
      $\alpha \leftarrow \alpha'$;
end

The aim of this step is to check if, for each P-node α, $D(x, y)$ has a constant value on S_α for each $y \in S \setminus S_\alpha$ and, if it is not the case, to transform α into a Q-node. For example, let us consider the dissimilarity D_1 (see Figure 4).

D_1	a	b	c	d	e	f
a	0	1	1	1	2	5
b		0	1	1	2	4
c			0	1	2	4
d				0	2	3
e					0	2
f						0



$\Gamma_1(a)$	1	1	1	1	0	0
$\Gamma_1(b)$	1	1	1	1	0	0
$\Gamma_1(c)$	1	1	1	1	0	0
$\Gamma_1(d)$	1	1	1	1	0	0
$\Gamma_1(e)$	1	1	1	1	1	1
$\Gamma_1(f)$	0	0	0	0	1	1
$\Gamma_0(a)$	1	0	0	0	0	0
$\Gamma_0(b)$	0	1	0	0	0	0
$\Gamma_0(c)$	0	0	1	0	0	0
$\Gamma_0(d)$	0	0	0	1	0	0
$\Gamma_0(e)$	0	0	0	0	1	0
$\Gamma_0(f)$	0	0	0	0	0	1
	a	b	c	d	e	f

Figure 4: The dissimilarity D_1, its sets $\Gamma_1(x)$ and the vertex-clique incidence matrix of $G_1(D_1)$

In Figure 4, the intervals $\Gamma_1(x)$ are represented with a vertical line above x. When $\Gamma_1(x) = \Gamma_1(y)$, the interval is represented once. At Step 1, from the vertex-clique incidence matrix of $G_1(D_1)$, the algorithm

constructs the PQ-tree of the left part of Figure 5. Then, at Step 2, the algorithms changes the P-node α into a Q-node and we get the PQ-tree on the right part of Figure 5. Notice that α, as a Q-node, will be changed again at Step 3.

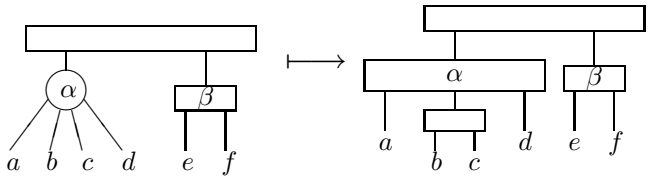


Figure 5: The PQ-tree built from the graph $G_1(D_1)$ and its transformation at Step 2

We remark that, when treating a P-node α, the algorithm tests $D(x, y)$ only for $y \in S_{\overline{\alpha}} \setminus S_\alpha$ (and x in S_α). If $\overline{\alpha}$ is not the root of $\mathcal{T}_K$, then the test for $y \in S \setminus S_{\overline{\alpha}}$ is done when treating $\overline{\alpha}$, which is also a P-node.

After Line 1, we get a partition of S_α into $S_\alpha^1 \cup S_\alpha^2 \cup \dots S_\alpha^{k_\alpha}$ such that, when S is sorted along a compatible order, $S_\alpha^1, S_\alpha^2, \dots S_\alpha^{k_\alpha}$ are, in this order, consecutive intervals of S. After Line 2, the node α' is such that after changing α into α' in $\mathcal{T}_K$, the permutations represented by $\mathcal{T}_K$ respect this condition.

If, before being treated, α is a non-basic P-node with children $\beta_1, \dots \beta_q$, then $S_\alpha = S_{\beta_1} \cup \dots \cup S_{\beta_q}$ and $S_{\beta_1}, \dots S_{\beta_q}$ are also intervals of S when S is sorted along a compatible order. The aim of the SubTree-Insertion procedure is to make $\mathcal{T}_K$ also respecting this condition. If D is Robinsonian and, if β is a descendant of α, then, we are in one of the following two cases:

1. $\exists\, i \in [1, \dots, k_\alpha]$ such that $S_\beta \subset S_\alpha^i$.
2. $\exists\, i, j \in [1, \dots, k_\alpha], i < j$ such that:
   - $S_\beta \cap S_\alpha^k = S_\alpha^k$ for $i < k < j$.
   - $S_\beta \cap S_\alpha^k = \emptyset$ for $k < i$ or $j < k$.

In Case 2, we say that $[\nu_i, \dots, \nu_j]$ is the *intersection interval* of β and, if β is a Q-node with children $\gamma_1, \dots, \gamma_p$, we can fix the orientation of the γ_i relatively to that of $S_\alpha^1, S_\alpha^2, \dots, S_\alpha^{k_\alpha}$. More precisely, up to a reversal of the γ_i, we have:
$\exists\, j_1 < j_2 < \dots j_{p+1}$ such that $\forall i \in [1, \dots, p]$:
If $k < j_i$ or $k > j_{i+1}$, $S_\alpha^k \cap S_{\gamma_i} = \emptyset$.
If $j_i < k < j_{i+1}$, $S_\alpha^k \subset S_{\gamma_i}$.

The pseudo-code of SubTree-Insertion expands as:

Procedure SubTree-Insertion(β, α, *Bool*)
Input: α is a Q-node with children $\nu_1, \ldots, \nu_p$.
β is a node with children $\gamma_1, \ldots, \gamma_k$ and such that $S_\beta \subset S_\alpha$.
Bool is a boolean which is true if the algorithm has to consider the order of the children of β.
begin
  If There exists ν_i such that $S_\beta = S_{\nu_i}$ **Then**
    $\nu_i \leftarrow \beta$;
  Else
    If There exists ν_i such that $S_\beta \subset S_{\nu_i}$ **Then**
      Suppress among the children of ν_i those whose leaves are in S_β ;
      If *Bool* **Then**
        Add β among the children of α, just before ν_i ;
      Else
        Add β among the children of ν_i ;
    Else
      # Let $[\nu_l, \ldots, \nu_m]$ be the intersection interval of β
      Create two basic P-nodes ν'_l and ν'_m, the children of ν'_l (resp. ν'_m) are those of ν_l (resp. ν_m) which have an empty intersection with S_β ; Delete these children from ν_l and ν_m ;
      Add ν'_l and ν'_m among the children of α, ν'_l just before ν_l and ν'_m just after ν_m ;
      Create a Q-node α' with children $(\nu_l, \ldots, \nu_m)$;
      Replace, in $(\nu_1, \ldots, \nu_p)$, $(\nu_l, \ldots, \nu_m)$ by α' ;
      Bool' $\leftarrow$ (β is a Q-node) ;
      If *Bool'* **and** $\neg((S_{\nu_l} \cap S_{\gamma_1} \neq \emptyset) \wedge (S_{\nu_m} \cap S_{\gamma_p} \neq \emptyset))$ **Then**
        Reverse($\gamma_1, \gamma_2, \ldots, \gamma_p$) ;
      For $i \leftarrow 1$ **To** k **Do**
        SubTree-Insertion(γ_i, α', *Bool'*) ;
      Insert(children of α, α', children of α') ;
end

For example, with the dissimilarity D_2 of Figure 6, the algorithm, at Step 1, constructs the PQ-tree of Figure 7.

D_2	a	b	c	d	e	f	g	h
a	0	2	2	2	2	2	4	6
b		0	1	1	2	2	3	6
c			0	1	2	2	3	6
d				0	2	2	3	5
e					0	1	3	5
f						0	3	4
g							0	1
h								0

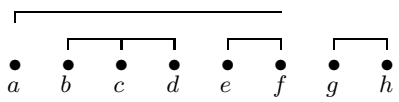


Figure 6: A dissimilarity and its sets $\Gamma_1(x)$

At Step 2, from the P-node α of the PQ-tree in Figure 7 and the last two columns of the dissimilarity of Figure 6, the algorithm constructs the P-node α' in Figure 7. Then, after SubTree-Insertion, the algorithm transforms the PQ-tree of Figure 7 into the one of Figure 8.

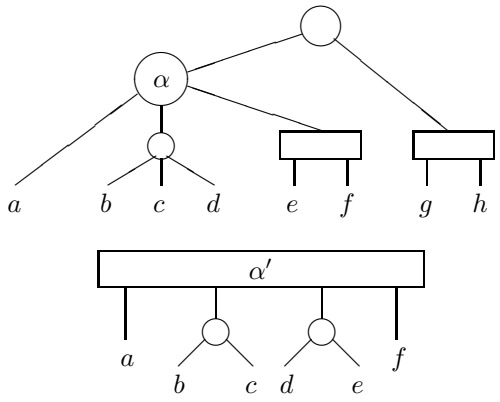


Figure 7: The PQ-tree built from the graph $G_1(D_2)$ and the Q-node α' built at Line 2 of Step 2

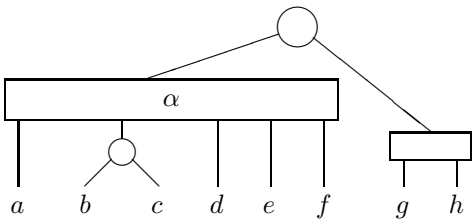


Figure 8: The PQ-tree of the dissimilarity of Figure 6 after Step 2

We now give the complexity of this step: The procedure REFINE does not change the order of the leaves. More precisely, within each S^i_α, the order of the leaves is the same as the one in α (and thus as the one in β). So, all the tests and operations in SUBTREE-INSERTION take at most linear time. There are at most $\log(|S_\beta|)$ recursive calls; thus, in step 2, the loop "**forall** *child* β *of* α *such that* $|S_\beta| > 1$ **do** SUBTREEINSERTION$(\beta, \alpha', False)$" takes $O(\log(|S_\alpha|))$. Thus the complexity of the treatment of one P-node α is the complexity of REFINE$(S_\alpha, \overline{S_\alpha})$, which is $O(p^2)$, where p is the greatest of $|S_\alpha|$, $|\overline{S_\alpha}|$. It is possible for p to be in $O(n)$, and there may exist $O(n)$ P-nodes. But if we look at step 2 as a whole, each $D(x, y)$ is used at most twice: one with x in a S_α and y in $\overline{S_\alpha}$, and one with y in a $S_{\alpha'}$ and y in $\overline{S_{\alpha'}}$. So, step 2 runs in $O(n^2)$ time. In addition, we have:

Property 7. *Let α be a P-node of $\mathcal{T}_K$ after Step 2. If D is Robinsonian with PQ-tree $\mathcal{A}$, then there exists a node α' of $\mathcal{A}$ such that $S_{\alpha'} = S_\alpha$.*

Proof. Let α' be the node of $\mathcal{A}$ such that $S_\alpha \subset S_{\alpha'}$ and such that none of its children has this property. Let us suppose that the claim is false, i.e. that $S_\alpha \neq S_{\alpha'}$. For any $x \in S \setminus S_\alpha$, $D(x, y)$ has a constant value for $y \in S_\alpha$. So, for every permutation represented by $\mathcal{A}$, the reversal of S_α is possible. Thus α' is a P-node with children $\beta_1, \ldots \beta_k, \ldots, \beta_p$, with $k > 1$ and $S_\alpha = S_{\beta_1} \cup \ldots \cup S_{\beta_k}$. There exists d_α such that, for every $i \neq j \in \{1, \ldots, p\}, x \in S_{\beta_i}, y \in S_{\beta_j}, D(x, y) = d_\alpha$. Let $x \in S_\alpha$, we set $I = \{y \in S, D(x, y) \leq d_\alpha\}$. The Γ-interval I is contained in all the Γ-intervals containing S_α; in addition, $S_{\alpha'} \subset I$. The Γ-interval I has been considered to construct $\mathcal{T}_K$ (otherwise, α would be the root of $\mathcal{T}_K$). So (as $S_\alpha \neq I$), there exist $x \in S$, $d > 0$ such that the Γ-interval $J = \{y \in S, D(x, y) \leq d\}$ contains points of $S_{\alpha'} \setminus S_\alpha$ and points of $S \setminus S_{\alpha'}$ and is such that $J \cap S_\alpha = \emptyset$. So, there exist $y \in S \setminus S_{\alpha'}$, $z \in S_{\alpha'}$ such that $D(y, z) \leq d$. By Property 6, $D(y, z') \leq d$ for every $z' \in S_{\alpha'}$, which contradicts $J \cap S_\alpha = \emptyset$. □

4.3 Treatment of Q-nodes (Step 3)

This step aims at checking, for every Q-node α whether α fulfills Property 6. For this step, we will consider all nodes with two children as Q-nodes. We first give some definitions and properties.

Let α be a Q-node and let β be a node which is not a descendant of α (i.e. either $S_\alpha \subset S_\beta$ or $S_\alpha \cap S_\beta = \emptyset$). We say that α is *orientable* with respect to β if there exist x, y in S_α and z in $S_\beta \setminus S_\alpha$ such that $D(x,z) \neq D(y,z)$.

Let α be a Q-node with children $(\gamma_1, \ldots, \gamma_p)$. The two *extreme points* a_α and b_α of α are defined as follows:

- if γ_1 (resp. γ_p) is a leaf x, then x is a_α (resp. b_α).
- If γ_1 (resp. γ_p) is a P-node, we take for a_α (resp. b_α) any point of S_{γ_1} (resp. S_{γ_p}).
- If γ_1 (resp. γ_p) is a Q-node with extreme points x and y, we choose one of them for a_α (resp. b_α).

Claim 3. *If a Q-node α is orientable with respect to a node β, then there exists z in $S_\beta \setminus S_\alpha$ such that $D(a_\alpha, z) \neq D(b_\alpha, z)$.*

Proof. For any z in $S_\beta \setminus S_\alpha$, and any x, y in S_α, either $D(a_\alpha,z) \leq D(x,z) \leq D(b_\alpha,z)$ and $D(a_\alpha,z) \leq D(y,z) \leq D(b_\alpha,z)$ or $D(a_\alpha,z) \geq D(x,z) \geq D(b_\alpha,z)$ and $D(a_\alpha,z) \geq D(y,z) \geq D(b_\alpha,z)$. In both cases, if $D(x,z) \neq D(y,z)$, then $D(a_\alpha,z) \neq D(b_\alpha,z)$. □

For any pair of Q-nodes $\{\alpha, \beta\}$, let us compare $D(a_\alpha,a_\beta), D(a_\alpha,b_\beta), D(b_\alpha,a_\beta), D(b_\alpha,b_\beta)$. If S_α and S_β are disjoint, then, up to symmetry, we are in one of the following cases:

- $D(a_\alpha,a_\beta) = D(a_\alpha,b_\beta) = D(b_\alpha,a_\beta) = D(b_\alpha,b_\beta)$
  Then α and β are not orientable with respect to each other.

- $D(a_\alpha,a_\beta) < D(a_\alpha,b_\beta)$ and $D(b_\alpha,a_\beta) < D(a_\alpha,a_\beta)$
  Then both α and β are orientable with respect to the other. In this case, if D is Robinsonian, for any ordering of S compatible with D, the quadruplet $\{a_\alpha, b_\alpha, a_\beta, b_\beta\}$ is ordered as $(a_\alpha, b_\alpha, a_\beta, b_\beta)$ or $(b_\beta, a_\beta, b_\alpha, a_\alpha)$.

- $D(a_\alpha,a_\beta) = D(b_\alpha,a_\beta) < D(a_\alpha,b_\beta) = D(b_\alpha,b_\beta)$
  In this case, β is orientable with respect to α : if D is Robinsonian, for any ordering of S compatible with D, the quadruplet $\{a_\alpha, b_\alpha, a_\beta, b_\beta\}$ is ordered as $(a_\alpha, b_\alpha, a_\beta, b_\beta)$, $(b_\alpha, a_\alpha, a_\beta, b_\beta)$, $(b_\beta, a_\beta, b_\alpha, a_\alpha)$ or $(b_\beta, a_\beta, a_\alpha, b_\alpha)$.

  It could be possible, even when D is Robinsonian, that, for a point x of $S_\beta \setminus \{a_\beta, b_\beta\}$, we have $D(x,b_\alpha) < D(x,a_\alpha)$ (for example, with $S_\alpha = \{a_\alpha, b_\alpha\}$, $S_\beta = \{a_\beta, x, b_\beta\}$, $D(a_\alpha,a_\beta) = D(b_\alpha,a_\beta) = 1$, $D(a_\alpha,b_\beta) = D(b_\alpha,b_\beta) = 4$, $D(x,b_\alpha) = 2$ and $D(x,a_\alpha) = 3$). If this is the case, α is also orientable relatively to β, that is to say that, if D is Robinsonian, for any ordering of S compatible with D, $\{a_\alpha, b_\alpha, a_\beta, b_\beta\}$ is ordered as $(a_\alpha, b_\alpha, a_\beta, b_\beta)$ or $(b_\beta, a_\beta, b_\alpha, a_\alpha)$. We then say that α is orientable *via an internal point* of β. If α is orientable via an internal point of β, we define the dissimilarity D' on S by:

  - if $x \neq a_\alpha$ and $y \neq a_\beta$, then $D'(x,y) = D(x,y)$
  - $D'(a_\alpha,a_\beta) = D'(a_\alpha,a_\beta) + \epsilon$, where ϵ is a positive real, small enough not to interfere with the other values of D. For instance, if all the values of D are integer, we can take $\epsilon = 0.1$.

Claim 4. *D' is Robinsonian if and only if D is Robinsonian; in addition, if D and D' are Robinsonian, they have the same set of compatible permutations.*

The advantage of D' upon D is that, at Step 4, the algorithm needs to consider only the extreme points of the sets S_α.
If S_β is a subset of S_α, then, up to symmetry, we are in one of the following cases:

- If β is not an extreme child of α and if $D(a_\alpha, a_\beta) < D(a_\alpha, b_\beta)$ or $D(b_\alpha, a_\beta) > D(v_\alpha, b_\beta)$, then β is orientable with respect to α. In this case, it is possible to *delete* β from $\mathcal{T}$, that is to say to replace, among the children of α, β by its children (see Figure 9).

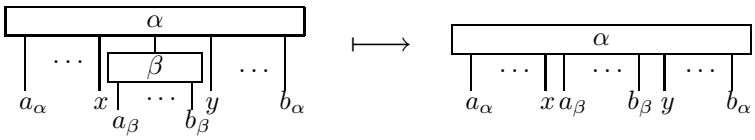


Figure 9: Deletion of a child of a Q-node

This transformation corresponds to INSERT(children of α, β, children of β).

- $D(a_\alpha, a_\beta) = D(a_\alpha, b_\beta)$ and $D(b_\alpha, a_\beta) = D(b_\alpha, b_\beta)$.
It may be possible that, for a point x in $S_\alpha \setminus (S_\beta \cup \{a_\alpha, b_\alpha\})$, $D(x, a_\beta) \neq D(x, b_\beta)$; in fact, we are then in the previous case and it is possible to delete β from $\mathcal{T}$. As before, we shall say that β is orientable *via an internal point.*

- If β is an extreme child of α, for instance the last one, and if $D(a_\alpha, a_\beta) < D(a_\alpha, b_\beta)$, then, as in the first case, β is orientable with respect to α and it is possible to delete β from $\mathcal{T}$ via the transformation of Figure 10. After this transformation, b_α has to be equal to b_β, although, before the transformation,

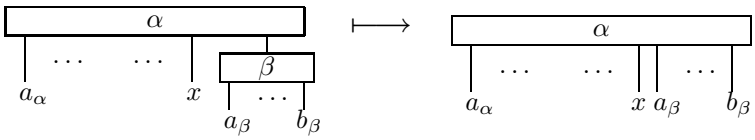


Figure 10: Deletion of a extreme child of a Q-node

b_α can be equal to a_β or b_β.

- β is an extreme child of α, for instance the last one, and $D(a_\alpha, a_\beta) = D(a_\alpha, b_\beta)$.
It may be possible that, for a point x in $S_\alpha \setminus (S_\beta \cup \{a_\alpha\})$, $D(x, a_\beta) \neq D(x, b_\beta)$; we shall say that β is orientable with respect to α via an internal point. In fact, we are then in the previous case and it is possible to delete β from $\mathcal{T}$.

We have the following properties:

Claim 5. *Let α be a Q-node orientable with respect to a P-node β. Then, for any $x \in S_\beta$, $D(x, a_\alpha) \neq D(x, b_\alpha)$.*

Proof. After Step 2, both $D(x, a_\alpha)$ and $D(x, b_\alpha)$ has a constant value for all $x \in S_\beta$. □

Claim 6. *Let α be a Q-node, child of a P-node β and γ be a node such that $S_\gamma \cap S_\beta = \emptyset$, then α is not orientable with respect to γ. In addition, if α is orientable, it is orientable with respect to β or to a descendant of β.*

Proof. Since a_α and b_α are in S_β, after Step 2, for every x in $S \setminus S_\beta$, $D(x, a_\alpha) = D(x, b_\alpha)$. □

Claim 7. *Let α be a Q-node orientable with respect to a node β and γ be an ancestor of β. Then α is orientable with respect to to γ.*

Proof. As α is orientable with respect to β, there exists $x \in S_\beta$ such that $D(a_\alpha, x) \neq D((b_\alpha, x)$. Since $S_\beta \subset S_\gamma$, $x \in S_\gamma$ and α is orientable with respect to γ. □

Property 8. *Let α and β be Q-nodes, α being a child of β. If α is orientable (with respect to β or to another node), then it can be deleted.*

Proof. If α is orientable with respect to a child of β, then, by Claim 7, α is orientable with respect to β and thus can be deleted from $\mathcal{T}$. If α is orientable with respect to a node γ and not with respect to β, there exists z in $S_\gamma \setminus S_\beta$ such that $D(a_\alpha, z) \neq D(b_\alpha, z)$ (let us suppose, for instance, that $D(a_\alpha, z) < D(b_\alpha, z)$). As a_α and b_α are in S_β, β is orientable with respect to γ. So, we have either $D(a_\beta, z) \leq D(a_\alpha, z) < D(b_\alpha, z) \leq D(b_\beta, z)$ or $D(b_\beta, z) \leq D(a_\alpha, z) < D(b_\alpha, z) \leq D(a_\beta, z)$. In both cases, α can be deleted from $\mathcal{T}$. □

We recall that a node α is said to be *K-big* if $|S_\alpha| > K$. A node β, child of a P-node α, is said to be *K-closed* if $\Gamma_K(x) \not\subset S_\beta$ for all $x \in S_\beta$; otherwise, β is said to be *K-free*.

Property 9. *Let α be an orientable Q-node. If α cannot be deleted from $\mathcal{T}$, then:*
- *α is a K-big child of a P-node γ.*
- *γ has a K-big child β such that α is orientable with respect to β.*

Proof. By Property 8, α is the child of a P-node γ. By Claims 6 and 7 α is orientable with respect to γ. Let x in S_γ be such that $D(x, a_\alpha) \neq D(x, b_\alpha)$, there exists a child β of γ such that $x \in S_\beta$; α is orientable with respect to β.
Let us suppose that β is a K-closed node, then, by Property 5, for every y in S_β, $D(y, z)$ has a constant value on $S_\gamma \setminus S_\beta$. It would be impossible that $D(x, a_\alpha) \neq D(x, b_\alpha)$. So β is a K-free child of γ; by Property 4, β is K-big.
Let us suppose that α is a K-closed node, $D(a_\alpha, x)$ and $D(b_\alpha, x)$ are K-known, and they are different. So, among the permutations which are compatible with the K-known values of D, we can have $(a_\alpha < b_\alpha < x)$ or $(b_\alpha < a_\alpha < x)$ but not both. But after step 1, $\mathcal{T}$ represents the set of permutations of S which are compatible with the K-known values of D. For these orders, we can reverse the a_α and b_α without changing the place of x. So both $(a_\alpha < b_\alpha < x)$ and $(b_\alpha < a_\alpha < x)$ appear in the permutations compatible with $\mathcal{T}$. So α is a K-free child of γ; by property 4, α is K-big. □

We can now give the pseudo-code of step 3:

step 3: Treatment of Q-Nodes
Output: A subset S' of S.
begin
  Compute-Extreme-Points ;
  ForAll Q-node α (# $\mathcal{T}$ is traversed in postorder) **Do**
    Q-Node-Examination(α) ;
end

Compute-Extreme-Points computes a_α and b_α for each Q-node α and Q-Node-Examination deletes Q-nodes that can be deleted or changes D when a Q-node is orientable via an internal point.

We now give an example to illustrate this step. If we consider the dissimilarity D_1 of Figure 4 and its PQ-tree after Step 2 (see Figure 5), at Step 3, since $D_1(b, f) < D_1(a, f)$ (a and b are the extreme points of α) and $D_1(a, e) < D_1(a, f)$, both α and β are deleted and this PQ-tree is transformed into the PQ-tree of Figure 11.

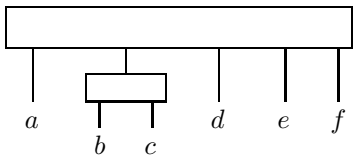


Figure 11: The PQ-tree for the dissimilarity D_1 after Step 3

We now consider the dissimilarity D_3 (see Figure 12). At Step 1, the algorithm builds the PQ-tree of Figure 13, which remains unchanged at Step 2. At Step 3, since α_2 is orientable via an internal point of α_1 (namely b), the dissimilarity D_3 is changed into the one of Figure 13.

D_3	a	b	c	d	e	f
a	0	1	2	5	5	5
b		0	1	3	4	5
c			0	3	3	3
d				0	1	2
e					0	1
f						0

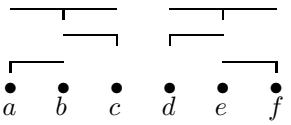
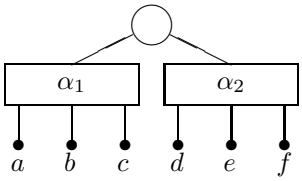


Figure 12: A dissimilarity and its sets $\Gamma_1(x)$



D_3	a	b	c	d	e	f
a	0	1	2	5	5	**5.1**
b		0	1	3	4	5
c			0	3	3	3
d				0	1	2
e					0	1
f						0

Figure 13: The PQ-tree built from the graph $G_1(D_3)$ and the dissimilarity D_3 after Step 3

In worst case, Q-Node-Examination(α) compares $D(x, a_\alpha)$ and $D(x, b_\alpha)$ for all x in $S \setminus S_\alpha$. So Q-Node-Examination runs in $O(n)$ time. As $\mathcal{T}$ cannot have more than n Q-nodes, this step runs in $O(n^2)$ time.

4.4 Recursive Calls and Merging (Step 4)

This step aims at orienting Q-nodes if applicable. An orientable Q-node is the child of a P-node, and, by Property 7, these P-nodes can be treated independently. We denote by $D|_{S'}$ the dissimilarity D applied on a subset S' of S. The pseudo-code of this step is the following:

step 4: Recursive Calls and Merging
begin
  ForAll K-big P-node α (# $\mathcal{T}$ is traversed in postorder) **Do**
    $S'_\alpha \leftarrow$ Representatives(α) ;
    If $|S'_\alpha| > 2$ **Then**
      # Otherwise, there is no Q-node to orient
      Apply the whole algorithm (steps 1—5) to $D|_{S'_\alpha}$;
      If $D|_{S'_\alpha}$ is not Robinsonian **Then**
        Stop ; # D is not Robinsonian
      Else
        # $\mathcal{T}'$ is the PQ-tree of $D|_{S'_\alpha}$
        Merging($\alpha, \mathcal{T}'$) ;
end

Representatives(α) constructs a set S'_α which contains:

- a_β and b_β for each K-big Q-node β which is a child of α.
- An element of S_γ for each K-big P-node γ which is a child of α.

By Properties 4 and 5, there is no need to consider K-small children of α (if D is Robinsonian and β is a K-small child of α, then β is a node of the PQ-tree $\mathcal{A}$ of D; in addition, if α' is the node of $\mathcal{A}$ such that $S_{\alpha'} = S_\alpha$, then β is (in $\mathcal{A}$) a child of α'). Since D has been changed at Step 3, by Claim 4, it is sufficient to consider $D|_{S'_\alpha}$. The aim of MERGING is to rearrange $\mathcal{T}(\alpha)$ according to what has been done during the recursive call on α. Its pseudo-code is :

Procedure MERGING($\alpha, \mathcal{T}'$)
Input α is a P-node.
$\mathcal{T}'$ is a PQ-tree on a subset of S_α.
begin
  If Not all the subtrees of $\mathcal{T}'$ are leaves or Q-nodes with two leaves a_β and b_β, where β is a Q-node of $\mathcal{T}$ **Then**
    # Otherwise, no Q-node has been oriented
    ForAll K-big child γ of α **Do**
      If γ is a P-node **Then**
        # Let x be the representative of γ, x is a leaf of $\mathcal{T}'$
        Replace, in $\mathcal{T}'$, x by γ ;
      If γ is a Q-node **Then**
        # Let a_γ and b_γ be the representatives of γ
        # a_γ and b_γ are leaves of $\mathcal{T}'$ and are consecutive children of a Q-node ξ
        # Let $\nu_1, \ldots, \nu_p$ be the children of γ
        Replace, among the children of ξ, a_γ, b_γ by $\nu_1, \ldots, \nu_p$;
    If α has K-small children $\eta_1, \ldots, \eta_q$ **Then**
      If The root of $\mathcal{T}'$ is a P-node or a Q-node with two children **Then**
        Add $\eta_1, \ldots, \eta_q$ among the children of the root of $\mathcal{T}'$;
        # If the root of $\mathcal{T}'$ is a Q-node with two children, it becomes a P-node
        Replace α by $\mathcal{T}'$;
      Else
        Delete the K-big children of α ;
        Add $\mathcal{T}'$ among the children of α ;
    Else
      Replace α by $\mathcal{T}'$;
end

We now give the complexity of this step.
MERGING($\alpha, \mathcal{T}'$) runs in $O(|S_\alpha|)$ time, so, at this step, the calls of MERGING not in recursive calls are, altogether, in $O(n^2)$.
Let us calculate the total number of representatives.

There are less than n/K K-big nodes with no K-big descendants, and thus less than n/K K-big nodes with more than two K-big descendants such that none of the two is a descendant of the other.

Let α be a node with only one K-big descendant β. If both α and β have representatives, then the parent of β is a P-node with more than two children and α is a Q-node. So $S_\alpha \setminus S_\beta$ contains at least three elements (see Figure 14).

Since a Q-node has at most two representatives, and a P-node at most one :

$$\sum_\alpha |S'_\alpha| \leq 2 \times n/3 + 4 \times n/K$$

We shall see below that step 5 (as steps 1— 3) runs in $O(n^2)$ time. By taking $K > 12$, we get $\sum_\alpha |S'_\alpha| < 0.98 \cdot n$. Thus step 4 runs in $O(n^2)$ time, and so does the whole algorithm.

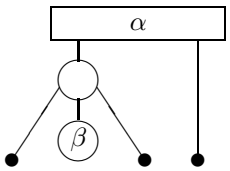


Figure 14:

4.5 Verification (Step 5)

If D is Robinsonian, after Step 4, $\mathcal{T}$ represents the set of all the permutations compatible with D and any permutation represented by $\mathcal{T}$ corresponds to a linear ordering. Conversely, if D is not Robinsonian, Steps 1—4 of the algorithm may produce a PQ-tree: for instance, in Step 3, we do not check if there is some contradiction between the orientations of the Q-nodes. In this case, no permutation on S corresponds to a linear ordering. So, the algorithm has to check if D is actually Robinsonian.

To do that, the algorithm has only to choose a permutation among the ones compatible with $\mathcal{T}$ and verify that it corresponds to a linear ordering, that is to say to verify that every row and every column is not decreasing when starting from the diagonal. This step runs in $O(n^2)$ time.

5 A complete example

We now show on an example how our algorithm runs. Let us consider the following dissimilarity:

D	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}	x_{19}
x_1	0	9	*2*	11	6	11	6	6	9	11	6	11	6	*5*	11	11	9	11	6
x_2		0	9	11	*2*	11	*3*	6	*1*	11	6	11	6	9	11	11	*1*	11	3
x_3			0	11	6	11	6	6	9	11	6	11	6	*1*	11	11	9	11	6
x_4				0	11	8	11	11	11	8	11	8	11	11	*1*	8	11	*2*	11
x_5					0	11	*3*	4	*2*	11	4	11	4	6	11	11	*2*	11	*1*
x_6						0	11	11	11	*1*	11	5	11	11	6	*3*	11	6	11
x_7							0	4	*3*	11	4	11	4	6	11	11	*3*	11	*2*
x_8								0	5	11	*1*	11	*3*	*4*	11	11	5	11	4
x_9									0	11	5	11	6	9	11	11	*1*	11	3
x_{10}										0	11	5	11	11	7	*2*	11	6	11
x_{11}											0	11	*2*	*4*	11	11	5	11	4
x_{12}												0	11	11	*2*	5	11	*1*	11
x_{13}													0	*4*	11	11	6	11	4
x_{14}														0	11	11	9	11	6
x_{15}															0	7	11	*1*	11
x_{16}																0	11	7	11
x_{17}																	0	11	3
x_{18}																		0	11
x_{19}																			0

We take, for this example, $K = 2$. The intervals $\Gamma_1(x_i)$ and $\Gamma_2(x_i)$ are represented in Figure 15:

In Figure 15, the intervals $\Gamma_1(x_i)$ and $\Gamma_2(x_i)$ are represented with a vertical line above x_i. When $\Gamma_1(x_i) = \Gamma_1(x_j)$ (or $\Gamma_2(x_i) = \Gamma_2(x_j)$), the interval is represented once, but if $\Gamma_1(x_i) = \Gamma_2(x_j)$, the interval is represented twice. To construct these intervals (or, more precisely, the vertex-clique incidence matrix of $G_2(D)$), we have used the values in italic of the matrix D.

- At Step 1 (from the vertex-clique incidence matrix of $G_2(D)$), we construct the PQ-tree $\mathcal{T}$ (see Figure 16).
- At Step 2:

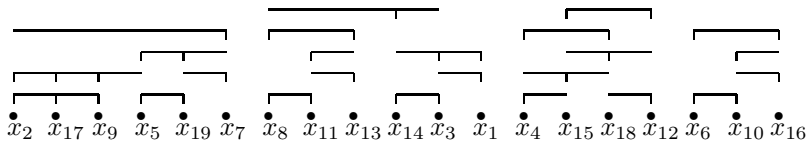


Figure 15: The sets $\Gamma_1(x_i)$ and $\Gamma_2(x_i)$

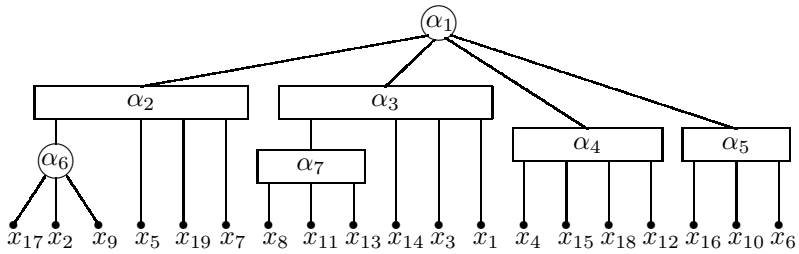


Figure 16: The PQ-tree $\mathcal{T}$ after Step 1

– On α_6, because of $D(x_2, x_{11}) > D(x_{17}, x_{11}) = D(x_9, x_{11}) \geq D(x_5, x_{11})$, we do the transformation of Figure 17.

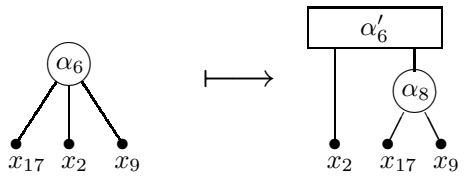


Figure 17:

– Since for all $i \neq 17, 9$, $D(x_i, x_{17}) = D(x_i, x_9)$, α_8 remains a P-node.

- At Step 3:
  – Since $D(x_2, x_8) > D(x_9, x_8)$, α'_6 is orientable with respect to α_3 and, by property 8, it can be deleted from $\mathcal{T}$. As $D(x_2, x_8) > D(x_5, x_8)$, α_2 is transformed into the Q-node of Figure 18.

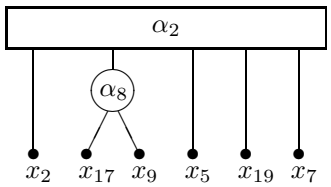


Figure 18:

  – Since $D(x_9, x_8) < D(x_9, x_{13})$, α_7 is orientable. As its parent is a Q-node, it can be deleted from $\mathcal{T}$. As $D(x_9, x_8) < D(x_9, x_1)$, α_3 is transformed into the Q-node of Figure 19.
  – Since $D(x_4, x_6) = D(x_4, x_{16}) > D(x_{12}, x_6) = D(x_{12}, x_{16})$, α_4 is orientable and, in order to check if α_5 is orientable, we have to consider the interior points of S_{α_4}. Since $D(x_{15}, x_6) < D(x_{15}, x_{16})$, α_5 is orientable and the algorithm adds 0.1 to the value of $D(x_4, x_{16})$.

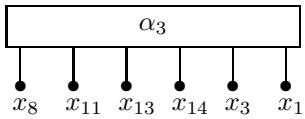


Figure 19:

– Since

$$D(x_2,x_4) = D(x_2,x_{12}) = D(x_7,x_4) = D(x_7,x_{12})$$
$$D(x_2,x_6) = D(x_2,x_{16}) = D(x_7,x_6) = D(x_7,x_{16})$$
$$D(x_8,x_4) = D(x_8,x_{12}) = D(x_1,x_4) = D(x_1,x_{12})$$
$$D(x_8,x_6) = D(x_8,x_{16}) = D(x_1,x_6) = D(x_1,x_{16})$$
$$(D(x_2,x_8) < D(x_2,x_1)) \wedge (D(x_2,x_1) > D(x_7,x_1))$$

nothing else has to be done at this step.

- At Step 4:
  $\alpha_2, \alpha_3, \alpha_4$ and α_5 are K-big, and they are all children of the P-node α_1.
  $S'_{\alpha_1} = \{x_1, x_2, x_4, x_6, x_7, x_8, x_{12}, x_{16}\}$.
  $D|_{S'_{\alpha_1}}$ and the sets $\Gamma_1(x_i)$ and $\Gamma_2(x_i)$ are represented in Figure 20.

$D\|_{S'_{\alpha_1}}$	x_1	x_2	x_4	x_6	x_7	x_8	x_{12}	x_{16}
x_1	0	*9*	11	11	*6*	*6*	11	11
x_2		0	11	11	*3*	*6*	11	11
x_4			0	*8*	11	11	*8*	*8.1*
x_6				0	11	11	*5*	*3*
x_7					0	*4*	11	11
x_8						0	11	11
x_{12}							0	*5*
x_{16}								0

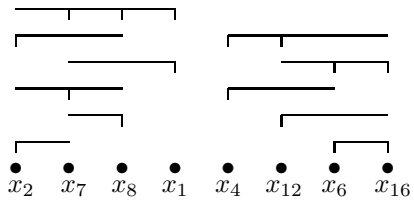


Figure 20: The dissimilarity $D|_{S'_{\alpha_1}}$ and its sets $\Gamma_1(x)$ and $\Gamma_2(x)$

– We first construct the PQ-tree $\mathcal{T}'$ of $G_2(D|_{S'_{\alpha_1}})$ (see Figure 21).

– On $D|_{S'_{\alpha_1}}$, nothing has to be done at Steps 2 and 3.

– At Step 4, the algorithm works on $D|_{S'_{\beta_1}}$, where $S'_{\beta_1} = \{x_2, x_1, x_4, x_{16}\}$. The sets $\Gamma_1(x_i)$ and $\Gamma_2(x_i)$ (for $D|_{S'_{\beta_1}}$) and the PQ-tree $\mathcal{T}$" of $G_2(D|_{S'_{\beta_1}})$ are represented in Figure 22. The PQ-tree $\mathcal{T}$" is not transformed by the algorithm, and so $\mathcal{T}''$ is (definitively) the PQ-tree of $D|_{S'_{\beta_1}}$. The merging of $\mathcal{T}'$ with $\mathcal{T}''$ does not change $\mathcal{T}'$ which is thus the PQ-tree of $D|_{S'_{\alpha_1}}$. The merging of $\mathcal{T}$ with $\mathcal{T}'$ induces the following transformations on $\mathcal{T}$:

  * α_2 and α_3 are transformed into the Q-node of Figure 23.
  * α_4 and α_5 are transformed into the Q-node of Figure 24.

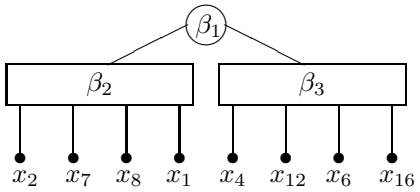


Figure 21: The PQ-tree $\mathcal{T}'$ for $D|_{S'_{\alpha_1}}$ after Step 1

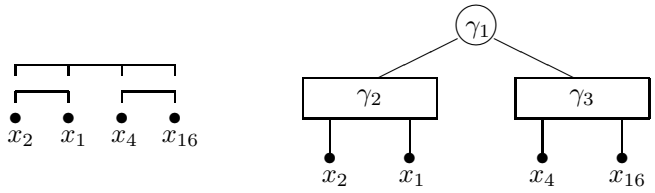


Figure 22: The sets $\Gamma_1(x)$ and $\Gamma_2(x)$ for $D|_{S'_{\beta_1}}$ and the PQ-tree of $D|_{S'_{\beta_1}}$

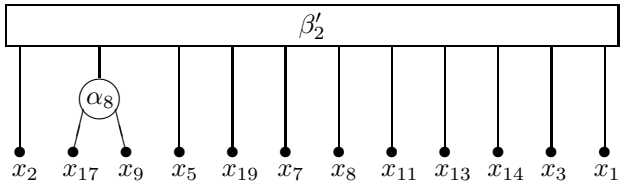


Figure 23:

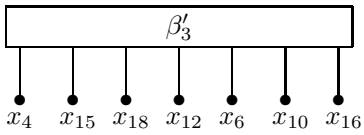


Figure 24:

So, we obtain the PQ-tree of Figure 25 as a potential PQ-tree for D.

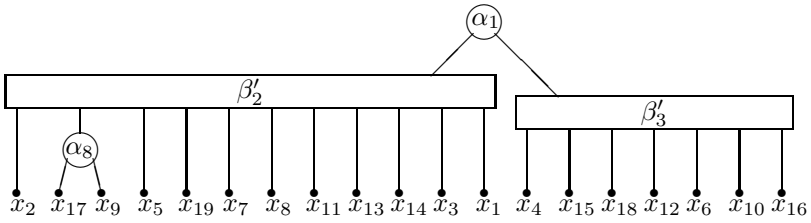


Figure 25: The PQ-tree $\mathcal{T}$ after Step 4

- At step 5, the algorithm checks that one of the permutations represented by the PQ-tree of Figure 25 corresponds to a linear order. This is the case because, when $\{x_1, x_2, \ldots, x_{19}\}$ is sorted along the permutation $(x_2, x_{17}, x_9, x_5, x_{19}, x_7, x_8, x_{11}, x_{13}, x_{14}, x_3, x_1, x_4, x_{15}, x_{18}, x_{12}, x_6, x_{10}, x_{16})$, the matrix D has all its rows and

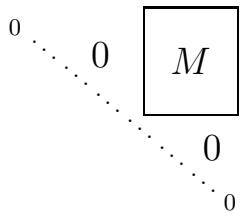
columns not decreasing from the diagonal:

D	x_2	x_{17}	x_9	x_5	x_{19}	x_7	x_8	x_{11}	x_{13}	x_{14}	x_3	x_1	x_4	x_{15}	x_{18}	x_{12}	x_6	x_{10}	x_{16}
x_2	0	1	1	2	3	3	6	6	6	9	9	9	11	11	11	11	11	11	11
x_{17}		0	1	2	3	3	5	5	6	9	9	9	11	11	11	11	11	11	11
x_9			0	2	3	3	5	5	6	9	9	9	11	11	11	11	11	11	11
x_5				0	1	2	4	4	4	6	6	6	11	11	11	11	11	11	11
x_{19}					0	2	4	4	4	6	6	6	11	11	11	11	11	11	11
x_7						0	4	4	4	6	6	6	11	11	11	11	11	11	11
x_8							0	1	3	4	6	6	11	11	11	11	11	11	11
x_{11}								0	2	4	6	6	11	11	11	11	11	11	11
x_{13}									0	4	6	6	11	11	11	11	11	11	11
x_{14}										0	1	5	11	11	11	11	11	11	11
x_3											0	2	11	11	11	11	11	11	11
x_1												0	11	11	11	11	11	11	11
x_4													0	1	2	8	8	8	8
x_{15}														0	1	2	6	7	7
x_{18}															0	1	6	6	7
x_{12}																0	5	5	5
x_6																	0	1	3
x_{10}																		0	2
x_{16}																			0

6 Some other applications

Let M be an $n \times k$ nonnegative matrix containing m nonzeros. The *doubly lexical ordering problem* consists to permute the rows and the columns of M in such a way that both the row vectors and the column vectors are in nondecreasing lexicographic order (see Lubiw 1985). There exists an $O(m \cdot \log(n+k)+k)$ algorithm for this problem (see Paige and Tarjan 1987). When M is a 0-1 matrix, the same problem can be solved in $O(n \cdot k)$ time (see Spinrad 1993).

We consider here a strict version of doubly lexical ordering: we say that M admits a *2D-ordering* if it is possible to permute the rows and the columns of M in such a way that all the rows and columns are in nondecreasing order. We remark that, contrary to the doubly lexical ordering, some matrices do not admit a 2D-ordering (for instance $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$). Given an $n \times k$ positive matrix M, the following $(n+k) \times (n+k)$ dissimilarity D_M is Robinsonian if and only if M admits a 2D-ordering:



In addition, if D_M is Robinsonian, since M is positive, for any compatible permutation, the first n entries are not mixed with the last k ones. So, the permutations (on the $n+k$ entries) compatible with this Robinsonian dissimilarity are exactly the ones for which M is 2D-ordered.

So, our algorithm can be adapted to this problem:
Given an $(n \times k)$ non negative matrix M:

1. Check if the dissimilarity $D(M) = \begin{bmatrix} 0 & M+E \\ M^t+E & 0 \end{bmatrix}$ is Robinsonian (E is the matrix defined by setting $E[i,j]=1$ for all i,j).
2. If $D(M)$ is Robinsonian, then any compatible ordering of $D(M)$ corresponds to a 2D-ordering of M. Otherwise, a 2D-ordering of M does not exist.

This algorithm solves the 2D-ordering problem in $O((n+k)^2)$, which is optimal in worst case (i.e. when M is dense and $n \approx k$).

An *ultrametric* is a distance D on a set S such that for any x, y and z in S, $D(x,y) \leq \max\{D(x,z), D(y,z)\}$. Ultrametrics have many applications in classification because they are in one-to-one correspondence with hierarchies (see Barthélemy and Guénoche 1991 and Critchley and Fichet 1994). Ultrametrics are a particular case of Robinsonian dissimilarities. More precisely, they are the Robinsonian dissimilarities for which the PQ-tree has only P-nodes (see Seston 2008b). Our algorithm can be adapted to ultrametrics recognition:
Given a dissimilarity D:

1. Check if D is Robinsonian.
2. If D is Robinsonian, verify that its PQ-tree has only P-nodes.

This algorithm runs in $O(n^2)$, where n is the number of elements of S, which is optimal.

Let D be a dissimilarity on an n-set S, and k an integer in $[0, \ldots, n-1]$.
We define the dissimilarity D^k by:
$D^k(x,y) = D(x,y)$ if $D(x,y)$ is k-known (i.e. if $x \in \Gamma_k(y)$ or $y \in \Gamma_k(x)$).
$D^k(x,y) = \infty$ otherwise.
We say that D is *k-Robinsonian* if D^k is Robinsonian. Equivalently, D is k-Robinsonian if its graph G_k is an interval graph. This generalizes Robinsonian dissimilarities, since $(n-1)$-Robinsonian is equivalent to Robinsonian. In addition, every dissimilarity is 0-Robinsonian and if a dissimilarity D is k-Robinsonian, then D is $(k-1)$-Robinsonian. So, with a dichotomic search, it is possible to determine, in $O(n^2 \log n)$, the greatest k such that D is k-Robinsonian.

References

J.P. Barthélemy and F. Brucker (2003), "NP-Hard Approximation Problems in Overlapping Clustering", *Journal of Classification* 18, 159–183.

J.P. Barthélemy and A. Guénoche (1991), *Trees and Proximity Representations*, J. Wiley & sons.

S. Benzer (1962), "The Fine Structure of the Gene", *Scientific American* 206:1, 70–84.

M.W. Berry, B. Hendrickson and P. Raghavan (1996), "Sparse Matrix Reordering Schemes for Browsing Hypertext", in *The Mathematics of Numerical Analysis*, J. Renegar, M. Shub and S. Smale Eds., pp 99–123, American Mathematical Society, Lectures in Applied Mathematics.

L. Boneva (1980), "Seriation with Applications in Philology", in *Mathematical Statistics*, R. Bartoszyński, J. Koronacki and R. Zieliński Eds., pp 73–82, PWN Polish Scientific Publishers.

K.S. Booth (1975), *PQ-Tree Algorithms*, Ph.D. Thesis, University of California.

K.S. Booth and G.S. Lueker (1976), "Testing for the Consecutive Ones Property, Interval Graphs and Graph Planarity Using PQ-Tree Algorithm", *Journal of Computer and System Sciences* 13, 335–379.

M.J. Brusco (2002), "A Branch-and-Bound Algorithm for Fitting Anti-Robinson Structures to Symmetric Dissimilarity Matrices", *Psychometrica* 67, 459–471.

G. Caraux and S. Pinloche (2005), "PermutMatrix: a Graphical Environment to Arrange Gene Expression Profiles in Optimal Linear Order", *Bioinformatics* 21, 1280–1281.

C.H. Chen, H.G. Hwu, W.J. Jang, C.H. Kao, Y.J. Tien, S. Tzeng and H.M. Wu (2004), "Matrix Visualisation and Information Mining", *COMPSTAT'2004*, Prague.

V. Chepoi and B. Fichet (1997), "Recognition of Robinsonian Dissimilarities", *Journal of Classification* 14, 311–325.

V. Chepoi, B. Fichet and M. Seston (2009), "Seriation in the Presence of Errors: NP-Hardness of l_∞-Fitting Robinson Structures to Dissimilarity Matrices", *Journal of Classification* 26, 279–296.

V. Chepoi and M. Seston (2011), "Seriation in the Presence of Errors: A Factor 16 Approximation Algorithm for l_∞-Fitting Robinson Structures to Distances", *Algorithmica* 59, 521–568.

F. Critchley and B. Fichet (1994), "The Partial Order by Inclusion of the Principal Classes of Dissimilarities on a Finite Set, and Some of Their Basic Properties", in *Classification and Dissimilarity Analysis*, B. Van Cutsem Ed., pp 5–65, Springer Verlag, Lecture Notes in Statistics.

E. Diday (1986), "Orders and Overlapping Clusters by Pyramids", in *Multidimensionnal Data Analysis*, J. de Leeuw, W. Heiser, J. Meulman and F. Critchley Eds., pp 201–234. DSWO.

C. Durand and B. Fichet (1988), "One-to-One Correspondences in Pyramidal Representation: an Unified Approach", in *Classification and Related Methods of Data Analysis*, H.H. Bock Ed., pp 85–90. North-Holland.

D.R. Fulkerson and O.A. Gross (1965), "Incidence Matrices and Interval Graphs", *Pacific Journal of Mathematics* 15, 835–855.

P.C. Gilmore and A.J. Hoffman (1964), "A Characterization of Comparability Graphs and of Interval Graphs", *Canadian Journal of Mathematics* 16, 539–548.

M.C. Golumbic (1980), *Algorithmic Graph Theory and Perfect Graphs*, Academic Press.

M. Habib, R. McConnell, C. Paul and L. Viennot (2000), "Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing", *Theoretical Computer Science* 234, 59–84.

D. Halperin (1994), "Musical Chronology by Seriation", *Computer and the Humanities* 28, 13–18.

V. Heun (2008), "Analysis of a Modification of Gusfield's Recursive Algorithm for Reconstructing Ultrametric Trees", *Information Processing Letters* 108, 222–225.

L. Hubert (1974), "Some Applications of Graph Theory and Related Nonmetric Techniques to Problems of Approximate Seriation: The Case of Symmetric Proximity Measures", *British Journal of Mathematical and Statistical Psychology* 27, 133–153.

D.G. Kendall (1969), "Incidence Matrices, Interval Graphs and Seriation in Archeology", *Pacific Journal of Mathematics* 28:3, 565–570.

A. Lubiw (1985), "Doubly Lexical Orderings of Matrices", *Proceedings of the 17th Annual ACM Symposium on the Theory of Computing*, 396–404.

G.S. Lueker (1975), *Interval Graph Algorithms*, Ph.D. Thesis, Princeton University.

I. Miklós, I. Somodi and J. Podani (2005), "Rearrangement of Ecological Data Matrices via Markov Chain Monte Carlo Simulation", *Ecology* 86:12, 39–49.

B. Mirkin and S. Rodin (1984), *Graphs and Genes*, Springer-Verlag.

R. Paige and R.E. Tarjan (1987), "Three Partition Refinement Algorithms", *SIAM Journal on Computing* 16, 973–989.

W.M.F. Petrie (1899), "Sequences in Prehistoric Remains", *Journal of the Anthropological Institute of Great Britain and Ireland* 29, 295–301.

W.S. Robinson (1951), "A Method for Chronologically Ordering Archeological Deposits", *American Antiquity* 16, 293–301.

M. Seston (2008a), "A Simple Algorithm to Recognize Robinsonian Dissimilarities", *COMPSTAT'2008*, Porto.

M. Seston (2008b), *Dissimilarités de Robinson : Algorithmes de Reconnaissance et d'Approximation*, Ph.D. Thesis, Université de la Méditerranée.

J.P. Spinrad (1993), "Doubly Lexical Ordering of Dense 0-1 Matrices", *Information Processing Letters* 45, 229–235.

A. Strehl and J. Ghosh (2003), "Relationship-Based Clustering and Visualization for High-Dimensional DataMining", *INFORMS Journal on Computing* 15, 208–230.