Two simple but efficient algorithms to recognize Robinson dissimilarities

M. Carmona, V. Chepoi, G. Naves, Pascal Préa

Two simple but efficient algorithms to recognize Robinson dissimilarities [1]

M. Carmona[a,b], V. Chepoi[a], G. Naves[a], and P. Préa[a,b]

[a]LIS, Aix-Marseille Université, CNRS and Université de Toulon, Marseille, France

[b]École Centrale Marseille, Marseille, France

{mikhael.carmona,victor.chepoi,guyslain.naves,pascal.prea}@lis-lab.fr

Abstract. A dissimilarity d on a set S of size n is said to be *Robinson* if its matrix can be symmetrically permuted so that its elements do not decrease when moving away from the main diagonal along any row or column. Equivalently, S admits a total order $<$ such that $i < j < k$ implies that $d(i,j) \leq d(i,k)$ and $d(j,k) \leq d(i,k)$. Intuitively, d is Robinson if S can be represented by points on a line. Recognizing Robinson dissimilarities has numerous applications in seriation and classification. Robinson dissimilarities also play an important role in the recognition of tractable cases for TSP. In this paper, we present two simple algorithms (inspired by Quicksort) to recognize Robinson dissimilarities. One of these algorithms runs in $O(n^2 \log n)$, the other one runs in $O(n^3)$ in worst case and in $O(n^2)$ on average.

Key Words: Robinson dissimilarity; Classification, Seriation; PQ-Tree, Partition Refinement.

1. Introduction

A major issue in classification and data analysis is to visualize simple geometrical and relational structures between objects based on their pairwise dissimilarities. Many applied algorithmic problems ranging from archaeological dating through DNA sequencing and numerical ecology to sparse matrix reordering and overlapping clustering involve ordering a set of objects so that closely coupled elements are placed near each other. For example, the classical *seriation problem*, introduced by Robinson (1951) as a tool to seriate archaeological deposits, asks to find a simultaneous ordering (or permutation) of the rows and the columns of the dissimilarity matrix with the objective that small values should be concentrated around the main diagonal as closely as possible, whereas large values should fall as far from it as possible. This goal is best achieved by considering the so-called Robinson property: a dissimilarity matrix A is said to have the *Robinson property* if its values increase monotonically in the rows and the columns when moving away from the main diagonal in both directions. In case of $(0,1)$-matrices, the Robinson property is best known as the *Consecutive One Property*. A dissimilarity matrix A with Robinson property is called a *Robinson matrix* (or a R-matrix, see Atkins, Boman and Hendrickson (1998)). A *Robinsonian matrix* (called also a *pre-R-matrix* by Atkins and al. 1998) is a dissimilarity matrix A which can be transformed by permuting its rows and columns to a dissimilarity matrix having the Robinson property. The permutation which leads to a matrix with the Robinson property is called a *compatible order*. Instead of dissimilarities, many papers on seriation consider similarities; in this case, "increase monotonically" is replaced by "decrease monotonically".

[1]This research was supported in part by ANR project DISTANCIA (ANR-17 CE40-0015) and has received funding from Excellence Initiative of Aix-Marseille - A*MIDEX (Archimedes Institute AMX-19-IET-009), a French "Investissements d'Avenir" Programme.

1.1. **Related work.** Due to the importance of Robinson dissimilarities in seriation and classification, the algorithmic problem of recognizing Robinsonian dissimilarities on n points attracted the interest of many authors and several polynomial time algorithms for solving this problems have been proposed. The existing recognition algorithms can be classified into *combinatorial* and *spectral*. All combinatorial algorithms are based on the correspondence between Robinson dissimilarities and interval hypergraphs/unit interval graphs. The main difficulty arising in recognition algorithms is the existence of several compatible orders for the whole matrix or for some its submatrices (if the whole matrix is Robinson).

Historically, the first recognition algorithm was given in 1984 by Mirkin and Rodin and consists in testing if the hypergraph of balls of the dissimilarity is an interval hypergraph; it runs in $O(n^4)$ time and uses $O(n^3)$ space. Chepoi and Fichet (1997) gave a simple divide-and-conquer algorithm running in $O(n^3)$ time and using $O(n^2)$ space algorithms to recognize Robinson dissimilarities. The algorithm divides the set of points into subsets and consequently refine the obtained subsets into blocks to which the recursion is applied. Seston (2008) presented another $O(n^3)$ time and $O(n^2)$ space algorithm, by using threshold graphs defined by the input dissimilarity. Seston (2008) consequently improved the complexity of the second algorithm to $O(n^2 \log n)$. Finally, in 2014 Préa and Fortin (2014) presented an algorithm running in optimal $O(n^2)$ time. The optimal complexity of the algorithm of Préa and Fortin (2014) is due to the use of the PQ-trees of Booth and Lueker (1976) as a data structure for encoding all compatible orders. Even if optimal, the algorithm of Préa and Fortin (2014) is far from being simple and efficient in practice.

Subsequently, two new recognition algorithms were proposed by Laurent and Seminaroti (2017): they presented an algorithm of complexity $O(\alpha \cdot n)$ based on classical LexBFS traversal and divide-and-conquer paradigm (where α is the depth of the recursion tree, which is at most the number of distinct nonzero elements of the input matrix), and an $O(n^2 \log n)$ algorithm, which extends LexBFS to weighted matrices and is used as a multisweep traversal. Finally, Laurent, Seminaroti and Tanigawa (2017) presented a structural characterization of Robinson matrices in terms of forbidden substructures, extending the notion of asteroidal triples in graphs to weighted graphs.

The spectral approach to the recognition of Robinson (dis)similarities was originally introduced by Atkins et al. (1998) and was subsequently used in numerous papers (see, for example, Fogel, d'Aspremont and Vojvonic (2016) and the references therein). The method is based on the computation of the second smallest eigenvalue of the Laplacian of a similarity matrix A and of its eigenvector, which are called the *Fiedler value* and the *Fiedler vector* of the matrix A. This leads to an algorithm of complexity $O(nT(n) + n^2 \log n)$ to recognize if a similarity matrix is Robinson, where $T(n)$ is the complexity of computing the Fiedler vector of a matrix. The Fiedler vector is computed by the Lanczos algorithm, which is an iterative numerical algorithm that at each iteration performs a multiplication of the input matrix A by a vector. The authors mention that the algorithm converges in fewer than n iterations, often only $O(\sqrt{n})$. Their algorithm has complexity $O(n^4)$ in the worst case and $O(n^{3.5})$ on average.

Real data contains errors, therefore the dissimilarity between the objects can be measured only approximately and the resulting dissimilarity matrix fails to satisfy the Robinson property. In this case, we are led to the problem of approximating a dissimilarity by a Robinson dissimilarity. As an error measure one can use the usual ℓ_p-distance between two matrices of equal size. However this ℓ_p-fitting problem has been shown to be NP-hard for

$p = 1$ (see Barthélemy and Brucker (2001)) and for $p = \infty$ (see Chepoi, Fichet and Seston (2009)); no efficient algorithm is known for other values of p. Various heuristics for this optimization problem have been considered in (see Hubert (1974) and Hubert, Arabie and Meulman (2006) and papers cited therein). The approximability of this fitting problem for any $1 \leq p < \infty$ is open. Chepoi and Seston (2011) presented a polynomial factor 16 approximation for the ℓ_∞-fitting problem. Given the input dissimilarity matrix A, the algorithm uses the fact that the ℓ_∞-fitting problem is polynomial when the total order $\prec$ (which is derived from the existence of super-dominants) and the fact that the optimal fitting error belongs to a well-defined list of size $O(n^4)$. Running a binary search on this list, for each potential error value $\epsilon > 0$, the algorithm either detects that there is no Robinson matrix approximating A with error at most ϵ or returns a Robinson approximation with factor at most 16ϵ.

Similarly to the classical correspondence between ultrametrics and hierarchies, there is a one-to-one correspondence between Robinson dissimilarities and pseudohierarchies due to Diday (1986) and Durand and Fichet (1988). Pseudohierarchies are now classical examples of classification with overlapping classes. As pseudohierarchies are a generalization of hierarchies, Robinson dissimilarities are a generalization of ultrametrics.

Robinson dissimilarities are also linked with the Traveling Salesman Problem (TSP). First, TSP can be polynomially solved on Robinson dissimilarity matrices by returning any compatible order. Kalmanson (1975) and Demidenko (1976) dissimilarity matrices are two other types of related dissimilarities on which TSP can be polynomially solved (see Deineko et al. (2014) for a study of some of these cases). They are defined quite similarly to Robinson dissimilarities: a dissimilarity on a set S is *Demidenko* (respectively, *Kalmanson*) if there exists a linear (respectively, *circular*) order $\{x_1, \ldots, x_n\}$ such that $i < j < k < \ell$ implies $d(x_i, x_j) + d(x_k, x_\ell) \leq d(x_i, x_k) + d(x_j, x_\ell)$ (respectively, $d(x_i, x_k) + d(x_j, x_\ell) \geq \max\{d(x_i, x_j) + d(x_k, x_\ell), d(x_i, x_\ell) + d(x_j, x_k)\}$). Notice that the currently best algorithm for recognizing Demidenko matrices (and finding a permutation of the points which lead to the Demidenko condition) is due to Çela et al. (2023) and is based on the recognition of Robinson dissimilarities as a subroutine.

1.2. **Our results.** The recognition of Robinson dissimilarities can be viewed as a 2-dimensional version of the classical sorting problem. Therefore, it is natural to investigate in which way the sorting algorithms can be generalized to this 2-dimensional setting. The goal of this paper is to propose two new algorithms to recognize Robinson dissimilarities based on the idea of QuickSort of partitioning the elements with respect to randomly chosen pivots. In our setting, the pivots are randomly chosen pairs of points $\{x, y\}$ and the partition consists in determining, when, say, $x < y$, which points must be located to the left of x, between x and y, and to the right of y. At the difference of QuickSort, this partition step is much less evident because of ambiguity caused by equalities between distances from x to y and from x or y to other points. We deal with this difficulty in two different ways:

- Do a precise study of the ambiguities and of the use of the PQ-tree structure (see Section 2.2). This yields to an algorithm which, similarly to QuickSort, is simple and optimal ($O(n^2)$ time) on average but less efficient ($O(n^3)$ time) in the worst case.
- Use the partition refinement (see Section 2.3). It appears that partition refinement was sufficient to recognize Robinson dissimilarities, and so we designed a simple algorithm without pivots and partitioning. This algorithm runs in $O(n^2 \log n)$ time.

1.3. **Paper's organization.** The paper is organized as follows. In Section 2 we recall the definitions related to Robinson dissimilarities and we recall two classical tools (PQ-trees and refinement of partitions) that will be used by our algorithms. In Section 3, we will show how to partition the input S according to a pair of points. This is the main tool of the first of our algorithms. In Section 4, we describe our first algorithm, which is iterative and that we called IRRI for ITERATIVE-ROBINSON-RECOGNITION-WITH-INTERVALS. In Section 5, we give the second algorithm, which is recursive and that we called RRRR for RECURSIVE-ROBINSON-RECOGNITION-BY-REFINEMENT. In Appendix A, we show experimentally that Algorithm IRRI has complexity $O(n^2)$ on average.

2. Preliminaries

2.1. **Robinsonian dissimilarities.** Let $S = \{p_1, \ldots, p_n\}$ be a set of n elements, called *points*. A partial order $\prec$ on S is called *linear* (or *total*) if any two elements of S are comparable. A *dissimilarity* on S is a symmetric function d from S^2 to the nonnegative real numbers and vanishing on the main diagonal, *i.e.* $d(x, y) = d(y, x) \geq 0$ and $d(x, y) = 0$ if $x = y$. Then $d(x, y)$ is called the *distance* between x and y and (S, d) is called a *dissimilarity space*. A dissimilarity d and a linear order $\prec$ on S are called *compatible* if $x \prec z \prec y$ implies that $d(x, y) \geq \max\{d(x, z), d(z, y)\}$. If d and $\prec$ are compatible, then d is also compatible with the linear order $\prec^{\text{op}}$ opposite to $\prec$. A dissimilarity d on S is said to be *Robinson* if it admits a compatible order (Robinson 1951). Equivalently, d is Robinson if its distance matrix $D = (d(p_i, p_j))$ can be symmetrically permuted so that its elements do not decrease when moving away from the main diagonal along any row or column. Such a dissimilarity matrix D is called *Robinson* (Crichtley and Fichet 1994, Diday 1986, Durand and Fichet 1988 and Hubert 1974) and we will call (S, d) a *Robinson space*.

The *ball* of *radius* $r \geq 0$ and *center* $x \in S$ is the set $B_r(x) := \{y \in S : d(x, y) \leq r\}$. From the definition of a Robinson dissimilarity immediately follows that d is Robinson if and only if there exists a linear order $\prec$ on S such that all balls $B_r(x)$ of (S, d) are intervals of $\prec$. Moreover, this property holds for all compatible orders. Finally notice that if $S' \subset S$, then the restriction $d|_{S'}$ of d to S' is Robinsonian and the restriction of any compatible order $\prec$ of d to S' is a compatible order of $d|_{S'}$.

Basic examples of Robinson dissimilarities are the ultrametrics. Recall, that d is an *ultrametric* if $d(x, y) \leq \max\{d(x, z), d(y, z)\}$ for all $x, y, z \in S$. Another basic example of a Robinson dissimilarity is provided by the standard *line-distance* between n points $p_1 < \ldots < p_n$ of $\mathbb{R}$. Notice that any line-distance has exactly two compatible orders: the order $p_1 < \ldots < p_n$ defined by the coordinates of the points and its opposite. We say that a Robinson dissimilarity d is *straight* if d has exactly two compatible orders. All line-distances are straight but the converse is not true.

A set $B \subset S$ is called a *block* if B is an interval in any compatible order, *ie*, if $x, y \in B$ and $x \prec z \prec y$ in some compatible order $\prec$ implies that $z \in B$. Two disjoint blocks are *consecutive* if their union is also a block. More generally, the blocks in a sequence $B_1, \ldots, B_k$ are *consecutive* if B_i and B_{i+1} are consecutive blocks for all $1 \leq i < k$.

Let U and V be two disjoints subsets of S. We say that U is *independent* from V if for all $x, y \in U$, $z \in V$, we have $d(x, z) = d(y, z)$, *i.e.* if the points of U cannot be distinguished from V. If U is independent from $S \setminus U$, we say that U is *independent*.

Notice that an independent set may not be a block (if d is the constant dissimilarity on a set S, every subset of S is independent, but the only blocks of (S, d) are S and the singletons) and that a block is not necessary independent (if $S = [n]$ and $d(i, j) = |i - j|$, then every interval of S is a block but the only independent sets are S and the singletons).

2.2. **PQ-trees and the Consecutive One's Property.** A PQ-tree is a tree-based data structure introduced by Booth and Lueker in 1976 to efficiently encode a family of permutations on a set S in which various subsets of S occur consecutively. A *PQ-tree* over a set S is a rooted, ordered tree T whose leaves are the elements of S and whose internal nodes are distinguished as P-nodes or Q-nodes. The children of a *P-node* can be arbitrarily permuted. The children of a *Q-node* are ordered and the only permutation we can apply to this ordered list is to reverse it. We use the convention that P-nodes are represented by circles or ellipses and Q-nodes are represented by rectangles. For a P-node or Q-node α of T, we denote by $S(\alpha)$ the set of all leaves in the subtree of T rooted at α. Two PQ-trees are said to be *equivalent* if one can be transformed into the other by applying a sequence of the following two equivalence transformations.

(1) Arbitrarily permute the children of a P-node.
(2) Reverse the children of a Q-node.

Example 1. The PQ-tree of Figure 1 has one Q-node (the root) and one P-node α with $S(\alpha) = \{1, 2, 3\}$ corresponding to all permutations of the elements $1, 2, 3$. Consequently, the equivalence class represented by this PQ-tree corresponds precisely to the set of 12 permutations of the forms $(\pi, 4, 5, 6, 7)$ and $(7, 6, 5, 4, \pi)$, where π is any permutation on $\{1, 2, 3\}$.

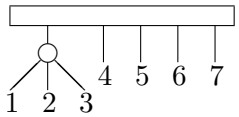

FIGURE 1. A PQ-tree.

PQ-trees are used in DNA sequencing (to create a contig map from DNA fragments), testing a matrix for the consecutive ones property, recognizing interval graphs, and testing planarity of a graph (for last three applications see the original paper by Booth and Lueker (1976). Préa and Fortin (2014) used PQ-trees to encode the compatible orderings of a Robinson dissimilarity space (S, d), namely, that the set of all orders (permutations) compatible with d correspond to the equivalence class represented by a PQ-tree on S. We recall this correspondence.

A $(0, 1)$-matrix A has the *Consecutive Ones Property* (*C1P*) if its columns can be permuted in such a way that in all rows the 1s appear consecutively. Such an order is called *compatible*. If A is a C1P-matrix, then the sets of all its compatible permutations can be represented by a PQ-tree (Booth and Lueker 1976). Let $\mathbf{B}$ denote the set of all distinct balls of a dissimilarity space (S, d), *i.e.* $\mathbf{B} := \{B_r(x) : r \in \mathrm{Im}(d), x \in S\}$, where $\mathrm{Im}(d) := \{\delta \in \mathbb{R} : \exists x, y \in S : d(x, y) = \delta\}$. Let Π be the $\{0, 1\}$-matrix whose columns are indexed by the points of S and rows by the balls of $\mathbf{B}$: for $x \in S$ and $B \in \mathbf{B}$ we define $\Pi(B, x) := 1$ if $x \in B$ and $\Pi(B, x) := 0$ otherwise. The following simple result establishes a link between Robinson dissimilarities and C1P-matrices:

Proposition 1 (Mirkin and Rodin (1984))**.** *A dissimilarity d on S is Robinson if and only if the matrix Π satisfies the C1P. There exists a bijection between the orders compatible with d and the orders compatible with Π.*

Since the sets of all compatible permutations of a C1P-matrix can be represented by a PQ-tree, from Proposition 1 we obtain:

Corollary 1. *The set of all orders compatible with a Robinson dissimilarity space (S, d) can be represented by a PQ-tree.*

Booth and Lueker (1976) designed an iterative algorithm which determines if a matrix M has the C1P. If the answer is "yes", the algorithm of Booth and Lueker constructs the corresponding PQ-tree. The algorithm is sketched in Algorithm 1, where:

- Universal-PQ-Tree($[n]$) returns a PQ-tree representing all $n!$ permutations on $[n]$. This PQ-tree has one internal node, which is a P-node.
- Given a PQ-tree T on a set S and $L \subset S$, PQ-Tree-Update(T, L) returns a PQ-tree representing all permutations encoded by T in which the elements of L are consecutive (form an interval). If there is no such permutation or if T is **None**, then it returns **None**. PQ-Tree-Update(T, L) mainly goes through T via a bottom-up traversal and modify (or not) each node accordingly to its type (P-node or Q-node), the type of its children and the repartition of the elements of L among its children.

Algorithm 1: Booth-Lueker

Input: A $(0,1)$-matrix M with n columns and m lines
Output: A PQ-tree T representing all compatible permutations of M if M has the C1P and **None** otherwise

```
begin
    T ← Universal-PQ-Tree([n]) ;
    forall l line of M do
        L ← {i : l[i] = 1} ;
        T ←PQ-Tree-Update(T, L)
    return T
```

Functions Universal-PQ-Tree and PQ-Tree-Update run in $O(n)$, so Algorithm Booth-Lueker runs in $O(nm)$.

2.3. **Refinement procedures.** The general algorithmic paradigm of partition refinement was introduced by Paige and Tarjan (1987). For recognizing Robinson dissimilarities, this paradigm has already been used in Chepoi and Fichet (1997), Mirkin and Rodin (1984), and Préa and Fortin (2014), as well as in some other papers.

Within the field of Robinson dissimilarities, the paradigm of partition refinement can be expressed in the following way: given a Robinson space (S, d) and two disjoint subsets U and V of S, *refining* U (with respect to V) consists in partitioning U into $U_1 \cup U_2 \cup \ldots \cup U_k$ in such a way that there exists a subset V' of V such that:

- for all $0 < i \leq k$ and for all $x, y \in U_i, z \in V$, the equality $d(x, z) = d(y, z)$ holds.
- for all $0 < i < j \leq k$ and for all $x \in U_i, y \in U_j, z \in V$, if $z \in V'$, then $d(x, z) \leq d(y, z)$, and if $z \in V \setminus V'$, then $d(x, z) \geq d(y, z)$.

Notice that this operation is not possible with arbitrary dissimilarity spaces, for example if $U = \{u_1, u_2, u_3\}, V = \{v_1, v_2\}$ and $d(u_1, v_1) = d(u_1, v_2) < d(u_2, v_1) = d(u_3, v_2) < d(u_3, v_1) = d(u_2, v_2)$. However if (S, d) is Robinson and U is an interval of $U \cup V$ when $U \cup V$ is sorted along a compatible order, then it is possible to refine U with respect to V. In this case, V' is the set of the points to the left of U and $V \setminus V'$ is the set of the points to the right of U (or vice-versa) or whose position is unknown; in addition, each U_i is an interval of U when $U \cup V$ is sorted along a compatible order. More precisely, we will consider the procedure Refine(U, V), which:

- Takes as input a block U, divided into a sequence of consecutive blocks $(U_1, \ldots, U_k)$ and a set $V \subset S \setminus U$.
- Returns a sequence of consecutive blocks $\mathcal{B} = (U_{1,1}, \ldots, U_{1,k_1}, U_{2,1}, \ldots U_{2,k_2}, \ldots, U_{k,1}, \ldots, U_{k,k_k})$ and a partition $V^+ \cup V^- \cup V^\circ$ of V such that:
  (1) for all i, j, $U_{i,j} \subset U_i$,
  (2) for all i, j, $x, y \in U_{i,j}$, and $z \in V$, $d(x, z) = d(y, z)$ holds,
  (3) for all i, j, i', j' such that $i < i'$ or $(i = i'$ and $j < j')$, and for all $x \in U_{i,j}, y \in U_{i',j'}$:
    (i) if $z \in V^+$, then $d(x, z) \geq d(y, z)$,
    (ii) if $z \in V^-$, then $d(x, z) \leq d(y, z)$,
  (4) $z \in V^\circ$ if and only if for all $x, y \in U$, $d(x, z) = d(y, z)$ holds,
  (5) for all i, j, i', j' with $(i, j) \neq (i', j')$, there exists $z \in V$ such that for all $x \in U_{i,j}, y \in U_{i',j'}$, $d(x, z) \neq d(y, z)$ holds.

Notice that:

- Refine(U, V) runs in $O(mp \log p)$, where $p = |U|$ and $m = |V|$.
- If V and V' are two subsets of $S \setminus U$, refining U relatively to V and then refining U relatively to V' is equivalent to refining U relatively to $V \cup V'$.

3. Partitioning S with respect to a pivot-pair

Let (S, d) be a Robinson space with an arbitrary input order on S, which is not compatible with d. A *pivot-pair* is any pair $\{x, y\}$ of distinct points of S. By an *xy-order* we will mean any linear order $\prec$ compatible with d such that $x \prec y$. The aim of this section is to determine how the points of $S \setminus \{x, y\}$ can be located relatively to x and y in any xy-order. This defines a partition of S into 13 classes. We show that some of these classes are blocks of (S, d). Moreover, we establish the order between the resulting blocks and between the remaining classes and these blocks.

Given two points $z, t \in S \setminus \{x, y\}$, we say that z is *before* t *relatively* to the pair $\{x, y\}$ (with the notation $z \prec_{xy} t$ or $z \prec t$ if there is no ambiguity) if $z \prec t$ for every xy-order $\prec$. For two disjoint subsets U, V of S we write $U \prec_{xy} V$ if $u \prec_{xy} v$ for any $u \in U$ and any $v \in V$. Since the reversal of any xy-order is a yx-order and that any compatible order is either an xy-order or an yx-order, $z \prec_{xy} t$ if and only if $t \prec_{yx} z$. Thus U is a block of d if and only if U is an interval of any xy-order. We say that point z_2 is *d-between* two points z_1 and z_3 if z_2 is between z_1 and z_3 (*i.e.* $z_1 \prec z_2 \prec z_3$ or $z_3 \prec z_2 \prec z_1$) for any compatible order. As above, z_2 is d-between z_1 and z_3 if and only if z_2 is between z_1 and z_3 for any compatible xy-order. Consequently, without loss of generality we will further consider only xy-orders.

3.1. The partition $S = L \cup M \cup R \cup X \cup Y \cup A^\circ \cup A^=$.

Let x, y be any pivot pair of S. Comparing for each other point z of S the three distances $d(z, x), d(z, y)$, and $d(x, y)$, we

partition the set S as follows:

$$S = L \cup M \cup R \cup X \cup Y \cup A^\circ \cup A^=,$$

where:

$$\begin{aligned}
L &= \{z \in S : d(z,y) > \max(d(z,x), d(x,y)\},\\
M &= \{z \in S : d(x,y) > \max(d(z,x), d(z,y)\},\\
R &= \{z \in S : d(z,x) > \max(d(z,y), d(x,y)\},\\
X &= \{z \in S : d(z,y) = d(x,y) > d(z,x)\},\\
Y &= \{z \in S : d(z,x) = d(x,y) > d(z,y)\},\\
A^\circ &= \{z \in S : d(z,y) = d(z,x) > d(x,y)\},\\
A^= &= \{z \in S : d(z,y) = d(z,x) = d(x,y)\}.
\end{aligned}$$

Furthermore, we partition the set L into $L = L_\ell \cup L_m \cup L_r$, the set M into $M = M_\ell \cup M_m \cup M_r$, and the set R into $R = R_\ell \cup R_m \cup R_r$, where:

$$\begin{aligned}
L_\ell &= \{z \in S : d(z,y) > d(z,x) > d(x,y)\},\\
L_m &= \{z \in S : d(z,y) > d(z,x) = d(x,y)\},\\
L_r &= \{z \in S : d(z,y) > d(x,y) > d(z,x)\},\\
M_\ell &= \{z \in S : d(x,y) > d(z,y) > d(z,x)\},\\
M_m &= \{z \in S : d(x,y) > d(z,y) = d(z,x)\},\\
M_r &= \{z \in S : d(x,y) > d(z,x) > d(z,y)\},\\
R_\ell &= \{z \in S : d(z,x) > d(x,y) > d(z,y)\},\\
R_m &= \{z \in S : d(z,x) > d(z,y) = d(x,y)\},\\
R_r &= \{z \in S : d(z,x) > d(z,y) > d(x,y)\}.
\end{aligned}$$

We now continue with properties of these sets. We call the sets L, M, R, X, and Y *non-ambiguous* and the sets A° and $A^=$ *ambiguous*. In the following three subsections (*i.e.* until Lemma 12), we assume that d is Robinson.

3.2. **Properties of the non-ambiguous sets.** We consider the properties of the non-ambiguous sets L, M, R, X, and Y. Notice first that $x \in X$ and $y \in Y$.

Lemma 1. *X and Y are blocks.*

Proof. We will prove the result for X. Pick any $z \in X$ and let t be any point which is d-between z and x. We assert that $t \in X$. First notice that $z \prec_{xy} y$. Indeed, if for an xy-order $\prec$ we have $x \prec y \prec z$, then $d(x,y) \leq d(x,z)$, contrary to the assumption that $z \in X$. Thus for any xy-order $\prec$ we have $z \prec t \prec x \prec y$ or $x \prec t \prec z \prec y$. In both cases, we have $d(y,t) \in [d(z,y), d(x,y)]$, yielding $d(y,t) = d(x,y)$. Since $d(x,t) \leq d(x,z) < d(x,y)$, we conclude that $t \in X$. □

Lemma 2. *$L \prec_{xy} X \prec_{xy} M \prec_{xy} Y \prec_{xy} R$.*

Proof. We first show that $L \prec_{xy} X$. Pick any $z \in L$. Since $d(x,z) < d(y,z)$, y cannot be d-between x and z and since $d(x,y) < d(z,y)$, z cannot be d-between x and y. Consequently, $z \prec_{xy} y$ and since X is a block, we get $L \prec_{xy} X$. Similarly, we deduce that $Y \prec_{xy} R$.

We now show that $X \prec_{xy} M$. Let $z \in M$. Since $d(y,z) < d(x,y)$, x cannot be d-between z and y, so $x \prec_{xy} z$, yielding $X \prec_{xy} M$. Similarly, we have $M \prec_{xy} Y$. □

Lemma 3. *$L_\ell \prec_{xy} L_m \prec_{xy} L_r$ and $R_\ell \prec_{xy} R_m \prec_{xy} R_r$.*

Proof. Pick any $u \in L_\ell$, $v \in L_m$, and $w \in L_r$. Then we have $d(x,u) > d(x,v) > d(x,w)$. Since $L \prec_{xy} x$, we obtain $z \prec_{xy} v \prec_{xy} w$. The proof of the second statement is similar. □

Lemma 4. *$M_\ell \prec_{xy} M_m \prec_{xy} M_r$.*

Proof. We prove that $M_\ell \prec_{xy} M_m$ (the proof that $M_m \prec_{xy} M_r$ is similar). Let $z \in M_\ell$ and $t \in M_m$ and suppose by way of contradiction that for some xy-order $\prec$ we have $x \prec t \prec z \prec y$. Since t is between x and z, we must have $d(x,t) \le d(x,z)$. Since z is between t and y, we must have $d(y,z) \le d(y,t)$. This is impossible because $d(x,t) = d(y,t)$ and $d(x,z) < d(y,z)$. □

3.3. **Properties of the ambiguous set $A^=$.** We now provide relationships between the ambiguous set $A^=$ and the non-ambiguous sets. Namely, we show that $A^=$ is "mixed" with the sets L_r, X, M, Y, and R_ℓ.

Lemma 5. *$L_m \prec_{xy} A^= \prec_{xy} R_m$.*

Proof. We prove that $L_m \prec_{xy} A^=$ (the proof that $A^= \prec_{xy} R_m$ is similar). Pick any $z \in L_m$ and $t \in A^=$. Suppose by way of contradiction that for some xy-order $\prec$ we have $t \prec z$. Since $z \prec_{xy} x$, we have $z \prec x$, yielding $d(y,z) \le d(y,t)$, which is impossible because $d(y,z) > d(x,y) = d(y,t)$. □

A more precise location of the set $A^=$ can be given when one of the sets L_r or R_ℓ is nonempty:

Lemma 6. *If $L_r \neq \varnothing$, then $X \prec_{xy} A^=$. If $R_\ell \neq \varnothing$, then $A^= \prec_{xy} Y$. Consequently, if $L_r \neq \varnothing$ and $R_\ell \neq \varnothing$, then $X \prec_{xy} A^= \prec_{xy} Y$.*

Proof. Let $L_r \neq \varnothing$ (the proof of the case $R_\ell \neq \varnothing$ is similar). Pick any $z \in L_r$ and $t \in A^=$. Suppose by way of contradiction there exists an xy-order $\prec$ such that $t \prec x$. Then either $z \prec t \prec x \prec y$ or $t \prec z \prec x \prec y$. If $z \prec t \prec x \prec y$, then we would have $d(x,t) \le d(x,z)$, which is impossible because $d(x,z) < d(x,y) = d(x,t)$. If $t \prec z \prec x \prec y$, then we would have $d(y,z) \le d(y,t)$, which is impossible because $d(y,t) = d(x,y) < d(y,z)$. □

Lemma 7. *If $M \neq \varnothing$, then for all xy-orders $\prec$ and any $z \in A^=$, we have $z \prec x$ or $y \prec z$.*

Proof. Suppose by way of contradiction that there exists an xy-order $\prec$ such that $x \prec z \prec y$. Pick any $t \in M$. Then either we have $x \prec z \prec t \prec y$ or $x \prec t \prec z \prec y$. In the first case we will obtain $d(x,t) \ge d(x,z) = d(x,y)$ and in the second case we will obtain $d(t,y) \ge d(z,y) = d(x,y)$, contrary to the assumption $d(x,y) > \max\{d(x,t), d(t,y)\}$. □

By Lemma 8 below, A° does not "interfere" with X, Y and M. Therefore, from Lemma 7 we obtain:

Corollary 2. *If $M \neq \varnothing$, then M and $X \cup M \cup Y$ are blocks.*

From Lemmas 4 and 2, we have:

Corollary 3. *The sets M_ℓ, M_m and M_r are blocks.*

From Lemmas 6—7, we have:

Corollary 4. *At least one of the sets L_r, M, R_ℓ or $A^=$ is empty.*

Lemmas 6—7 and Corollary 4 are resumed in Table 1.

3.4. **Properties of the ambiguous set A°.** We now look at the set A°. First we show that A° may interfere only with the sets L_ℓ and R_r.

Lemma 8. *$L_m \cup L_r \cup X \cup M \cup A^= \cup Y \cup R_\ell \cup R_m$ is a block.*

Proof. Let $\prec$ be an xy-order and z be a point of $L_\ell \cup A^\circ \cup R_r$. Then $d(x,z) > d(x,y)$ and $d(y,z) > d(x,y)$. Let v_ℓ and v_r be the leftmost and the rightmost points of $L_m \cup L_r \cup X \cup M \cup A^= \cup Y \cup R_\ell \cup R_m$ with respect to $\prec$. Then $v_\ell \preceq x \prec y \preceq v_r$. Consequently, $v_\ell \in L_m \cup L_r \cup X \cup A^=$ and $v_r \in A^= \cup Y \cup R_\ell \cup R_m$, from which we conclude that $d(x,v_\ell) \leq d(x,y)$ and $d(y,v_r) \leq d(x,y)$.

Since $d(x,z) > d(x,y)$ and $d(x,v_\ell) \leq d(x,y)$, z cannot be located between v_ℓ and x. Since $d(y,z) > d(x,y)$ and $d(y,v_r) \leq d(x,y)$, z cannot be located between y and v_r. Finally, since $d(x,z) > d(x,y)$, z cannot be located between x and y. □

In order to determine the relations between A° and L_ℓ, R_r, we pick the points of L_ℓ and L_r farthest from y and x, respectively. Namely, let z_ℓ be a point of L_ℓ such that for all $z \in L_\ell$, we have $d(y,z) \leq d(y,z_\ell)$ and let z_r be a point of R_r such that for all $z \in R_r$, we have $d(x,z) \leq d(x,z_r)$. Finally, we define the following subsets of A°:

$$A^\circ_\ell := \{z \in A^\circ : d(y,z) < d(y,z_\ell) \text{ and } d(z_\ell,z) \leq d(z_\ell,x)\},$$

$$A^\circ_r := \{z \in A^\circ : d(x,z) < d(x,z_r) \text{ and } d(z_r,z) \leq d(z_r,y)\},$$

$$A^\circ_\circ := A^\circ \setminus (A^\circ_\ell \cup A^\circ_r).$$

If $L_\ell = \varnothing$, then we set $A^\circ_\ell := \varnothing$, and if $R_r = \varnothing$, then we set $A^\circ_r := \varnothing$.

Lemma 9. *There does not exist $z \in A^\circ$ such that $d(y,z) < d(y,z_\ell)$ and $d(z_\ell,x) < d(z_\ell,z) < d(z_\ell,y)$.*

Proof. Suppose by way of contradiction that such a point z exists, however there exists an xy-order $\prec$. Since $d(y,z) < d(y,z_\ell)$, $z_\ell \prec z$. As $d(z_\ell,x) < d(z_\ell,z) < d(z_\ell,y)$, z must be located between x and y, which is impossible because $d(x,z) > d(x,y)$. □

Similarly, we have:

Lemma 10. *There does not exist $z \in A^\circ$ such that $d(x,z) < d(x,z_r)$ and $d(z_r,y) < d(z_r,z) < d(z_r,x)$.*

Lemma 11. *$L_\ell \cup A^\circ_\ell$ and $R_r \cup A^\circ_r$ are blocks.*

Proof. Let $\prec$ be an xy-order and pick any $z \in A^\circ_\ell$. By Lemma 8, $z \prec L_m$ or $R_m \prec z$. Since $d(z_\ell,z) \leq d(z_\ell,x) < d(z_\ell,y)$, z cannot be at the right of R_m, hence $A^\circ_\ell \prec L_m$ and so $L_\ell \cup A^\circ_\ell \prec L_m$. Similarly, $R_m \prec R_r \cup A^\circ_r$.

Let $z \in L_\ell \cup A^\circ_\ell$ and $t \in A^\circ_\circ$. We suppose that $t \prec x$ (otherwise $L_\ell \cup A^\circ_\ell \prec t$ and we are done), so, if $d(z_\ell,t) > d(z_\ell,x)$, we have $t \prec z_\ell$. We prove now that $t \prec z$.

- If $d(z,y) < d(z_\ell,y)$, then $z_\ell \prec z$. So, if $d(z_\ell,t) > d(z_\ell,x)$, then $t \prec z$. If $d(z_\ell,t) \leq d(z_e ll,x)$, then, as $t \notin A^\circ_\ell$, we have $d(y,t) \geq d(y,z_\ell)$, and so $t \prec z_\ell$. Thus, in this case too, $t \prec z$.
- If $d(z,y) = d(z_\ell,y)$, then $z \in L_\ell$ and $d(z,x) < d(z,y)$. As $t \notin A^\circ_\ell$, we have $d(y,t) \geq d(y,z_\ell)$ or $d(z_\ell,t) > d(z_\ell,x)$. If $d(y,t) \geq d(y,z_\ell)$, then, as $d(t,x) = d(t,y)$, we have $d(z,x) < d(t,x)$ and so $t \prec x$. If $d(z_\ell,t) > d(z_\ell,x)$, then $t \prec z_\ell$ and $d(t,x) = d(t,y) \geq d(z_\ell,y) = d(z,y) > d(z,x)$. So $t \prec z$.

As it is impossible that $d(z,y) > d(z_\ell, y)$, $L_\ell \cup A^\circ_\ell$ is an interval. Similarly, $R_r \cup A^\circ_r$ is an interval. □

There may exist many points which, like z_ℓ or z_r, maximize the distance from y or x. By Lemma 11, the choice of z_ℓ and z_r among these points has no impact on the sets A°_ℓ and A°_r. In addition, from this lemma we obtain:

Corollary 5. *$A^\circ_\ell \cap A^\circ_r = \varnothing$.*

We conclude this section by showing that the set $S \setminus A^\circ_\circ$ defines an block.

Lemma 12. *$S \setminus A^\circ_\circ$ is a block.*

Proof. Let $\prec$ be an xy-order and pick any $z \in A^\circ_\circ$. First suppose that $z \prec x$ and let t be the leftmost point of $S \setminus A^\circ_\circ$ for $\prec$. If $L_\ell \cup A^\circ_\ell \neq \varnothing$, then $t \in L_\ell \cup A^\circ_\ell$ and, by Lemma 11, $z \prec t$. If $L_\ell \cup A^\circ_\ell = \varnothing$, then $d(x,t) \leq d(x,y) < d(x,z)$, and so $z \prec t$. Similarly, if $y \prec z$, then $S \setminus A^\circ_\circ \prec z$. Since z cannot be located between x and y, we conclude that $S \setminus A^\circ_\circ$ is a block. □

3.5. **Summary.** The links between the sets L_r, X, M, Y, R_ℓ and $A^=$ when $A^= \neq \varnothing$ can be resumed by Table 1. In this case, $(L_\ell \cup A^\circ_\ell,\ L_m,\ \mathcal{B},\ R_m,\ R_r \cup A^\circ_r)$ is a sequence of consecutive blocks. Notice that $L_\ell \cup A^\circ_\ell$, L_m, R_m or $R_r \cup A^\circ_r$ may be empty.

If $A^= = \varnothing$, then $(L_\ell \cup A^\circ_\ell,\ L_m,\ L_r,\ X,\ M,\ Y,\ R_\ell,\ R_m,\ R_r \cup A^\circ_r)$ is a sequence of consecutive blocks (which can all be empty except X and Y).

TABLE 1. The blocks induced by L_r, X, M, Y, R_ℓ and $A^=$ when $A^= \neq \varnothing$. The "main sequence" is a sequence $\mathcal{B}$ (which may be of length 1) of consecutive blocks. The "other blocks" are the induced blocks which are not included in the main sequence. When $L_r = R_\ell = \varnothing$ and $M \neq \varnothing$, the three other blocks are consecutive.

L_r	M	R_ℓ	Main sequence $\mathcal{B}$	Other blocks
$\varnothing$	$\varnothing$	$\varnothing$	$X \cup Y \cup A^=$	X ; Y
		$\neq \varnothing$	$X \cup A^=,\ Y,\ R_\ell$	X
	$\neq \varnothing$	$\varnothing$	$X \cup M \cup Y \cup A^=$	$X,\ M,\ Y$
		$\neq \varnothing$	$A^=,\ X,\ M,\ Y,\ R_\ell$	
$\neq \varnothing$	$\varnothing$	$\varnothing$	$L_r,\ X,\ Y \cup A^=$	Y
		$\neq \varnothing$	$L_r,\ X,\ A^=,\ Y,\ R_\ell$	
	$\neq \varnothing$	$\varnothing$	$L_r,\ X,\ M,\ Y,\ A^=$	
		$\neq \varnothing$	d is not Robinson	

Lemmas 9 and 10, Corollaries 4 and 5 can be settled as:

Theorem 1. *Let d be a dissimilarity on S. If one of the following conditions is satisfied*

- *L_r, M, R_ℓ and $A^=$ are all non empty,*
- *$A^\circ_\ell \cap A^\circ_r \neq \varnothing$,*
- *There exist $z \in A^\circ$ such that $d(y,z) < d(y,z_\ell)$ and $d(z_\ell,x) < d(z_\ell,z) < d(z_\ell,y)$,*
- *There exist $z \in A^\circ$ such that $d(x,z) < d(x,z_r)$ and $d(z_r,y) < d(z_r,z) < d(z_r,x)$,*

then d is not Robinson. In this case, we say that we have an internal contradiction.

It is thus simple to construct, from a pivot-pair (x,y), a set of intervals:

(1) Construct the sets $L_\ell, L_m, L_r, X, M, Y, R_\ell, R_m, R_r, A^=, A^\circ_\ell, A^\circ_r, A^\circ_\circ$

(2) From Table 1, transform these sets into sequences of consecutive blocks.
(3) Remove the empty blocks from the sequence.
(4) Take the blocks two by two to form the intervals.

For instance, if after Step (1), we have $A^=, L_r \neq \varnothing$ and $R_\ell = M = R_m = \varnothing$, then, after Step (2), we have two sequences of consecutive blocks: $(L_\ell \cup A^\circ_\ell, L_m, L_r, X, Y \cup A^=, R_m, R_r)$ and (Y). Since $R_m = \varnothing$, we remove it from the first sequence and so, after Step (4), we get the set $\{L_\ell \cup A^\circ_\ell \cup L_m, L_m \cup L_r, L_r \cup X, X \cup Y \cup A^=, Y \cup A^= \cup R_r, Y\}$.

We call this construction (Steps (1)–(4)) Intervals-From-Pivot-Pair(x, y). If, at Step (1), we have an internal contradiction, then Intervals-From-Pivot-Pair(x, y) returns **None**. Clearly, Intervals-From-Pivot-Pair runs in $O(n)$.

4. An iterative algorithm using PQ-trees

In this section, we present Algorithm irri (Iterative-Robinson-Recognition-with-Intervals), which is a direct application of Sections 3 and 2.2. Algorithm irri considers all pairs $\{x, y\}$ of S as pivot-pairs and for each of them computes the partition from Section 3 and its intervals. Then the algorithm updates the current PQ-tree T by subsequently applying PQ-Tree-Update(T, I) for each occurring interval. We first give a "basic" version of irri (Algorithm 2) which runs n $O(n^3)$ in all cases and then an "elaborate" version of irri (Algorithm 3) which runs in $O(n^3)$ in worst case but in $O(n^2)$ on average (this will be experimentally shown in Appendix A).

Algorithm 2: Iterative-Robinson-Recognition-with-Intervals (Basic Version)

Input: A dissimilarity d on a set S
Output: A PQ-tree T which represents all permutations compatible with d if d is Robinson; **None** otherwise

```
begin
    T ← Universal-PQ-Tree(S) ;
    forall {x, y} ⊂ S do
        I_xy ← Intervals-From-Pivot-Pair(x, y);
        if I_xy = None then
            return None
        forall I ∈ I_xy do
            T ← PQ-Tree-Update(T, I)
    return T
```

Proposition 2. *If d is Robinson, then Algorithm 2 returns a PQ-tree that represents all compatible orders of d, otherwise, it returns* **None**. *Algorithm 2 runs in $O(n^3)$ time.*

Proof. Clearly, if d is Robinson, then Algorithm 2 returns a PQ-tree T such that all permutations compatible with d are represented by the PQ-trees equivalent to T. In addition, for every pair $\{x, y\} \subset S$ $(x \neq y)$, if we denote $d(x, y)$ by δ, then $X \cup L_\ell \cup L_m \cup M \cup A^= \cup Y = B_\delta(x)$. So all balls of d, for all centers and all radius, are implicitly considered by Algorithm 2 and the result follows from Proposition 1.

For each pivot-pair $\{x, y\} \subset S$, $\mathcal{I}_{xy}$ contains at most 13 intervals. Since each interval is treated by PQ-Tree-Update in $O(n)$, Algorithm 2 runs in $O(n^3)$. □

At each step of Algorithm 2, we consider several blocks, not only the one defined by the ball $B_\delta(x)$. Therefore, it may happen that the algorithm builds the PQ-tree of d before checking all pivot-pairs $\{x, y\}$. This option is incorporated in Algorithm 3.

Algorithm 3: ITERATIVE-ROBINSON-RECOGNITION-WITH-INTERVALS (IRRI)

```
Input: A dissimilarity d on a n-set S
Output: A permutation σ, compatible with d if d is Robinson; None otherwise
begin
    T ← UNIVERSAL-PQ-TREE(S) ;
1   κ ← n ; k ← 1 ;
    repeat
2       while k < κ do
            Randomly chose a new pivot-pair {x, y} ⊂ S ;
            I_xy ←INTERVALS-FROM-PIVOT-PAIR(x, y);
            if I_xy = None then
                return None
            forall I ∈ I_xy do
                T ← PQ-TREE-UPDATE(T, I) ;
                if T = None then
                    return None
            k ← k + 1 ;
        σ ← a permutation represented by T;
3       if σ is compatible with d then
            return σ
        κ ← 2 · κ
```

Given two points $u, v \in S$, if no pivot-pair chosen by the Algorithm 3 contains u or v, then the distance $d(u, v)$ is not considered by the algorithm, therefore Algorithm 3 cannot determine if d is not Robinson. So, Algorithm 3 must consider at least $O(n)$ pivot-pairs. Thus, at Line 1, we set κ (which is the initial number of tested pivot pairs) to n. Testing if an order is compatible (line 3) with d can be done in $O(n^2)$ time. To avoid making too many such tests, in the loop beginning at Line 2, we consider $2^{i-1} \times n$ pivots at the i-th occurrence of the **repeat** loop. Consequently, we have:

Proposition 3. *Algorithm 3 runs in $O(Kn)$, where K is the total number of considered pivot-pairs.*

Proof. Suppose that Algorithm 3 stops after testing a total number of K pivot-pairs (with a compatible permutation or **None** as answer). We set $P := K/n$. The test at Line 3 is made for $p = 1, 2, 4, 8, \ldots, P/2, P$, so it is made $\log P$ times. As each of these tests runs in $O(n^2)$, the total time taken by the tests at Line 3 is $O(\log P\, n^2)$, which is less than $O(Kn)$.

The complexity of the rest of the algorithm is dominated by the calls to INTERVALS-FROM-PIVOT-PAIR and PQ-TREE-UPDATE. Each of these calls runs in $O(n)$ and there are K such calls. So Algorithm 3 runs in $O(Kn)$. □

Consequently, Algorithm 3 runs in $O(n^3)$ in the worst case. Experimental simulations in Appendix A show that $K = O(n)$ on average, so Algorithm 3 has an average complexity of $O(n^2)$. In some cases, we can have $K = \Theta(n^2)$, as the following Examples 2 and 3 show:

Example 2. Let $S = \{1, \dots, n\}$, $d(1, i) = i$ for $i > 1$ and $d(i, j) = 1$ for $1 < i < j$. For every couple $(x, y) = (i, j)$ with $1 < i, j$, we have $X = \{i\}$, $Y = \{j\}$, $L = \{1\}$ and $A^{=} = S \setminus \{1, i, j\}$. So, to determine a compatible order, the algorithm must consider all pairs of the form $\{1, i\}$ as pivot-pairs. Since pivots are selected at random, we have to select $O(n^2)$ pairs in order to consider all pertinent ones.

Example 3. Consider the PQ-tree T_t with 3^t leaves, all at depth t and such that all the internal nodes are Q-nodes with three children; see T_3 on Figure 2.

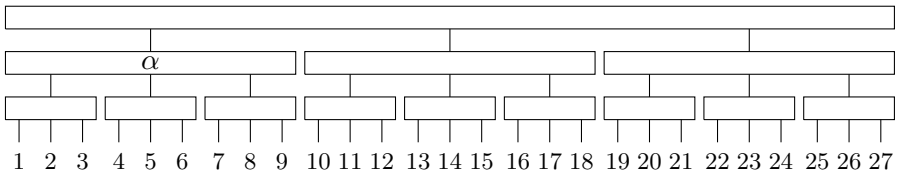

FIGURE 2. The PQ-tree T_3. The subtree with root α is T_2.

On $S = \{1, \dots, 3^t\}$ we define the dissimilarity d_t in the following way: for $i, j \in S$, let α be the lowest common ancestor of i and j and $\beta_1, \beta_2, \beta_3$ be the three children (in this order) of α. We set $d_t(i, j) := 2h$ if i or j is a descendant of β_2 and $d_t(i, j) := 2h + 1$ otherwise, where h is the (graph) distance between α and i. Clearly, d_t is a Robinson dissimilarity. Recall that for a node β of T_t, we define $S(\beta) \subseteq S$ as the set of leaves under β. If we partition S with respect to the pivot-pair $\{i, j\}$, then, up to symmetry, we get:

- $X = S(\beta_1)$, $Y = S_{\beta_3}$, $M_m = S(\beta_2)$ and $A^{\circ}_{\circ} = S \setminus S(\alpha)$ if $i \in S(\beta_1)$ and $j \in S(\beta_3)$.
- $X = S(\beta_1)$, $Y = S(\beta_2)$, $R_m = S(\beta_3)$ and $A^{\circ}_{\circ} = S \setminus S(\alpha)$ if $i \in S(\beta_1)$ and $j \in S(\beta_2)$.

To determine an order compatible with d_t, we have to consider a pivot-pair $\{i, j\} \in S(\alpha)$ for every internal node α, so we must check $O(n^2)$ pivot-pairs.

5. A recursive algorithm by partition refinement

This section is devoted to Algorithm Recursive-Robinson-Recognition-by-Refinement (rrrr). Contrary to irri, it is recursive and does not use PQ-trees, but partition refinement.

5.1. Preliminaries.

Algorithm rrrr is based on the two following Claims 1 and 2 (whose proof is elementary) and on the Refine procedure defined in Section 2.3:

Claim 1. *Let (S, d) be a Robinson space. If S is partitioned into a sequence of consecutive blocks $(B_{1,1}, \dots, B_{1,k_1}, B_{2,1}, \dots, B_{2,k_2}, \dots, B_{k,1}, \dots, B_{k,k_k})$ such that:*

- *for all $1 \leq i \neq i' \leq k, 1 \leq j \leq k_i, 1 \leq j' \leq k_{i'}$, $x, y \in B_{i,j}, z \in B_{i',j'}$, we have $d(x, z) = d(y, z)$,*
- *for all $1 \leq i \leq k, 1 \leq j_1 < j_2 \leq k_i$, there exist $i' \neq i, 1 \leq j' \leq k_{i'}$ such that for all $x \in B_{i,j_1}, y \in B_{i,j_2}, z \in B_{i',j'}$, we have $d(x, z) \neq d(y, z)$,*
- *for all $1 \leq i < i' \leq k, 1 \leq j_1 < j_2 \leq k_i, 1 \leq j' \leq k_{i'}, x \in B_{i,j_1}, y \in B_{i,j_2}, z \in B_{i',j'}$, we have $d(x, z) \geq d(y, z)$,*
- *for all $1 \leq i' < i \leq k, 1 \leq j_1 < j_2 \leq k_i, 1 \leq j' \leq k_{i'}, x \in B_{i,j_1}, y \in B_{i,j_2}, z \in B_{i',j'}$, we have $d(x, z) \leq d(y, z)$.*

Additionally, if for all $1 \leq i \leq k$, π_i is a permutation of $B_i := \bigcup_{j=1}^{k_i} B_{i,j}$ such that

- *π_i is a compatible order on $(B_i, d|_{B_i})$,*
- *for all $1 \leq j_1 < j_2 \leq i_k$, $x \in B_{i,j_1}, y \in B_{i,j_2}$, we have $x \prec_{\pi_i} y$,*

then $\pi_1\pi_2 \dots \pi_k$ is a compatible order of d.

Claim 2. *Let (S, d) be a Robinson space, U be an independent block of S, and $u \in U$. Then, for every compatible order σ of $d|_U$ and $\pi_1 u \pi_2$ of $d|_{S\setminus U\cup\{u\}}$, $\pi_1\sigma\pi_2$ is a compatible order of d. Moreover, every compatible order on (S, d) can be decomposed in this way.*

5.2. **The algorithm.** We now describe Algorithm RRRR. It takes as input a pair (S, U) where (S, d) is a dissimilarity space and U is a subset of S of size $\neq 1$ (U may be empty) and returns, if (S, d) is Robinson, a compatible order of S. Sets S and U are given as sequences (eventually of length 1) of disjoint subsets $(B_1, \dots, B_k)$ and $(U_1, \dots, U_{k'})$. If (S, d) is Robinson, then $(B_1, \dots, B_k)$ and $(U_1, \dots, U_{k'})$ will be sequences of consecutive blocks (so U is a block). The initial call of RRRR will be with $U = \varnothing$ and $k = k' = 1$.

If $|S| \leq 2$ or if d is the constant dissimilarity, then (S, d) is Robinson and RRRR **returns** any order such that $B_1 < \ldots < B_k$. If all blocks B_i are singletons, then RRRR **returns** the total order on these sets. Otherwise:

(1) If $U = \varnothing$, then:
  (a) If S is given by a sequence $(B_1, \dots, B_k)$ with $k > 1$, then take as U any set B_i of size > 1.
  (b) Otherwise, as d is not constant, then there exist $x \in S$ and $\delta > 0$ such that $B_\delta(x) \subsetneq S$. Set $U := B_\delta(x)$. (Section 5.3 shows how to find such a ball $B_\delta(x)$ or to decide that d is constant in $O(n)$ time, after a $O(n^2 \log n)$ preprocessing). Notice that $|B_\delta(x)| > 1$ and that, if (S, d) is Robinson, then $B_\delta(x)$ is a block.
(2) Refine U relatively to $V := S \setminus U$.
(3) Two cases may occur:
  (a) If $V^- = V^+ = \varnothing$, then U is an independent block. By Claim 2, after performing the following operations (i)-(iii), the algorithm returns a compatible order in Step (4) if (S, d) is Robinson.
    (i) Pick $u \in U$.
    (ii) Apply RRRR on $(U, \varnothing)$. We get a permutation σ of U.
    (iii) Apply RRRR on $(V \cup \{u\}, \varnothing)$. We get a permutation $\pi_1 u \pi_2$ of $V \cup \{u\}$.
  (b) If $V^- \cup V^+ \neq \varnothing$, then U is partitioned into a sequence of disjoint subsets $(U_1, \dots, U_p)$ with $p > 1$.
    (i) Randomly choose $u_1 \in U_1$ and $u_p \in U_p$.
      - If $V^- \neq \varnothing$, then take $v^- \in V^-$ such that $d(v^-, u_1)$ is maximal and set $W^- := \{w \in V^\circ : d(u_1, w) \leq d(u_1, v^-) \wedge d(v^-, w) \leq d(v^-, u_1)\}$.
      - If $V^- = \varnothing$, then $W^- := \varnothing$.
      - If $V^+ \neq \varnothing$, then take $v^+ \in V^+$ such that $d(v^+, u_p)$ is maximal and set $W^+ := \{w \in V^\circ : d(u_k, w) \leq d(u_p, v^+) \wedge d(v^+, w) \leq d(v^+, u_p)\}$.
      - If $V^+ = \varnothing$, then $W^+ := \varnothing$.

    By Lemma 13, $\overline{V^-} := V^- \cup W^-$, U, and $\overline{V^+} := V^+ \cup W^+$ are consecutive blocks.
    (ii) Refine $\overline{V^-}$ and $\overline{V^+}$ relatively to U. If these sets are nonempty, then we get $\overline{V^-}$ as a sequence $(B_1^-, \dots, B_{k^-}^-)$ and $\overline{V^+}$ as $(B_1^+, \dots, B_{k^+}^+)$. Note that in each case, U is unchanged.

(iii) Pick a point $u \in U$ and set $U' := \overline{V^-} \cup \{u\} \cup \overline{V^+}$, given as the sequence $(B_1^-, \ldots, B_{k^-}^-, \{u\}, B_1^+, \ldots, B_{k^+}^+)$.
(iv) Run RRRR on $(U = (U_1, \ldots, U_k), \varnothing)$ and get a permutation σ of U.
(v) Run RRRR on $(V^\circ \cup U', U')$ and get a permutation $\pi := \pi_1 u \pi_2$.

(4) **Return** $\pi_1 \sigma \pi_2$.

The pseudo-code of Algorithm RRRR is given by Algorithm 4.

Algorithm 4: RECURSIVE-ROBINSON-RECOGNITION-BY-REFINEMENT (RRRR)

Input: A set S, given by an ordered partition $(B_1, \ldots, B_k)$, a set $U \subset S$ with $|U| \neq 1$, given by an ordered partition $(U_1, \ldots, U_{k'})$, a dissimilarity d on S.
If d is Robinson, then $(B_1, \ldots, B_k)$ and $(U_1, \ldots, U_{k'})$ are sequences of consecutive blocks. We may have $k = 1$, $k' = 1$ or $U = \varnothing$.
Output: A permutation of S which, when (S, d) is Robinson, is compatible with d.
begin
  if $|S| \leq 2$ **or** d *is a constant dissimilarity* **or** $\forall i \in \{1, \ldots, k\}, |B_i| = 1$ **then**
    return *any permutation of* S *such that* $B_1 \prec \ldots \prec B_k$
  if $U = \varnothing$ **then**
    if $k > 1$ **then**
      Let B_i be such that $|B_i| > 1$;
      $U \leftarrow (B_i)$
    else
      Let $x \in S, \delta > 0$ be such that $B_\delta(x) \neq \varnothing$;
      $U \leftarrow (B_\delta(x))$
  REFINE$(U, S \setminus U)$ `// We get` $S \setminus U$ `partitioned into` V^+, V^-, V°
  if $V^+ = V^- = \varnothing$ **then**
    Let $u \in U$;
    $\sigma \leftarrow$ RRRR$(U, \varnothing)$;
    $\pi_1 u \pi_2 \leftarrow$ RRRR$(V^\circ \cup \{u\}, \varnothing)$
  else
    `// In this case,` U `is partitioned into` $(U_1, \ldots, U_p)$ `with` $p > 1$
    Let $u_1 \in U_1, u_p \in U_p$;
    if $V^- \neq \varnothing$ **then**
      Let $v^- \in V^-$ such that $d(v^-, u_1)$ is maximum ;
      $V^- \leftarrow V^- \cup \{w \in V^\circ : d(u_1, w) \leq d(u_1, v^-)$ and $d(v^-, w) \leq d(v^-, u_1)\}$
    if $V^+ \neq \varnothing$ **then**
      Let $v^+ \in V^+$ such that $d(v^+, u_p)$ is maximum ;
      $V^+ \leftarrow V^+ \cup \{w \in V^\circ : d(u_k, w) \leq d(u_k, v^+)$ and $d(v^+, w) \leq d(v^+, u_k)\}$
    REFINE(V^-, U) `// We get` V^- `partitioned into` $(B_1^-, \ldots, B_{k^-}^-)$
    REFINE(V^+, U) `// We get` V^+ `partitioned into` $(B_1^+, \ldots, B_{k^+}^+)$
    Let $u \in U$;
    $U' \leftarrow V^- \cup \{u\} \cup V^+$ `//` $U' = (B_1^-, \ldots, B_{k^-}^-, \{u\}, B_1^+, \ldots, B_{k^+}^+)$
    $\sigma \leftarrow$ RRRR$(U = (U_1, \ldots, U_p), \varnothing)$;
    $\pi_1 u \pi_2 \leftarrow$ RRRR$(V^\circ \cup U', U')$ `//` $V^\circ \cup U'$ `is considered as a unique block`
  return $\pi_1 \sigma \pi_2$

Lemma 13. *With the notations of Step (3-b-i): A point $w \in V^\circ$ is d-between a point of V^- and u_1 if and only if $w \in W^-$; a point $w \in V^\circ$ is d-between a point of V^+ and u_p if and only if $w \in W^+$.*

Proof. We prove the first assertion (the proof of the second one is similar). For a given u_1u_p-order, let v'_ℓ be the leftmost point of V^-. Even if $v'_\ell \neq v_\ell$, we have $d(v'_\ell, u_1) = d(v_\ell, u_1)$. So the "only if" part is the definition of Robinson dissimilarities.

To prove the converse, we only consider u_1u_p-orders. Since $d(w, u_1) \leq d(v^-, u_1)$, we have $d(w, u_p) < d(v^-, u_p)$, so w is on the right of v^-. As $d(v^-, w) \leq d(v^-, u_1)$, we have $d(v^-, w) < d(v^-, u_p)$ and w is on the left of u_p. As U is a block, w is on the left of u_1. □

When we call RRRR at Step (3-a-ii) or (3-b-iv), U has to be partitioned into a sequence of consecutive blocks $(U_1, \ldots, U_k)$ such that, $\forall 1 \leq i \leq k, x, y \in U_i, z \in S \setminus U$, we have $d(x, z) = d(y, z)$, *i.e.* U has to be refined relatively to $S \setminus U$.

If there is a sequence of Step (3-b), and so of calls by Step (3-b-iv), then we get a sequence of sets $U^0, U^1 = U'^0, U^2 = U'^1, \ldots$ By Step (2), each U^i is refined relatively to $S \setminus \bigcup_{j<i} U^j$. But, at Step (3-b-ii), U^i is refined relatively to U^{i-1} but not relatively to $\bigcup_{j<i-1} U^j$. But as $\forall x, y \in \bigcup_{j<i-1} U^j, z \in U^i$, we have $d(x, z) = d(y, z)$ and U^{i-1} contains a point of $\bigcup_{j<i-1} U^j$ (at Step (3-b-iii), we pick $u \in \bigcup_{j<i-1} U^j$), refining U^i relatively to U^{i-1} is equivalent to refining U^i relatively to $\bigcup_{j<i} U^j$.

So, we have:

Proposition 4. *If d is Robinson,* RRRR$(S, \varnothing)$ *returns a compatible order.*

Notice in addition that the construction of W^- and W^+ at Step (3-b-i) is similar to the construction of A°_ℓ and A°_r for Algorithms 2 and 3 (see Section 3.4)

5.3. **Finding a block.** Before running RRRR we compute, for each $x \in S$, the balls $B_\delta(x)$ centered in x and of radius $\delta > 0$ that we give as a sequence $\mathcal{B}(x) := (B_\delta(x))$, sorted by increasing values of δ. This takes (for all $x \in S$) $O(n^2 \log n)$. There are three cases where RRRR can be called with $U = \varnothing$:

- The initial call: finding a ball $B_\delta(x)$ which is different from S or determining that d is a constant dissimilarity takes $O(n)$: we only have to check the first element of each sequence $\mathcal{B}(x)$.
- At Step (3-a-ii) or (3-b-iv). If U is not partitioned into a sequence of consecutive blocks $(U_1, \ldots, U_k)$, then for all $x, y \in U, z \in S \setminus U$, we have $d(x, y) \leq d(x, z) = d(y, z)$. So $d|_U$ is constant if and only if for all $x \in U$, the first element $B_{\delta_1}(x)$ of $\mathcal{B}(x)$ is such that $|B_{\delta_1}(x)| \geq |U|$. Thus checking if d is constant or finding a block can be done in $O(n)$.
- At Step (3-a-iii) with V not partitioned into a sequence of consecutive blocks. For $x \in V$, as $d(x, y)$ has a constant value for $y \in U$, $d(x, y)$ has a constant value for $y \in V \cup \{u\}$ if and only if the first element $B_{\delta_1}(x)$ of $\mathcal{B}(x)$ is such that $|B_{\delta_1}(x)| \geq |S|$ (It is possible that $|B_{\delta_1}(x)| > S$ since the computation of $\mathcal{B}(x)$ has been made as a pre-treatment, on the whole set). We can check that in $O(|V|)$ time.

  We now look at the balls centered at u. If one of these balls contains a point not in U, it contains all of U. So, a ball $B_\delta(u)$ corresponds to a block of $V \cup \{u\}$ if and only if $|U| < |B_\delta(u)| < |U \cup V|$. If such a ball exists, it appears in the first $|U \cup V|$ elements of $\mathcal{B}(u)$. So checking the existence of such a ball takes $O(|U \cup V|)$.

5.4. **Complexity.** In this subsection, we prove the following result.

Proposition 5. *Algorithm* RRRR *runs in* $O(n^2 \log n)$ *time in worst case.*

Proof. We set $n := |S|$, $p := |U|$ and $m := n - p$. Apart from the recursive calls, the steps of RRRR have the following complexity: $O(n)$ for Steps (1), (3-b-i) and (3-b-iii), $O(mp \log p)$ for Step (2), $O(mp \log m)$ for Step (3-b-ii), and $O(1)$ for Step (3-a-i). So, apart from the recursive calls, there exists a constant C such that the number of operations taken by RRRR(S, U) is at most $Cmp \log n$.

We now show by induction on n that there exists a constant C' such that the number of operations taken by RRRR is at most $C'n^2 \log n$. The property trivially holds for small values of n. We suppose, with no loss of generality that $U \neq \varnothing$ (when $U = \varnothing$, at Step (1), the algorithm builds in $O(n)$ a non-empty set U). So we have $p \geq 2$. We set $C' > 2C$,

Suppose now that, for all $n' < n$, RRRR runs on a set S' with $|S'| = n'$ within at most $C'n'^2 \log n'$ operations. Since $C' > 2C$ and $p \geq 2$, we obtain the following bounds for the total number of operations for RRRR(S, U):

$$\begin{aligned}
&\leq C'p^2 \log p + C'(m+1)^2 \log(m+1) + Cmp \log n \\
&\leq (C'p^2 + C'(m+1)^2 + Cmp) \log n \leq C'(p^2 + (m+1)^2 + \frac{mp}{2}) \log n \\
&\leq C'(p+m)^2 \log n = C'n^2 \log n.
\end{aligned}$$

This concludes the complexity analysis of Algorithm RRRR. □

Remark 1. It is possible to make REFINE(U, V) run in $O(\max(|U|^2, |V|^2))$ but, if we apply this version of REFINE to RRRR, we get an algorithm in $O(n^3)$ instead of $O(n^2 \log n)$, for instance with a sequence of recursive calls at Step (3-b-v) with, at each time, $|U| = 2$.

6. Conclusion

In regard to the algorithm of Section 4, one can note that the decomposition of M into $M_\ell \cup M_m \cup M_r$ changes neither the theoretical complexity nor the results of the empirical tests. The two examples at the end of Section 4 lead to the following open questions:

Question 1: In these two examples, only $O(n)$ pivots need to be considered and Algorithm 3 checks $O(n^2)$ pivots to find $O(n)$ pertinent ones. Does there exist dissimilarities which actually need $O(n^2)$ pivots? Or at least more than $O(n)$ ones?

Question 2: If every dissimilarity needs less than $O(n^2)$ pivots (*i.e.* if the answer of Question 1 is "No"), is it possible to derandomize Algorithm 3 in such a way that pertinent pivots are selected first? In this case, Algorithm IRRI would be optimal in $O(n^2)$.

As mentioned is Section 2.3, partition refinement has already been used to recognize Robinson dissimilarities, but the algorithm RRRR is the first one to use only this paradigm. Moreover, although it uses only one tool, it is quite efficient since it runs in $O(n^2 \log n)$.

Data Availability

The code we use for simulations is available from the corresponding author upon request. We do not analyze or generate other datasets.

References

[1] ATKINS, J.E., BOMAN, E.G., and HENDRICKSON, B. (1998), Spectral algorithm for seriation and the consecutive ones problem, *SIAM Journal on Computing 28*, 297–310.

[2] BARTHÉLEMY, J.-P. and BRUCKER, F. (2001), NP-hard approximation problems in overlapping clustering, *Journal of Classification 18*, 159–183.

[3] BOOTH, K.S. and LUEKER, G.S. (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.

[4] ÇELA, E., DEINEKO, V. and WOEGINGER G.J. (2023), Recognising permuted Demidenko matrices, ArXiv:2302.05191v1.

[5] CHEPOI, V. and FICHET, B. (1997), Recognition of Robinsonian dissimilarities, *Journal of Classification 14*, 311–325.

[6] CHEPOI, V., FICHET, B. and SESTON, M (2009), Seriation in the presence of errors: NP-hardness of l_∞-fitting Robinson structures to dissimilarity matrices, *Journal of Classification 26*, 279–296.

[7] CHEPOI, V., and SESTON, M (2011), Seriation in the presence of errors: an approximation algorithm for fitting Robinson structures to dissimilarity matrices, *Algorithmica 59*, 521—568.

[8] CRITCHLEY, F. and FICHET, B. (1994), The partial ordre by inclusion of the principal classe of dissimilarity on a finite set, and some properties of their basic properties, In *Classification and Dissimilarity Analysis*, B. van Cutsen Ed., Lecture Notes In Statistics, 5–65.

[9] DEINEKO, V., KLINZ, B., TISKIN, A. and WOEGINGER, G.J. (2014), Four-point conditions for the TSP: The complete complexity classification, *Discrete optimization 14*, 147–159.

[10] DEMIDENKO, V.M. (1976), A special case of traveling salesman problem, *Izvestiya Akademii Nauk Belarusi. Seriya Fiziko-Matematicheskikh Nauk 5*, 28-32 (in Russian).

[11] DIDAY, E. (1986), Orders and overlapping clusters by pyramids, in *Multidimensionnal Data Analysis*, J. de Leeuw, W. Heiser, J. Meulman and F. Critchley Eds., 201–234, DSWO.

[12] DURAND, C. and FICHET, B. (1988), One-to-one correspondences in pyramidal representation: an unified approach, in *Classification and Related Methods of Data Analysis*, H.H. Bock Ed., 85–90, North-Holland.

[13] FOGEL, F., D'ASPREMONT, A., and VOJNOVIC, M. (2016), Spectral ranking using seriation, *J. Mach. Learn. Res., 17: Paper No. 88.*

[14] HUBERT, L. (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 Statistics and Psychology 27*, 133–153.

[15] HUBERT, L., ARABIE, P., and MEULMAN, J. (2006), *The structural representation of proximity matrices with Matlab*, ASA-SIAM Series on Statistics and Applied Probability.

[16] KALMANSON, K. (1975), Edgeconvex circuits and the travelling salesman problem, *Canadian J. Math.*, 27 (1975), 1000–1010.

[17] LAURENT, M. and SEMINAROTI, M. (2017), A Lex-BFS-based recognition algorithm for Robinsonian matrices, *Discr. Appl. Math. 222*, 151–165.

[18] LAURENT, M. and SEMINAROTI, M. (2017), Similarity-first search: a new algorithm with application to Robinsonian matrix recognition, *SIAM J. Discr. Math., 31*, 1765-1800.

[19] LAURENT, M., SEMINAROTI, M., and TANIGAWA, S.-I. (2017), A structural characterization for certifying Robinsonian matrices, *Electronic J. Combin., 24-2.*

[20] MIRKIN, B. and RODIN, S. (1984), *Graphs and Genes*, Springer-Verlag.

[21] PAIGE, R. and TARJAN, R.E. (1987), Three partition refinement algorithms, *SIAM Journal on Computing, 16*, 973–989.

[22] PRÉA, P. and FORTIN, D. (2014), An optimal algorithm to recognize Robinsonian dissimilarities, *Journal of Classification, 31*, 351–385.

[23] ROBINSON, W.S. (1951), A method for chronologically ordering archeological deposits, *American Antiquity 16*, 293–301.

[24] SESTON, M. (2008), A Simple Algorithm to Recognize Robinsonian Dissimilarities, *COMPSTAT'2008*, Porto.

[25] SESTON, M. (2008), *Dissimilarités de Robinson : Algorithmes de Reconnaissance et d'Approximation*, Ph.D. Thesis, Université de la Méditerranée.

Appendix A. Empirical tests

The aim of this appendix is to show, from a practical point of view, that Algorithm 3 runs in $O(n^2)$ on average or, at least, is much more efficient than $O(n^3)$. More precisely, we made tests on Algorithm 2 and wrote down the number of pivot-pairs tested before obtaining a contradiction or getting the PQ-tree of the dissimilarity. Tests have been made for various generators of "random matrices". For each kind of generator (except for the Toeplitz matrices, see Figure 6), we have generated 1000 matrices of size 25×25, 50×50, 100×100, and 200×200 and noted the minimum number of pivot-pairs tested, the first decile (10% of the matrices need less than this number of pivot-pairs), the median, the last decile, and the maximum number of pivot-pairs tested. Notice that the total number of pivot-pairs is 300, 1225, 4950, and 19900 for 25×25, 50×50, 100×100, and 200×200 matrices respectively.

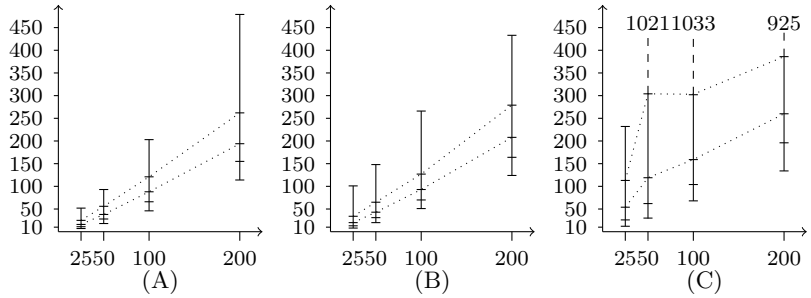

Figure 3. Results of tests on matrices M such that, $\forall i, j$, $M[i,j] = \max(M[i,j-1], M[i+1,j])$ with probability p and $M[i,j] = 1 + \max(M[i,j-1], M[i+1,j])$ with probability $1-p$. On (A), $p = 0.3$, on (B), $p = 0.6$ and on (C), $p = 0.9$. On the abscissa is the size of the matrix and on the ordinate the number of pivot-pairs used before getting the PQ-tree of the dissimilarity (the five marks corresponds with the minimum, the first decile, the median, the last decile and the maximum values of this number for 1000 trials).

We have first tested Algorithm 2 on Robinson matrices generated diagonals-after-diagonals, *i.e.* for each $i < j$, $M[i,j]$ is randomly chosen in $\{\max(M[i,j-1], M[i+1,j]), \max(M[i,j-1], M[i+1,j]) + 1\}$ (see Figure 3). Nearly all dissimilarities that are generated in this way are *flat* (have only one compatible order and its reverse). Although nearly all Robinson dissimilarities are flat (the cone of all Robinson dissimilarities on n points is the subspace of $\mathbb{R}^{\frac{n(n-1)}{2}}$ which is the union of $n!$ convex cones of Robinson dissimilarities with a given compatibility order and the flat Robinson dissimilarities correspond to the inner points of these full cones), we have also generated random Robinson dissimilarities on $[n]$ with several compatible orders. This has been done in the following way:

(1) Generate k random intervals on $[n]$.
(2) Generate the PQ-tree on $[n]$ which represents the set of permutations such that these k intervals (and only them) remain intervals for all these permutations. This is done by k calls of Function PQ-Tree-Update.
(3) Generate a Robinson dissimilarity admitting all these permutations (and only them) as compatible ones.

Such dissimilarities have been generated for $k \in \{\lceil n/10 \rceil, \lceil n/2 \rceil, n, \lceil 3n/2 \rceil, 2n, 3n\}$ and the results of the tests are shown in Figures 4 and 5. These dissimilarities have several compatible orders (see Table 2). We have also tested Algorithm 2 on Toeplitz matrices on $\{0, 1, 2\}$. A $\{0, 1, 2\}$-Toeplitz matrix is defined by $M[i, i] = 0$, $M[i, j] = 1$ if $0 < |i-j| \leq k$ and $M[i, j] = 2$ otherwise (these matrices are Robinson). For our purpose, the interest for these matrices is that they are flat and have very few different values, two characteristics which are, at a first glance, opposite to each other. The result of tests on Toeplitz matrices are shown in Figure 6.

TABLE 2. Average number of compatible permutations of the dissimilarities tested for Figures 4 and 5 for dissimilarities on n points whose compatible orders are represented by PQ-trees built from k intervals of $[n]$.

$n \setminus k$	$\lceil n/10 \rceil$	$\lceil 2n/10 \rceil$	$\lceil n/2 \rceil$	n	$\lceil 3n/2 \rceil$	$2n$
25	$3.65 \cdot 10^{22}$	$8.01 \cdot 10^{17}$	$4.95 \cdot 10^{8}$	554	23	8.85
50	$1.41 \cdot 10^{50}$	$1.48 \cdot 10^{33}$	$2.52 \cdot 10^{12}$	7863	80	11.6
100	$3.11 \cdot 10^{86}$	$2.16 \cdot 10^{51}$	$7.38 \cdot 10^{19}$	$8.74 \cdot 10^{7}$	740	32.2
200	$6.59 \cdot 10^{141}$	$1.58 \cdot 10^{90}$	$1.74 \cdot 10^{37}$	$3.61 \cdot 10^{13}$	89882	161

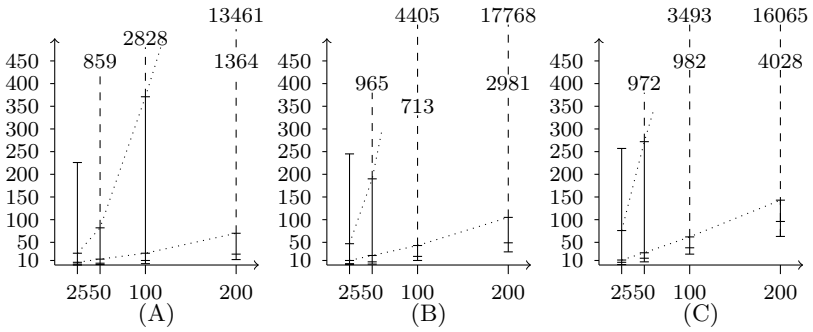

FIGURE 4. Results of tests on matrices generated in the following way: first, we generate a random PQ-tree by running the Booth and Lueker algorithm with k random intervals of $\{1, \ldots, n\}$. Then we generate a Robinson dissimilarity with this PQ-tree. Tests have been made for $n \in \{25, 50, 100, 200\}$ and $k = \lceil n/10 \rceil$ (A), $k = \lceil 2n/10 \rceil$ (B) and $k = \lceil n/2 \rceil$ (C). For each value of n and k, we have made 1000 tests and we have indicated the minimum number of used pivot-pairs, the first decile, the median, the last decile and the maximum value.

On Figures 3–6, one can see that the median value of the number of requested pivot-pairs grows linearly with the number of points. The last decile of the number of used pivot-pairs also grows linearly, except for dissimilarities generated from k intervals with $k < n$. In any case, this last decile is very low when compared with the total number of pivot-pairs.

Finally, we have also tested our algorithm on non-Robinson dissimilarities. On random dissimilarities, the maximum number of pivot-pairs before getting a contradiction is always very low: at most 5 for 200×200 matrices. We have also tested non-Robinson dissimilarities built from Robinson dissimilarities on which a few values have been modified so that the resulting dissimilarity is not Robinson:

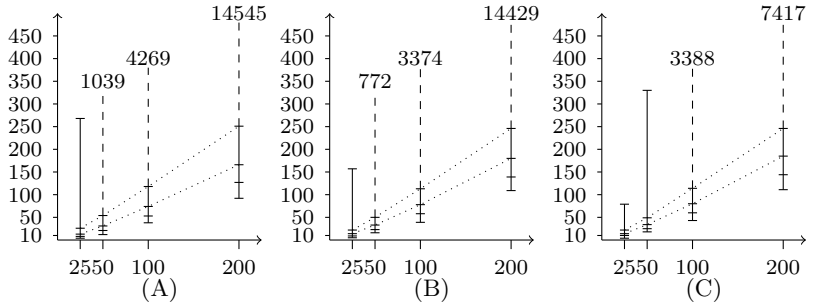

FIGURE 5. Results of tests on matrices generated in the following way: first, we generate a random PQ-tree by running the Booth and Lueker algorithm with k random intervals of $\{1, \ldots, n\}$. Then we generate a Robinson dissimilarity with this PQ-tree. Tests have been made for $n \in \{25, 50, 100, 200\}$ $k = n$ (A), $k = \lceil 3n/2 \rceil$ (B) and $k = 2n$ (C). For each value of n and k, we have made 1000 tests and we have indicated the minimum number of used pivot-pairs, the first decile, the median, the last decile and the maximum value.

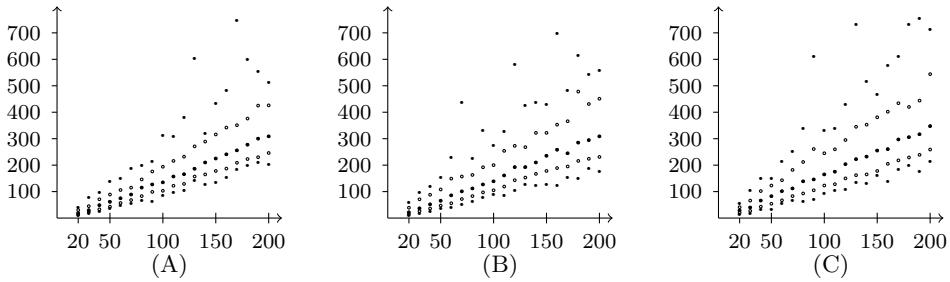

FIGURE 6. Results of tests on $\{0, 1, 2\}$-Toeplitz matrices. A $\{0, 1, 2\}$-Toeplitz matrix is defined by $M[i, i] = 0$, $M[i, j] = 1$ if $0 < |i - j| \leq k$ and $M[i, j] = 2$ otherwise (these matrices are Robinson). Tests have been made for $k = 1$ (A), $k = 5$ (B) and $k = 9$ (C) and matrix sizes in $\{20, 30, \ldots, 190, 200\}$. For each k and matrix size, we have made 100 tests and noted the number of effectively used pivot-pairs (the minimum value, the first decile, the median, the last decile and the maximum value).

- On matrices M such that, $\forall i, j$, $M[i, j] = \max(M[i, j-1], M[i+1, j])$ with probability 0.6 and $M[i, j] = 1 + \max(M[i, j-1], M[i+1, j])$ with probability 0.4 (as for Figure 3-B) with 1, 2, 5 or 10 modified elements. The results are shown in Figure 7.
- On $n \times n$ matrices such that the set of compatible orders is represented by a PQ-tree that can be built from $\lceil 2n/10 \rceil$ intervals (as for Figure 4-B) with 1, 2, 5 or 10 modified elements. The results are shown in Figure 8.
- On $n \times n$ matrices such that the set of compatible orders is represented by a PQ-tree that can be built from $\lceil 3n/2 \rceil$ intervals (as for Figure 5-B) with 1, 2, 5 or 10 modified elements. The results are shown in Figure 9.

We can see that if two or more values are modified, then the median value of the number of pivot-pairs used before getting a contradiction is nearly linear. If there are 5 or 10 modified values, then this is also the case for the last decile. In any case, even with only one modified value, the number of pivot-pairs used before getting a contradiction is much smaller than the total number of pivot-pairs.

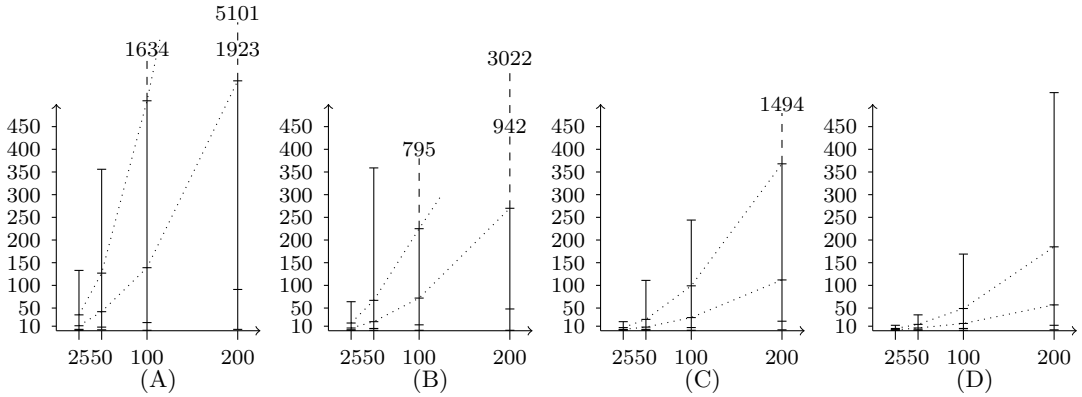

FIGURE 7. Results of tests on matrices generated as in Figure 3-(B) on which one value (A), two values (B), five values (C) and ten values (D) have been changed. All these dissimilarities are non Robinson. For each matrix size and number of modified values, we have made 1000 tests and we have indicated the minimum number of used pivot-pairs, the first decile, the median, the last decile and the maximum value.

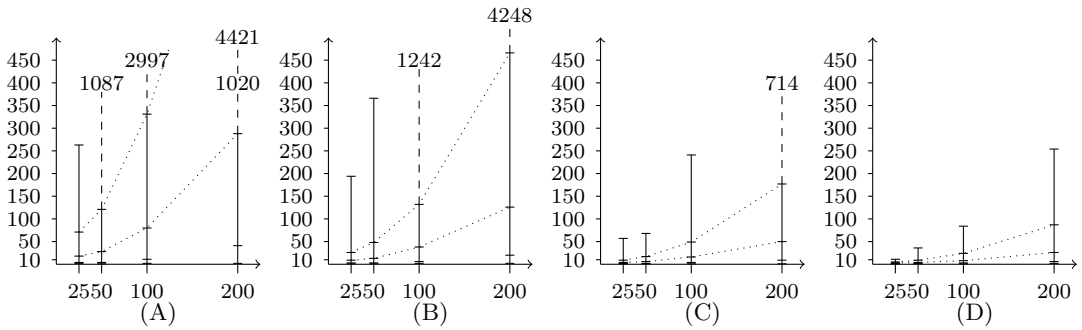

FIGURE 8. Results of tests on matrices generated as in Figure 4-(B) on which one value (A), two values (B), five values (C) and ten values (D) have been changed. All these dissimilarities are non Robinson, except, with one modified value, 246 dissimilarities (on a total of 1000 dissimilarities) on 25 points, 92 on 50 points, 28 on 100 points and 5 on 200 points, and with two modified values: 31 dissimilarities on 25 points and 4 on 50 points. For each matrix size and number of modified values, we have made 1000 tests and we have indicated the minimum number of used pivot-pairs, the first decile, the median, the last decile and the maximum value.

In conclusion, in practice, Algorithm 3 is very efficient and is not distinguishable from the optimal $O(n^2)$ complexity. The running average time in Python 3 on an ordinary desktop computer (Intel Core i5 at 2.7 GHz with 4 Go of memory) is approximatively 26 ms for a 50×50 matrix, 129 ms for a 100×100 matrix and 686 ms for a 200×200 matrix. In Table 3, we show the running time for diverse structures of distance corresponding with Figure 3–9.

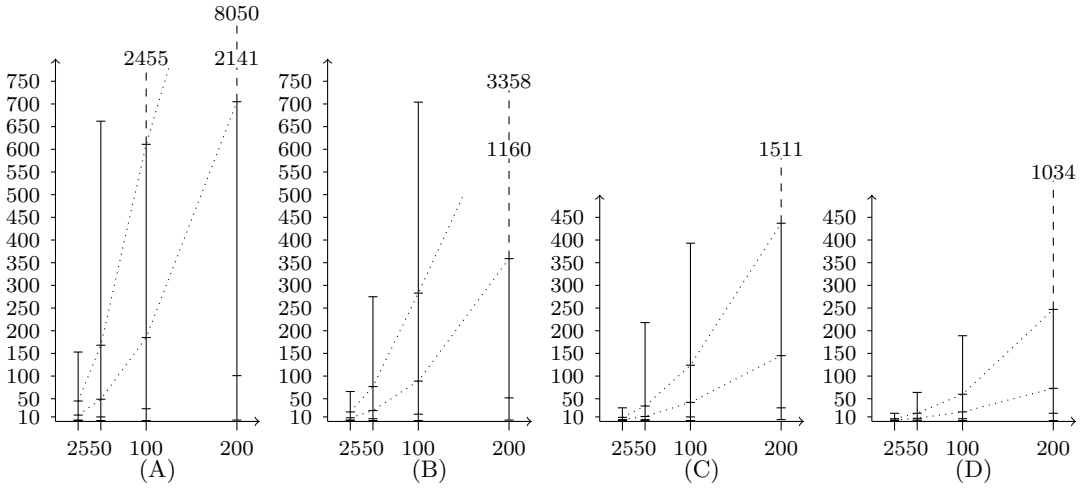

FIGURE 9. Results of tests on matrices generated as in Figure 5-(B) on which one value (A), two values (B), five values (C) and ten values (D) have been changed. All these dissimilarities are non Robinson. For each matrix size and number of modified values, we have made 1000 tests and we have indicated the minimum number of used pivot-pairs, the first decile, the median, the last decile and the maximum value.

TABLE 3. Average running time of Algorithm 3 on different structures of dissimilarities tested in Figures 3–9 for a number n of points in $\{50, 100, 200\}$. These values are the average of 1000 trials.

n	50	100	200
Figure 3 (A)	36 ms	162 ms	729 ms
Figure 3 (B)	39 ms	170 ms	750 ms
Figure 3 (C)	91 ms	275 ms	962 ms
Figure 4 (A)	25 ms	100 ms	427 ms
Figure 4 (B)	28 ms	113 ms	426 ms
Figure 4 (C)	22 ms	81 ms	311 ms
Figure 5 (A)	7 ms	28 ms	109 ms
Figure 5 (B)	7 ms	28 ms	121 ms
Figure 5 (C)	9 ms	32 ms	109 ms
Figure 6 (A)	33 ms	150 ms	637 ms
Figure 6 (B)	25 ms	95 ms	382 ms
Figure 6 (C)	28 ms	99ms	398 ms

n	50	100	200
Figure 7 (A)	38 ms	173 ms	743 ms
Figure 7 (B)	40 ms	184 ms	818 ms
Figure 7 (C)	39 ms	172 ms	759 ms
Figure 7 (D)	38 ms	183 ms	749 ms
Figure 8 (A)	57 ms	391 ms	3.18 s
Figure 8 (B)	29 ms	195 ms	1.45 s
Figure 8 (C)	12 ms	85 ms	616 ms
Figure 8 (D)	7 ms	49 ms	333 ms
Figure 9 (A)	26 ms	201 ms	1.40 s
Figure 9 (B)	12 ms	86 ms	676 ms
Figure 9 (C)	5 ms	31 ms	264 ms
Figure 9 (D)	3 ms	19 ms	121 ms