Mmodules and Robinson dissimilarities

Mikhael Carmona, Victor Chepoi, *Guyslain Naves*, Pascal Préa

Aix-Marseille University and École Centrale Marseille

June 21, 2023, UBC, Vancouver

Dissimilarity

Definition

(X, d) dissimilarity space:

- for all $x \in X$, $d(x, x) = 0$,
- for all $x, y \in X$ distinct, $d(x, y) = d(y, x) > 0$.

$d(x, y)$: the dissimilarity or distance between x and y; how much x and y are dissimilar.

No triangular inequality $(d(x, y) + d(y, z) \geq d(x, z))$.

Equivalent to nonnegative symmetric square matrices (rows and columns indexed by X) with zeros exactly on the diagonal.

Robinson matrix

Definition

M Robinson matrix: M is nonnegative symmetric square with zeros exactly on the diagonal, and each row (or column) is bitonic:

$$m_{i,1} \geq m_{i,2} \geq \ldots \geq m_{i,i} = 0$$

$$0 = m_{i,i} \leq m_{i,i+1} \leq \ldots \leq m_{i,n}$$

$$\begin{pmatrix} 0 & 1 & 4 & 5 & 5 & 6 \\ 1 & 0 & 2 & 4 & 5 & 5 \\ 4 & 2 & 0 & 3 & 3 & 4 \\ 5 & 4 & 3 & 0 & 1 & 2 \\ 5 & 5 & 3 & 1 & 0 & 1 \\ 6 & 5 & 4 & 2 & 1 & 0 \end{pmatrix}$$

Robinson space

A dissimilarity space (X, d) is Robinson if there is a permutation on X such that the associated matrix, with rows and columns ordered along that permutation, is Robinson.

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

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

Decision problem

Question

Given a dissimilarity space (X, d) with X finite, decide whether (X, d) is Robinson.

Definition

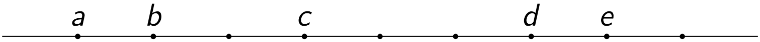
Compatible order: order for the rows and columns which makes the matrix Robinson.

Question

Given a Robinson space, find a (all) compatible order(s).

Example: line distances

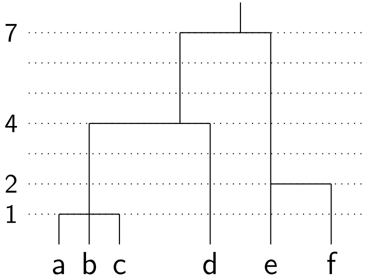
Line distances: take $X \subset \mathbb{R}$, and $d(x, y) = |x - y|$. The (restriction of the) usual order $<$ on $\mathbb{R}$ is a compatible order.



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

Example: ultrametrics

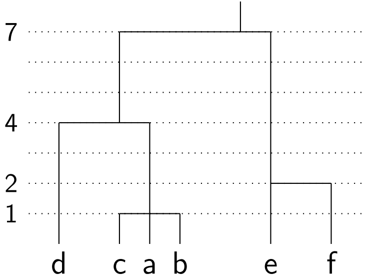
Ultrametrics: take X the leaves of a tree, where all leaves are at the same depth. Let $d(x, y)$ be the height of the least common ancestor of x and y. A left-to-right ordering of the leaves is a compatible order.



	a	b	c	d	e	f
a	0	1	1	4	7	7
b	1	0	1	4	7	7
c	1	1	0	4	7	7
d	4	4	4	0	7	7
e	7	7	7	7	0	2
f	7	7	7	7	2	0

Compatible orders in ultrametrics

- Any reordering of the children of each node of the tree gives a different compatible order.
- Actually, any compatible order is obtainable like this.
- Thus the tree encodes the set of compatible orders.

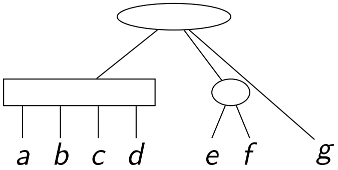


	d	c	a	b	e	f
d	0	4	4	4	7	7
c	4	0	1	1	7	7
b	4	1	0	1	7	7
a	4	1	1	0	7	7
e	7	7	7	7	0	2
f	7	7	7	7	2	0

PQ-trees

Definition

PQ-tree on X: tree with leaves X, and each internal node is either a P-node (Permutation-node, circle), or a Q-node (rectangle).



Represented orders

Equivalence on PQ-trees. PQ-trees are equivalent:

- by reordering *arbitrarily* the order of children of P-nodes (any permutation),

$$P(\alpha_1, \ldots, \alpha_k) \leftrightarrow P(\alpha_{\sigma(1)}, \ldots, \alpha_{\sigma(k)})$$

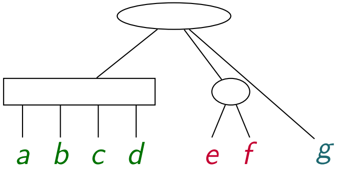
- by reversing the order of children of Q-nodes.

$$Q(\alpha_1, \alpha_2, \ldots, \alpha_k) \leftrightarrow Q(\alpha_k, \ldots, \alpha_2, \alpha_1)$$

Definition

An order on X is represented by a PQ-tree $\mathcal{T}$ if it is the left-to-right order of leaves of a PQ-tree equivalent to $\mathcal{T}$.

Represented orders (example)

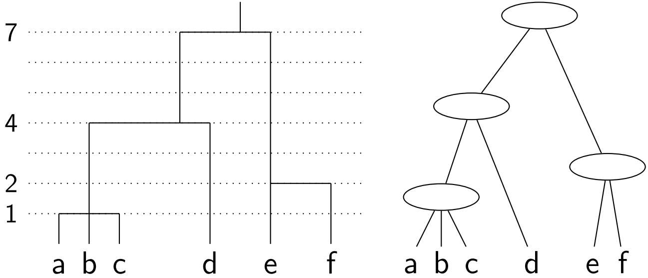


24 represented orders:

- abcdefg,
- dcbaefg,
- efgdcba,
- gabcdfe,
- fedcbag,
- abcdgfe,. . .

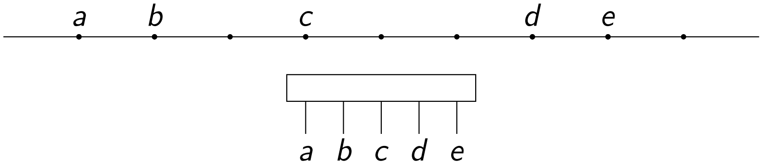
Compatible orders of an ultrametric

For an ultrametric, making each node a P-node gives a representation of the set of compatible orders.



Compatible order of a line distance

For a line distance, having a single Q-node, with children in the same order as on the line gives a representation of the set of compatible orders.



The consecutive-ones property

Let M be a $\{0,1\}$-matrix.

Question

Can we permute the columns of M, such that in each row, the 1s are consecutive?

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

LIS LABORATOIRE D'INFORMATIQUE & SYSTÈMES

The consecutive-ones property

Theorem (Booth and Lueker)

The set of permutations for which the 1s are consecutive is either empty, or representable by a PQ-tree.

Algorithm:

1. start with $\mathcal{T} := P(1, \ldots, n)$,
2. for each row, add to $\mathcal{T}$ the constraint that the 1s of that row must be consecutive (or fail).

Applying Booth and Lueker to Robinson

Definition

The ball with center $x \in X$ and radius $r \in \mathbb{N}$ is

$$B(x, r) := \{y \in X : d(x, y) \leq r\}.$$

	a	b	c	d	e	f
a	0	1	3	4	6	8
b	1	0	1	4	4	7
c	3	1	0	1	2	6
d	4	4	2	0	1	4
e	6	4	2	1	0	2
f	8	7	6	4	2	0

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

Balls with center c

Applying Booth and Lueker to Robinson

Theorem (Mirkin and Rodin)

A dissimilarity space is Robinson if and only if the incidence matrix of balls has the consecutive-ones property.

Furthermore, compatible orders are exactly those orders where the 1s are consecutives.

Theorem

The set of compatible orders of a Robinson space is representable by a PQ-tree.

Also, gives a polynomial-time algorithm.

LIS
LABORATOIRE D'INFORMATIQUE & SYSTÈMES

Deciding whether a dissimilarity is Robinson.

Let $n = |X|$.

- Mirkin and Rodin, 1984: $O(n^4)$,
- Chepoi and Fichet, 1997, divide-and-conquer: $O(n^3)$,
- Atkins, Boman, Hendrickson, 1998, spectral method: $O(nT(n) + n^2 \log n)$,
- Seston, 2008, threshold graphs: $O(n^2 \log n)$,
- Fortin and Préa, 2014, PQ-trees: $O(n^2)$,
- Laurent and Seminaroti, 2017, LexBFS: $O(n^2 + nm \log n)$.

$O(n^2)$ is optimal (size of the input).

LIS LABORATOIRE D'INFORMATIQUE & SYSTÈMES

Goals

- To find a simpler $O(n^2)$ algorithm, efficient in practice, that do not use Booth and Lueker algorithm,
- To study the correspondance between PQ-trees and mmodule trees (and ultrametrics).

This talk:

1. Introduction to Robinson spaces and PQ-trees (done),
2. Mmodules and their relations to PQ-trees,
3. Flat Robinson spaces.

Mmodules

Definition

Mmodule $M \subseteq X$: for each $x, y \in M$ and each $z \notin M$, $d(x,z) = d(y,z)$.

An mmodule is a set of elements indistinguishable from elements outside the set. Example: $\{d, e\}$.

	a	b	c	d	e
a	0	1	3	4	4
b		0	2	4	4
c			0	2	2
d				0	1
e					0

Mmodules

- X is an mmodule, so is any one-element set and $\emptyset$ (trivial mmodules).
- Reminiscent of modules in graph theory. Mmodule $=$ metric-module or matrix-module.
- Maximal modules in graphs can be computed with a partition refinement technique.
- Known as *clans* in symmetric 2-structure (Erhenfeucht and Rozenberg).

LABORATOIRE D'INFORMATIQUE & SYSTÈMES

Some properties of mmodules

Lemma

Let M_1, M_2 be mmodules, then

(i) $M_1 \cap M_2$ *is an mmodule,*

(ii) *if* $M_1 \cap M_2 \neq \emptyset$, *then* $M_1 \cup M_2$ *is an mmodule,*

(iii) *if* $M_1 \cap M_2 = \emptyset$, *then* $d(x, y)$ *is constant for* $x \in M_1$, $y \in M_2$.

Lemma

Let M_1, M_2 be distinct maximal mmodules (maximal by inclusion distinct from X), then

(i) *if* $M_1 \cap M_2 \neq \emptyset$, $M_1 \cup M_2 = X$,

(ii) *if* M *is an mmodule contained in* $M_1 \cup M_2$, *then* $M \subseteq M_1$ *or* $M \subseteq M_2$.

Partitions and copartition

Lemma

The maximal mmodules $\mathcal{M}_{\max}$ are either a partition of X, or their complements are a partition of X (that is, they are a copartition of X).

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

$\mathcal{M}_{\max}$: ab, cd, ef

	a	b	c	d	e	f
a	0	2	4	4	4	4
b		0	4	4	4	4
c			0	2	4	4
d				0	4	4
e					0	3
f						0

$\mathcal{M}_{\max}$: abcd, cdef, abef

The mmodule tree

Lemma

There is a unique tree, the mmodule tree, with leaves X, and inner nodes labelled $\cup$ and $\cap$, such that

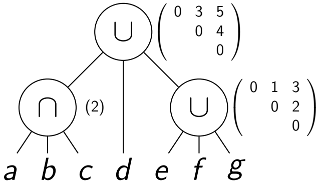
(i) *if a node α is a $\cup$-node, its arity is at least 3, and for any child β of α, $X(\beta)$ is an mmodule (partition case);*

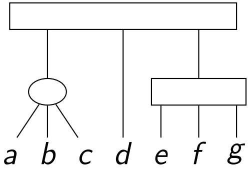
(ii) *if a node α is a $\cap$-node, its arity is at least 2, and for any children $\beta_1, \ldots, \beta_k$ of α, $X(\beta_1), \ldots, X(\beta_k)$ is an mmodule (copartition case);*

(iii) *any proper mmodule appears exactly once as in (i) or (ii).*

$X(\beta)$: set of leaves with ancestor β.
This holds for any dissimililarity space (not just Robinson).
The order of childrens does not matter.

Example of mmodule tree





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

Mmodule tree and PQ-tree

Erhenfeucht, Gabow, MacConnell, Sullivan 1994:
$O(|X|^2)$-time algorithm to build the mmodule tree.

Question

For Robinson spaces, are the mmodule tree and PQ-tree identical? Or at least can we build the PQ-tree from the mmodule tree?

At least, the order of children of Q-nodes matters, while the order of children of $\cap$-nodes does not.

Question

When restricted to a Robinson dissimilarity whose PQ-tree is a single Q-node, can we find the compatible order efficiently?

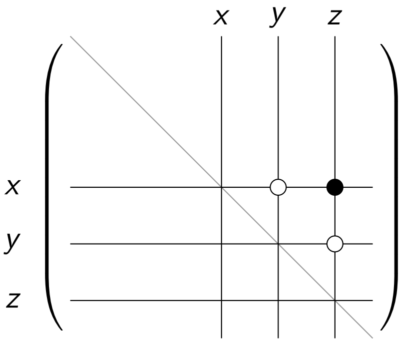
LIS
LABORATOIRE D'INFORMATIQUE & SYSTÈMES

An alternative definition for Robinson

Lemma

(X, d) is a Robinson space if and only if there is an order $<$ such that for any $x < y < z$,

$$\max\{d(x, y), d(y, z)\} \leq d(x, z).$$

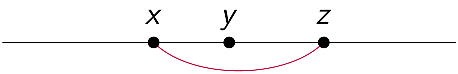


An alternative definition for Robinson

Lemma

(X, d) is a Robinson space if and only if there is an order $<$ such that for any $x < y < z$,

$$\max\{d(x, y), d(y, z)\} \leq d(x, z).$$



Block

Definition

Block of a set of permutations/orders on X: subset $B \subseteq X$ such that the elements of B are consecutive (an interval) in any of these permutations.

Lemma

Given a PQ-tree on X with block B,

(i) *either there is a node α with $B = X(\alpha)$,*

(ii) *or there is a Q-node $\alpha = Q(\beta_1, \dots, \beta_k)$ such that $B = X(\beta_i) \cup X(\beta_{i+1}) \cup \dots \cup X(\beta_j)$.*

PQ-nodes are mmodules

Lemma

Let α be a node of the PQ-tree for (X, d), then $X(\alpha)$ is an mmodule.

Proof. Let $x, y \in X(\alpha)$, $z \notin X(\alpha)$, and $<$ compatible order with $x < y < z$. Then $d(y, z) \leq d(x, z)$.

Reversing the order of $X(\alpha)$ in $<$ gives a compatible order $<'$ with $y <' x <' z$. Then $d(x, z) \leq d(y, z)$.

Thus $d(x, z) = d(y, z)$.

Characterization of PQ-nodes

Theorem

$M \subseteq X$ is a block and an mmodule iff there is a node α in the PQ-tree such that $M = X(\alpha)$.

Proof: show that for $Q(\beta_1, \ldots, \beta_k)$, for $i < j$ with $(i, j) \neq (1, k)$, $X(\beta_i) \cup \ldots \cup X(\beta_j)$ is not an mmodule.

LABORATOIRE D'INFORMATIQUE & SYSTÈMES

Flat Robinson spaces

Definition

Flat Robinson space: a Robinson space having only two compatible orders (reverse from each other).

Example: line distances are flat.

Flat Robinson space have PQ-tree reduced to a single internal node of type Q.

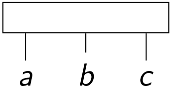
Corollary

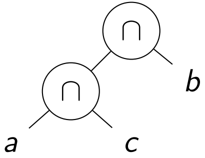
If all the mmodules are trivial (X and one-element sets), then (X, d) is flat and its PQ-tree has a single node of type Q.

Flat Robinson spaces have single node?

Is the converse true? Are the mmodules of flat Robinson spaces always trivial? No!

D	a	b	c
a	0	1	2
b		0	1
c			0





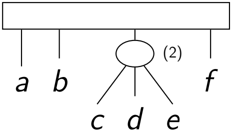
Consequence

PQ-tree and mmodule tree are not similar!

Conical node

Definition

conical node: a Q-node $Q(\beta_1, \ldots, \beta_k)$ with a (unique) child β_i and δ such that $d(\beta_i, \beta_j) = \delta$ for each $j \neq i$.
apex child: the child β_i in that case.
split child: β_i when it is a P-node with associated value δ.



$$\begin{pmatrix} 0 & 1 & 2 & 4 \\ & 0 & 2 & 3 \\ & & 0 & 2 \\ & & & 0 \end{pmatrix}$$

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

Conical nodes and flat Robinson spaces

Lemma

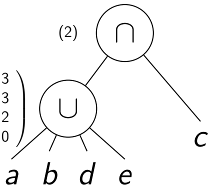
If (X, d) is flat Robinson space,

(i) *either all its mmodule are trivial,*

(ii) *or there is $p \in X$ and δ with $d(p,x) = \delta$ for all $x \in X \setminus \{p\}$. $X \setminus \{p\}$ is the only non-trivial mmodule. Also p is not in a diametral pair.*

In case (ii), the Q-root is conical, with leaf p apex.

$$\begin{pmatrix} 0 & 1 & 3 & 3 \\ & 0 & 2 & 3 \\ & & 0 & 2 \\ & & & 0 \end{pmatrix}$$



	a	b	c	d	e
a	0	1	2	3	3
b		0	2	2	3
c			0	2	2
d				0	2
e					0

Special $\cap$-node and large child

Definition

Special $\cap$-node: a $\cap$-node whose associated value is less than the diameter of one of its child.
Large child: the child (unique for Robinson space) of a $\cap$-node whose diameter is more than the $\cap$-node associated value

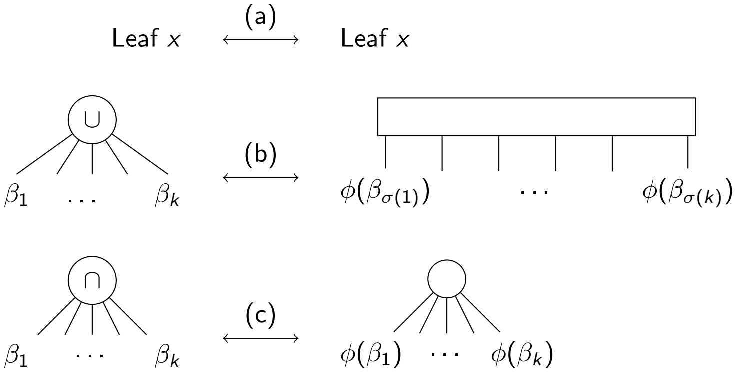
Conical Q-node and special $\cap$-node are the only bad cases, for which the correspondance

$$\cap\text{-node} \leftrightarrow \text{P-node}$$
$$\cup\text{-node} \leftrightarrow \text{Q-node}$$

does not work.

From mmodule trees to PQ-trees and back

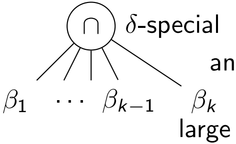
Easy cases (no special node, no conical node):



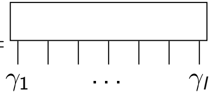
In case (b), it requires σ.

From mmodule trees to PQ-trees and back

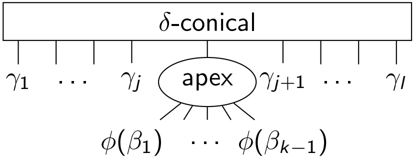
Bad case: special $\cap$-node to conical Q-node.



and $\phi(\beta_k) =$

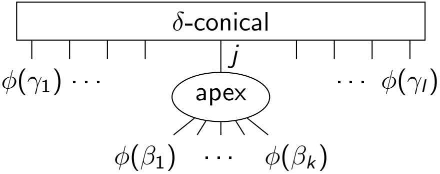


translates into

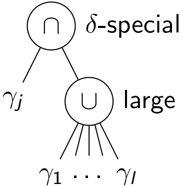


From mmodule trees to PQ-trees and back

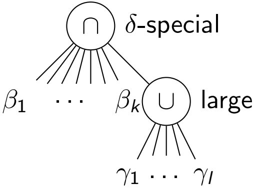
Bad case: conical Q-node to special $\cap$-node.



translates into



or



γ_j is standard

γ_j is split

Translation

This gives:

Theorem

Given a Robinson space, one can build the mmodule tree from the PQ-tree, and the PQ-tree from the module tree in time $O(|X|)$ (without counting the time spent to order children of Q-nodes).

Solving the flat Robinson case

It remains to show how to order the children of Q-node. More generally: find the compatible order (up to reversal) for a flat Robinson space.

1. choose arbitrarily a pivot p,
2. sort vertices by their *proximity* to p,
3. choose for each vertex its side, left or right of p.

Step 2 is based on the partition refinement algorithm (used to build the mmodule tree).

The partition refinement algorithm

(used by Erhenfeucht at al. to build the mmodule tree)

Input: a partition $\mathcal{P}$ of X

Output: a partition $\mathcal{P}'$, refining $\mathcal{P}$ (for all $S' \in \mathcal{P}'$, there is $S \in \mathcal{P}$ with $S' \subseteq S$), where each $S' \in \mathcal{P}'$ is an mmodule.

Idea: Keep for each part S a set of candidate *distinguishers* Z_S.

Invariant: For any $x, y \in S$, if there is $z \in X \setminus S$ with $d(x, z) \neq d(y, z)$, then $z \in Z_S$.

Procedure: Iteratively pick z in some Z_S, and refine S depending on the distances from z.

LIS LABORATOIRE D'INFORMATIQUE & SYSTÈMES

The partition refinement algorithm

$$
\left(\begin{array}{c|ccc|cccc}
 & a & b & c & d & e & f & g \\
\hline
a & 0 & 2 & 2 & 4 & 4 & 4 & 5 \\
b & & 0 & 2 & 4 & 4 & 4 & 5 \\
c & & & 0 & 3 & 3 & 4 & 4 \\
\hline
d & & & & 0 & 1 & 2 & 2 \\
e & & & & & 0 & 2 & 2 \\
f & & & & & & 0 & 2 \\
g & & & & & & & 0
\end{array}\right)
$$

$$\{(S, Z_S) : S \in \mathcal{P}\} = \{(abc, defg), (defg, abc)\}$$

The partition refinement algorithm

$$
\left(\begin{array}{c|ccc|cccc}
 & a & b & c & d & e & f & g \\
\hline
a & 0 & 2 & 2 & 4 & 4 & 4 & 5 \\
b & & 0 & 2 & 4 & 4 & 4 & 5 \\
c & & & 0 & 3 & 3 & 4 & 4 \\
\hline
d & & & & 0 & 1 & 2 & 2 \\
e & & & & & 0 & 2 & 2 \\
f & & & & & & 0 & 2 \\
g & & & & & & & 0
\end{array}\right)
$$

$$\{(abc, defg), (defg, abc)\}$$

$$\longrightarrow \{(ab, cefg), (c, abefg)(defg, abc)\}$$

The partition refinement algorithm

$$
\left(\begin{array}{c|cc|c|cccc}
 & a & b & c & d & e & f & g \\
\hline
a & 0 & 2 & 2 & 4 & 4 & 4 & 5 \\
b & & 0 & 2 & 4 & 4 & 4 & 5 \\
\hline
c & & & 0 & 3 & 3 & 4 & 4 \\
\hline
d & & & & 0 & 1 & 2 & 2 \\
e & & & & & 0 & 2 & 2 \\
f & & & & & & 0 & 2 \\
g & & & & & & & 0
\end{array}\right)
$$

$$\{(ab, cefg), (c, abefg)(defg, abc)\}$$

$$\longrightarrow \{(ab, \emptyset), (c, abefg), (defg, abc)\}$$

The partition refinement algorithm

$$
\left(\begin{array}{c|cc|c|cccc}
 & a & b & c & d & e & f & g \\
\hline
a & 0 & 2 & 2 & 4 & 4 & 4 & 5 \\
b & & 0 & 2 & 4 & 4 & 4 & 5 \\
\hline
c & & & 0 & 3 & 3 & 4 & 4 \\
\hline
d & & & & 0 & 1 & 2 & 2 \\
e & & & & & 0 & 2 & 2 \\
f & & & & & & 0 & 2 \\
g & & & & & & & 0
\end{array}\right)
$$

$$\{(ab,\emptyset),(c,\emptyset)(defg,abc)\}$$

$$\longrightarrow \{(ab,\emptyset),(c,\emptyset),(def,gbc),(g,defbc)\}$$

The partition refinement algorithm

$$
\left(\begin{array}{c|cc|c|ccc|c}
 & a & b & c & d & e & f & g \\
\hline
a & 0 & 2 & 2 & 4 & 4 & 4 & 5 \\
b & & 0 & 2 & \textcolor{red}{4} & \textcolor{red}{4} & \textcolor{red}{4} & 5 \\
\hline
c & & & 0 & 3 & 3 & 4 & 4 \\
\hline
d & & & & 0 & 1 & 2 & \textcolor{green}{2} \\
e & & & & & 0 & 2 & \textcolor{green}{2} \\
f & & & & & & 0 & \textcolor{green}{2} \\
\hline
g & & & & & & & 0
\end{array}\right)
$$

$$\{(ab, \emptyset), (c, \emptyset), (def, \textcolor{green}{g}\,\textcolor{red}{b}c), (g, defbc)\}$$

$$\longrightarrow \{(ab, \emptyset), (c, \emptyset), (def, c), (g, defbc)\}$$

The partition refinement algorithm

$$\begin{array}{cc}
 & \begin{array}{cc|c|ccc|c} a & b & c & d & e & f & g \end{array} \\
\begin{array}{c} a \\ b \\ c \\ d \\ e \\ f \\ g \end{array} & \left(\begin{array}{cc|c|ccc|c}
0 & 2 & 2 & 4 & 4 & 4 & 5 \\
 & 0 & 2 & 4 & 4 & 4 & 5 \\ \hline
 & & \color{green}{0} & \color{red}{3} & \color{red}{3} & \color{teal}{4} & 4 \\ \hline
 & & & 0 & 1 & 2 & 2 \\
 & & & & 0 & 2 & 2 \\
 & & & & & 0 & 2 \\ \hline
 & & & & & & 0
\end{array}\right)
\end{array}$$

$$\{(ab, \emptyset), (c, \emptyset), (def, \color{green}{c}), (g, defbc)\}$$

$$\longrightarrow \{(ab, \emptyset), (c, \emptyset), (\color{red}{de}, \color{teal}{f}), (\color{teal}{f}, \color{red}{de})(g, defbc)\}$$

The partition refinement algorithm

$$\begin{array}{c|cc|c|cc|c|c} & a & b & c & d & e & f & g \\ \hline a & 0 & 2 & 2 & 4 & 4 & 4 & 5 \\ b & & 0 & 2 & 4 & 4 & 4 & 5 \\ \hline c & & & 0 & 3 & 3 & 4 & 4 \\ \hline d & & & & 0 & 1 & \color{green}{2} & 2 \\ e & & & & & 0 & \color{green}{2} & 2 \\ \hline f & & & & & & 0 & 2 \\ \hline g & & & & & & & 0 \end{array}$$

$$\{(ab, \emptyset), (c, \emptyset), (de, \color{green}{f}), (f, de)(g, defbc)\}$$

$$\{(ab, \emptyset), (c, \emptyset), (de, \emptyset), (f, \emptyset)(g, \emptyset)\}$$

The partition refinement algorithm

$$\left(\begin{array}{c|cc|c|cc|c|c}
 & a & b & c & d & e & f & g \\
\hline
a & 0 & 2 & 2 & 4 & 4 & 4 & 5 \\
b & & 0 & 2 & 4 & 4 & 4 & 5 \\
\hline
c & & & 0 & 3 & 3 & 4 & 4 \\
\hline
d & & & & 0 & 1 & 2 & 2 \\
e & & & & & 0 & 2 & 2 \\
\hline
f & & & & & & 0 & 2 \\
\hline
g & & & & & & & 0
\end{array}\right)$$

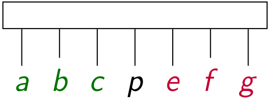
$$\mathcal{P}' = \{ab, c, de, f, g\}$$

p-proximity order

Definition

p-proximity order for a compatible order $<$: a total order $\prec$ for some $p \in X$, such that:

1. p is the minimum,
2. for all $q \in X$, $\{q\} \cup \{x \in X \ : \ x \prec q\}$ is an interval of $<$.



- $p \prec c \prec b \prec e \prec a \prec f \prec g$,
- $p \prec e \prec c \prec f \prec b \prec a \prec g$,
- $p \prec e \prec f \prec g \prec c \prec b \prec a$,
- $p \prec c \prec e \prec b \prec f \prec g \prec a$, ...

p-proximity order

$p \prec x \prec y$ is equivalent to saying that y is not between p and x in $<$.

Lemma

Let $x, y \in X \setminus \{p\}$, y is not between p and x if

(i) *either $d(p,x) < d(p,y)$;*

(ii) *or $d(p,x) = d(p,y)$ and there is $q \in X$ with $q \prec x$, $q \prec y$ and $d(q,x) < d(q,y)$;*

(iii) *or $d(p,x) = d(p,y)$ and there is $q \in X$ with $x \prec q$, $y \prec q$ and $d(y,q) < d(x,q)$.*

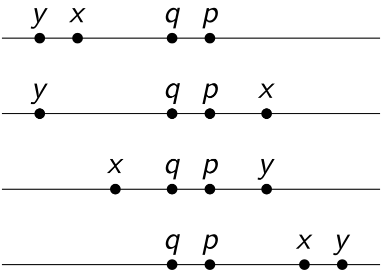
Case (ii): q is an *in-pivot*, case (iii): q is an *out-pivot*.

LABORATOIRE D'INFORMATIQUE & SYSTÈMES

Proof

Case (ii). Let $x, y \in X \setminus \{p\}$ with $d(p,x) = d(p,y)$. Let $q \in X$ with $q \prec x$, $q \prec y$ and $d(q,x) < d(q,y)$. Assume $q < p$.

Possible cases:



In any case, y is not between p and x.

Computing a p-proximity order

Modify the partition refinement algorithm:

$$(S, Z) \quad \Longrightarrow \quad (S, \mathrm{In}_S, \mathrm{Out}_S) \text{ with } \mathrm{In}_S \prec S \prec \mathrm{Out}_S$$

When refining S with in-pivot $q \in In_S$: partition S into

$$S_1 \cup S_2 \cup \ldots \cup S_k \text{ with } d(q, S_1) < d(q, S_2) < \ldots < d(q, S_k)$$

Set $S_1 \prec S_2 \prec \ldots \prec S_k$.
Set $\mathrm{In}_{S_i} = S_1 \cup \ldots \cup S_{i-1} \cup \mathrm{In}_S \setminus \{q\}$.
Set $\mathrm{Out}_{S_i} = S_{i+1} \cup \ldots \cup S_k \cup \mathrm{Out}_S$.

Slightly more complicated rules when $q \in \mathrm{Out}_S$.

Computing a p-proximity order

$$\left(\begin{array}{c|cccccc} & a & b & c & p & e & f \\ \hline a & 0 & 2 & 4 & 5 & 6 & 6 \\ b & & 0 & 2 & 3 & 4 & 6 \\ c & & & 0 & 3 & 4 & 6 \\ p & & & & 0 & 3 & 5 \\ e & & & & & 0 & 1 \\ f & & & & & & 0 \end{array}\right)$$

$$(p, \emptyset, \emptyset) \prec (abcef, p, \emptyset)$$

$$(p, \emptyset, \emptyset) \prec (bce, \emptyset, af) \prec (af, bce, \emptyset)$$

Computing a p-proximity order

$$\left(\begin{array}{c|cccccc} & a & b & c & p & e & f \\ \hline a & 0 & 2 & 4 & 5 & 6 & 6 \\ b & & 0 & 2 & 3 & 4 & 6 \\ c & & & 0 & 3 & 4 & 6 \\ p & & & & 0 & 3 & 5 \\ e & & & & & 0 & 1 \\ f & & & & & & 0 \end{array}\right)$$

$$(p, \emptyset, \emptyset) \prec (bce, \emptyset, af) \prec (af, bce, \emptyset)$$

$$(p, \emptyset, \emptyset) \prec (c, \emptyset, bef) \prec (b, c, ef) \prec (e, cb, f) \prec (af, bce, \emptyset)$$

Computing a p-proximity order

$$
\left(\begin{array}{c|cccccc}
 & a & b & c & p & e & f \\
\hline
a & 0 & 2 & 4 & 5 & 6 & 6 \\
b & & 0 & 2 & 3 & 4 & 6 \\
c & & & 0 & 3 & 4 & 6 \\
p & & & & 0 & 3 & 5 \\
e & & & & & 0 & 1 \\
f & & & & & & 0
\end{array}\right)
$$

$$(p, \emptyset, \emptyset) \prec (c, \emptyset, \emptyset) \prec (b, \emptyset, \emptyset) \prec (e, \emptyset, \emptyset) \prec (af, bce, \emptyset)$$

$$(p, \emptyset, \emptyset) \prec (c, \emptyset, \emptyset) \prec (b, \emptyset, \emptyset) \prec (e, \emptyset, \emptyset) \prec (a, ce, f) \prec (f, ace, \emptyset)$$

Computing a p-proximity order

$$
\left(\begin{array}{c|cccccc}
 & a & b & c & p & e & f \\
\hline
a & 0 & 2 & 4 & 5 & 6 & 6 \\
b & & 0 & 2 & 3 & 4 & 6 \\
c & & & 0 & 3 & 4 & 6 \\
p & & & & 0 & 3 & 5 \\
e & & & & & 0 & 1 \\
f & & & & & & 0
\end{array}\right)
$$

$$(p, \emptyset, \emptyset) \prec (c, \emptyset, \emptyset) \prec (b, \emptyset, \emptyset) \prec (e, \emptyset, \emptyset) \prec (a, \emptyset, \emptyset) \prec (f, \emptyset, \emptyset)$$

$$p \prec c \prec b \prec e \prec a \prec f$$

LABORATOIRE D'INFORMATIQUE & SYSTÈMES

How to use the p-proximity order

- Choose p that is not apex.
- Refine $\{\{p\}, X \setminus \{p\}\}$. Because no module (except possibly $X \setminus \{\mathrm{apex}\}$), the partition contains one-element sets only.
- Obtain a p-proximity order $\prec$.

$$p \prec e_1 \prec e_2 \prec \ldots \prec e_k$$

Next goal

Partition $X \setminus \{p\}$ into two sides $L \cup R$: elements at the left (resp. right) of p in $<$.

$$\prec \quad + \quad L \cup R \quad \Longrightarrow \quad <$$

LIS Laboratoire d'Informatique & Systèmes

Choosing side

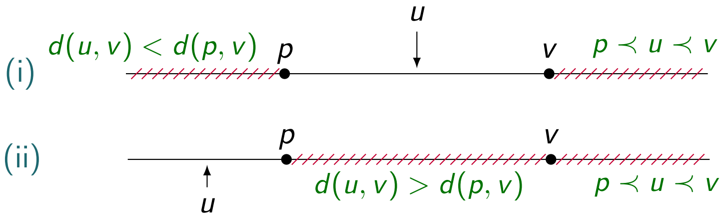
Lemma

Let $\prec$ be a p-proximity order, and $u \prec v$.

(i) $d(u,v) < d(p,v)$ *implies* $\mathrm{side}(u) = \mathrm{side}(v)$,

(ii) $d(u,v) > d(p,v)$ *implies* $\mathrm{side}(u) \neq \mathrm{side}(v)$.

Proof.



The constraint graph G

Define the constraint graph $G = (V, E)$:

$$\begin{aligned} V &= X \setminus \{p\} \\ E &= \{(u, v) \ : \ u \prec v \wedge d(u, v) \neq d(p, v)\} \end{aligned}$$

If G has a single connected component: pick a side for an arbitrary $x \in X \setminus \{p\}$, propagate to deduce L and R.

The component graph H

Definition

Tangled components C, C': C and C' are not comparable under $\prec$.

Define the component graph $H = (K, F)$:

$$K = \{C : C \text{ connected component of G}\}$$
$$F = \{(C, C') \ : \ C, C' \text{ are tangled}\}$$

Analysis of tangled components

Suppose C, C' are tangled. We may assume $x, z \in C$, $y \in C'$ with $xz \in E$ and $x \prec y \prec z$.

- $d(p, y) = d(x, y)$ (as $xy \notin E$),
- $d(p, z) = d(y, z)$ (as $yz \notin E$).

Lemma

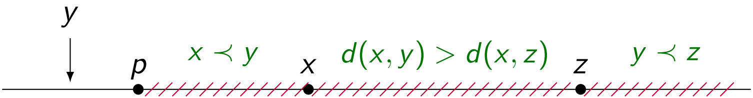
(i) *if* $d(x, z) < d(p, z)$, *then* $\mathrm{side}(x) = \mathrm{side}(z) \neq \mathrm{side}(y)$,

(ii) *if* $d(x, z) > d(p, z)$, *then* $\mathrm{side}(x) \neq \mathrm{side}(z) = \mathrm{side}(y)$

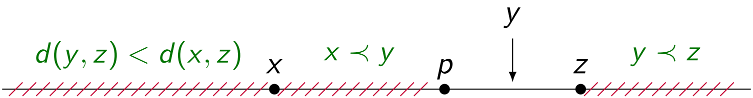
Thus if H has a single connected component: pick a side for an arbitrary $x \in X \setminus \{p\}$, propagate to deduce L and R.

We have $d(x,y) = d(p,y)$ and $d(y,z) = d(p,z)$ and $x \prec y \prec z$.

(i) If $d(x,z) < d(p,z)$, then $\mathrm{side}(x) = \mathrm{side}(z)$, and $d(x,z) < d(p,z) = d(y,z)$.



(ii) If $d(x,z) > d(p,z)$, then $\mathrm{side}(x) \neq \mathrm{side}(z)$, and $d(y,z) = d(p,z) < d(x,z)$.



Last missing piece

Now we just need:

Lemma

H is connected.

Proof. Let m be the maximum of $\prec$ and M be p plus the vertices not determined by $\mathrm{side}(m)$.

Let $x, y \in M$, $z \in X \setminus M$.
$x \prec z$ and $y \prec z$ (as no entanglement here).
Then $xz, yz \notin E$ thus $d(x,z) = d(p,z) = d(y,z)$.

M is an mmodule.
As (X, d) is flat (and m is not apex), $M = \{p\}$.

Algorithmically

```
let m ∈ X, maximum for ≺
let L = [], R = [m], Undecided = reverse(X \ {p, m})
for q ∈ X \ {p} in decreasing order for ≺ do
    let Skipped = []
    for x ∈ Undecided from first to last do
        if d(x, q) = d(p, q) then
            Skipped ← x · Skipped
        else
            if d(x, q) < d(p, q) ⇔ q ∈ L then
                L ← x · L, R ← Skipped ++ R
            else
                R ← x · R, L ← Skipped ++ L
            Skipped ← []
    Undecided ← reverse(Skipped)
return reverse(L) ++ [p] ++ R
```

proximity order + side bipartition

Flexible framework:

- build mmodule tree, translate to PQ-tree using flat Robinson ordering,
- build ordering of p-copoints (maximal mmodules not containing p), recurse on copoints and build PQ-tree,
- contract p-copoints to get a flat Robinson quotient space, merge compatible orders of copoints with order of quotient space.

Available implementation for the last method.

Open problems

- $o(n^2)$ with additional information (promise to be Robinson, minimum spanning tree for d, ...),
- extension to circular Robinson,
- extension to other topologies than the line,
- 3D-matrices.