ICTAI'14

Limassol
Nov. 10-12, 2014

A Generic Algorithmic Framework to Solve Special Versions of the Set Partitioning Problem

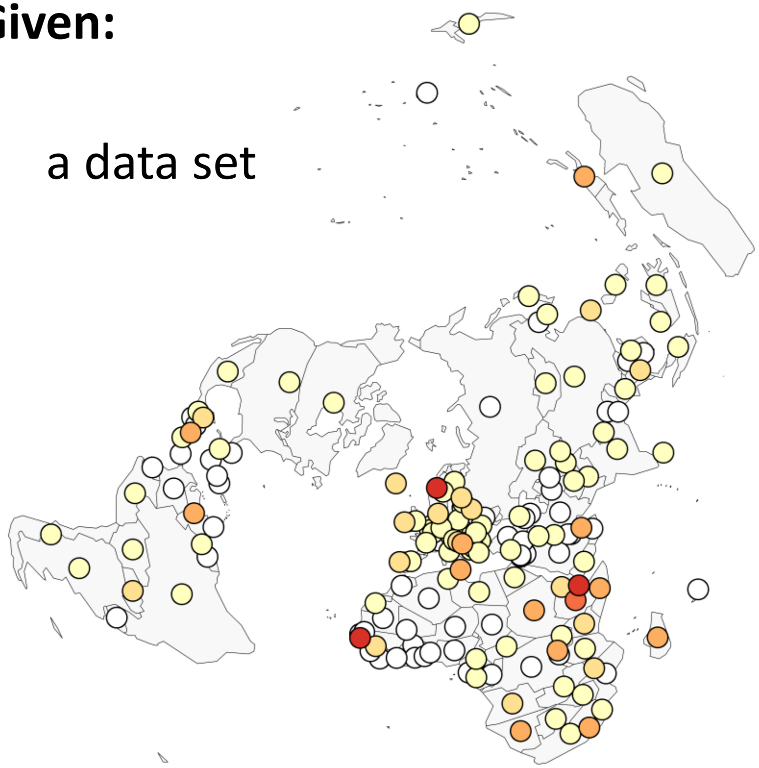
Robin Lamarche-Perrin[1], **Yves Demazeau**[2], **and Jean-Marc Vincent**[2]

[1] Max Planck Institute for Mathematics in the Sciences, Leipzig, Germany
[2] Laboratoire d'Informatique de Grenoble, France

Compression of Geographical Data

Given:

- a data set

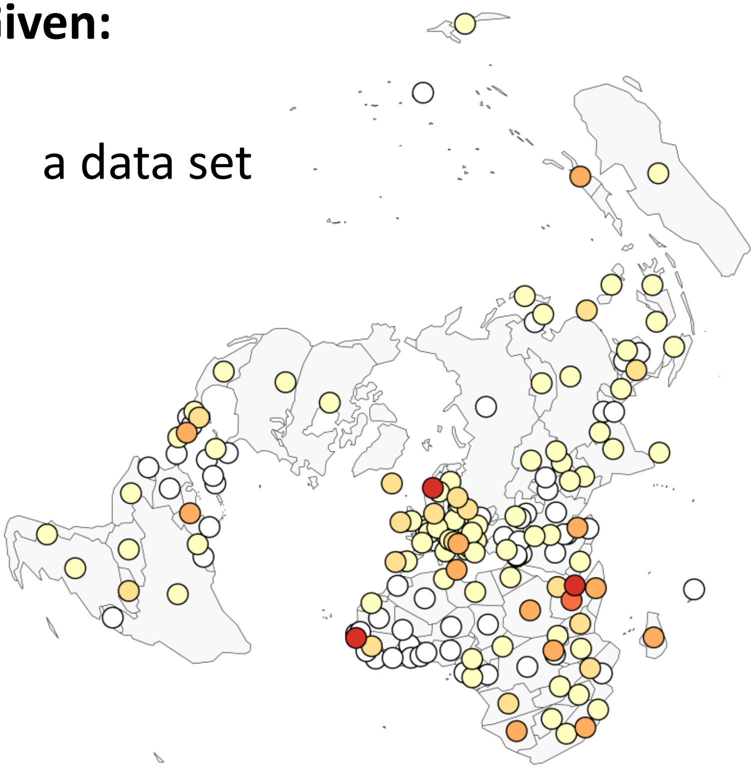


- a measure of information loss

Compression of Geographical Data

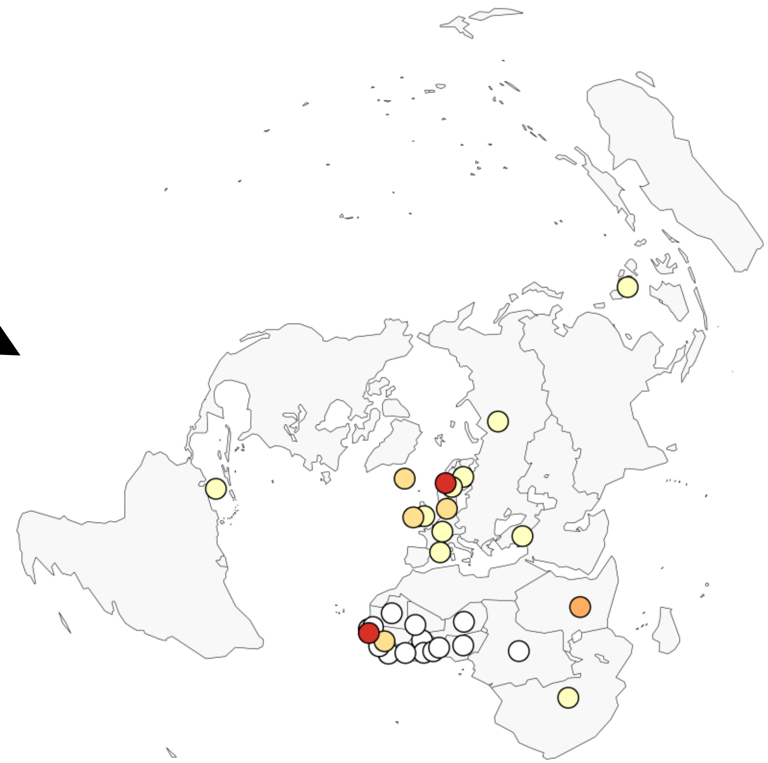
Given:

- a data set

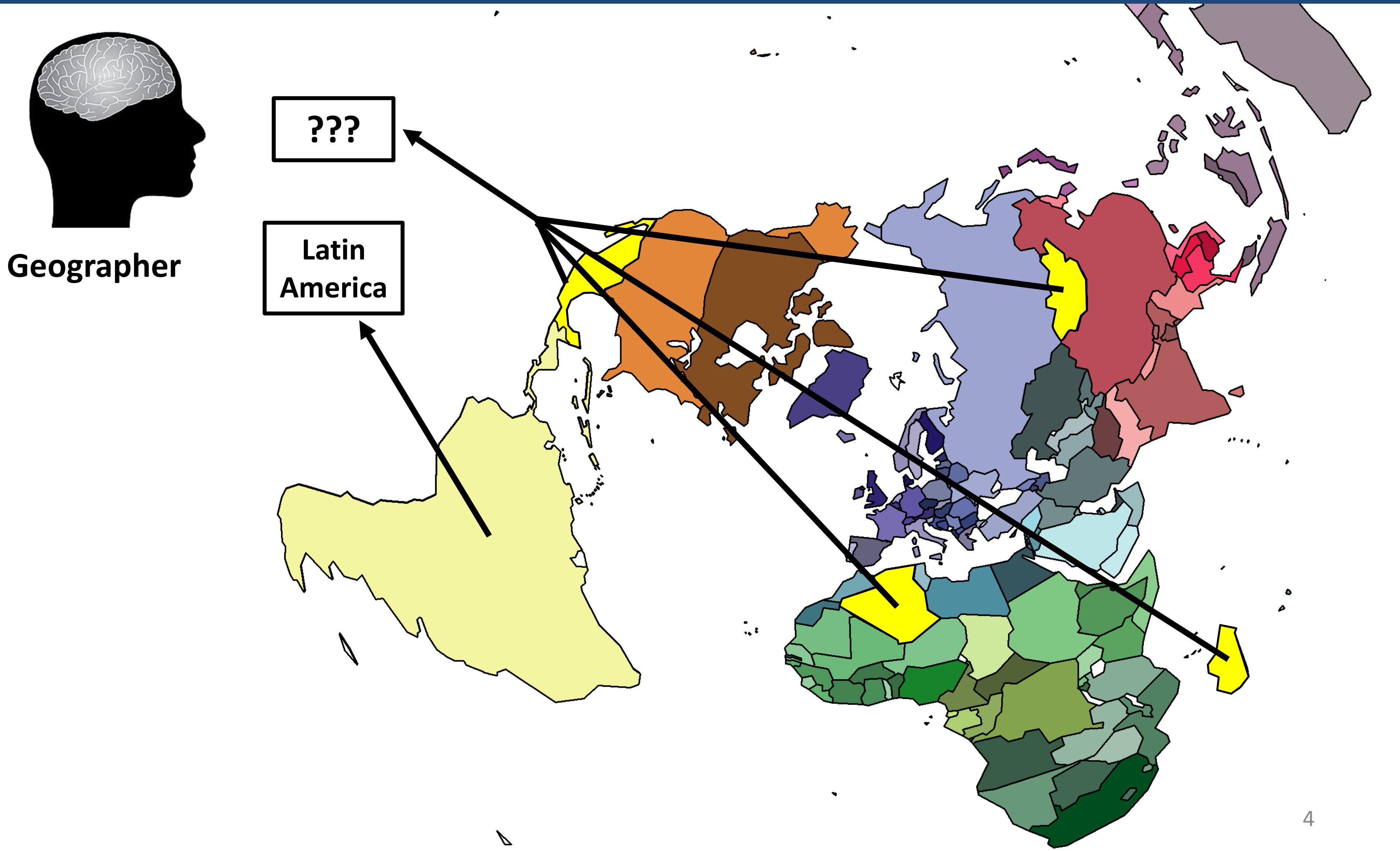


- a measure of information loss

Problem: compress the data while minimizing the information loss

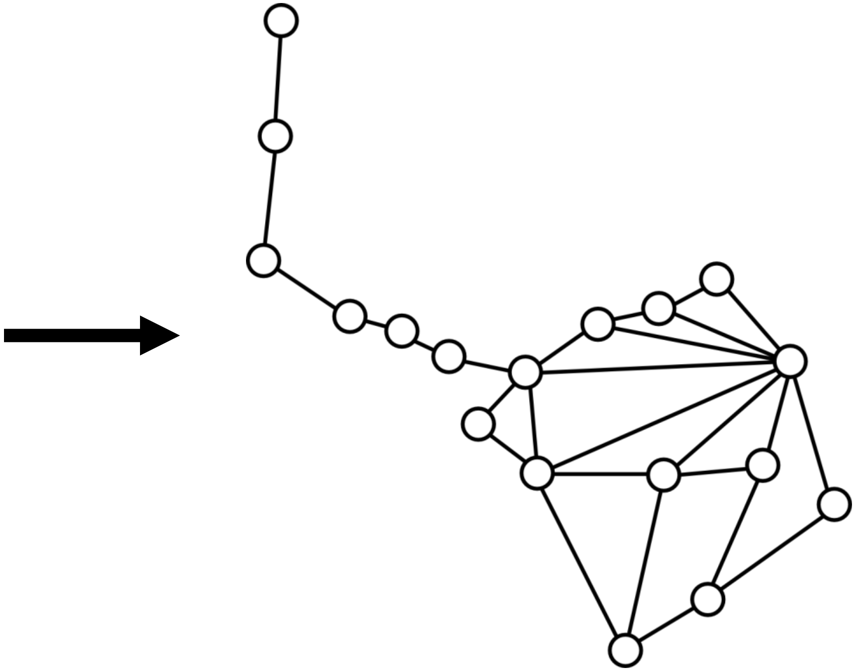


The Semantics of Geographical Aggregates



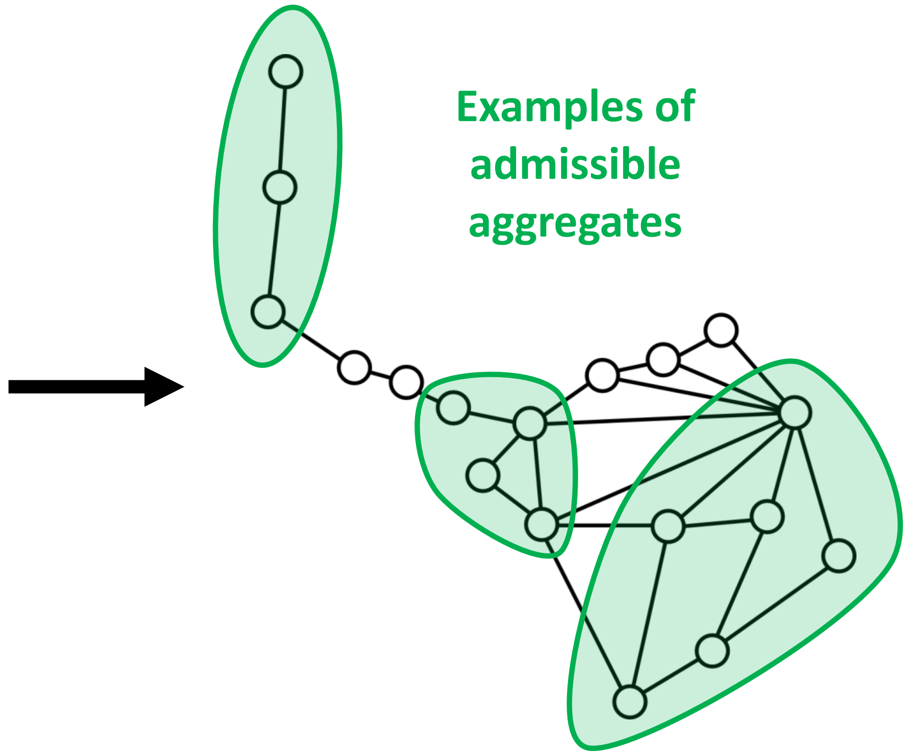
Preserving the Topological Structure

Admissible aggregates = Connected territorial units

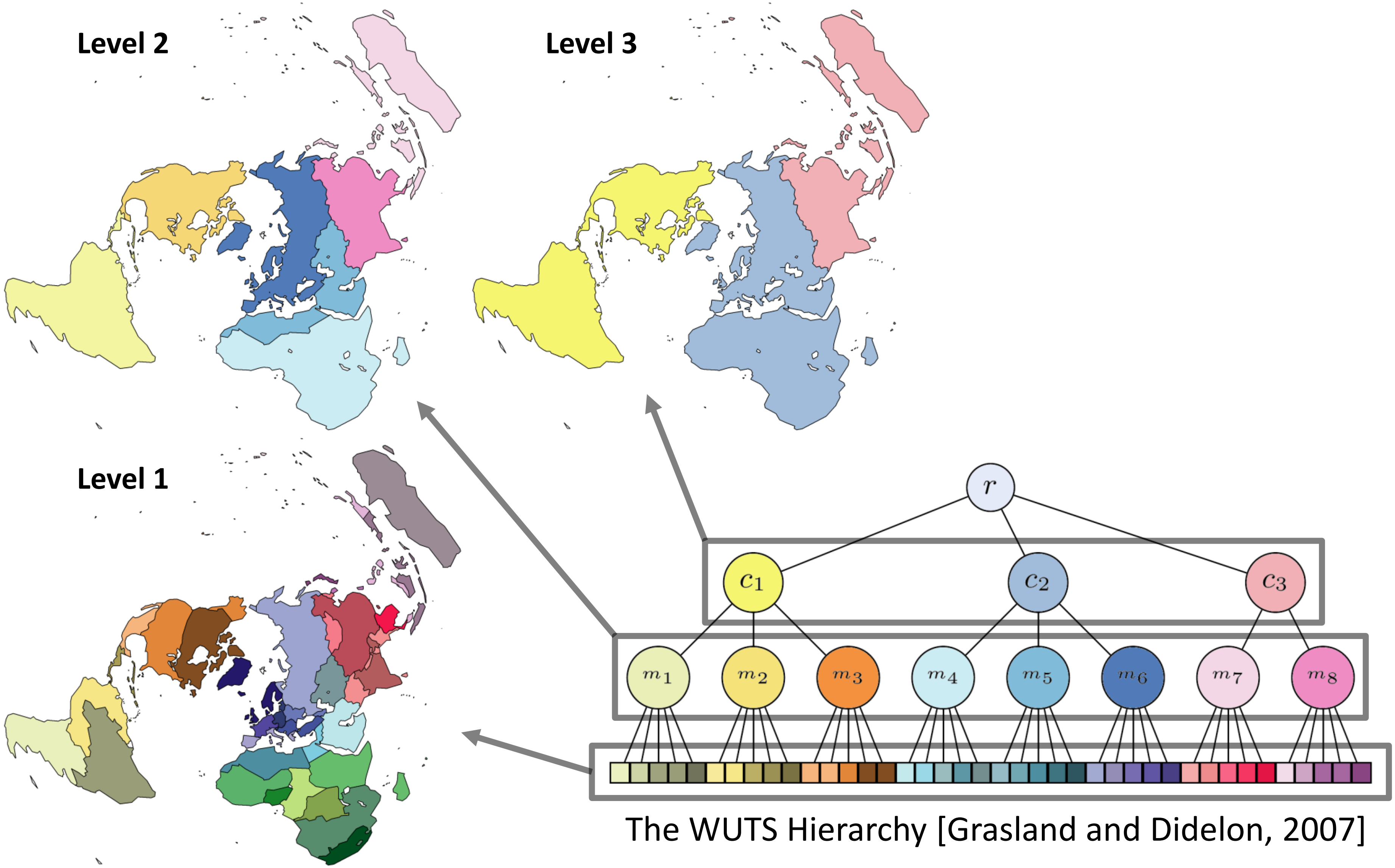


Preserving the Topological Structure

Admissible aggregates = Connected territorial units

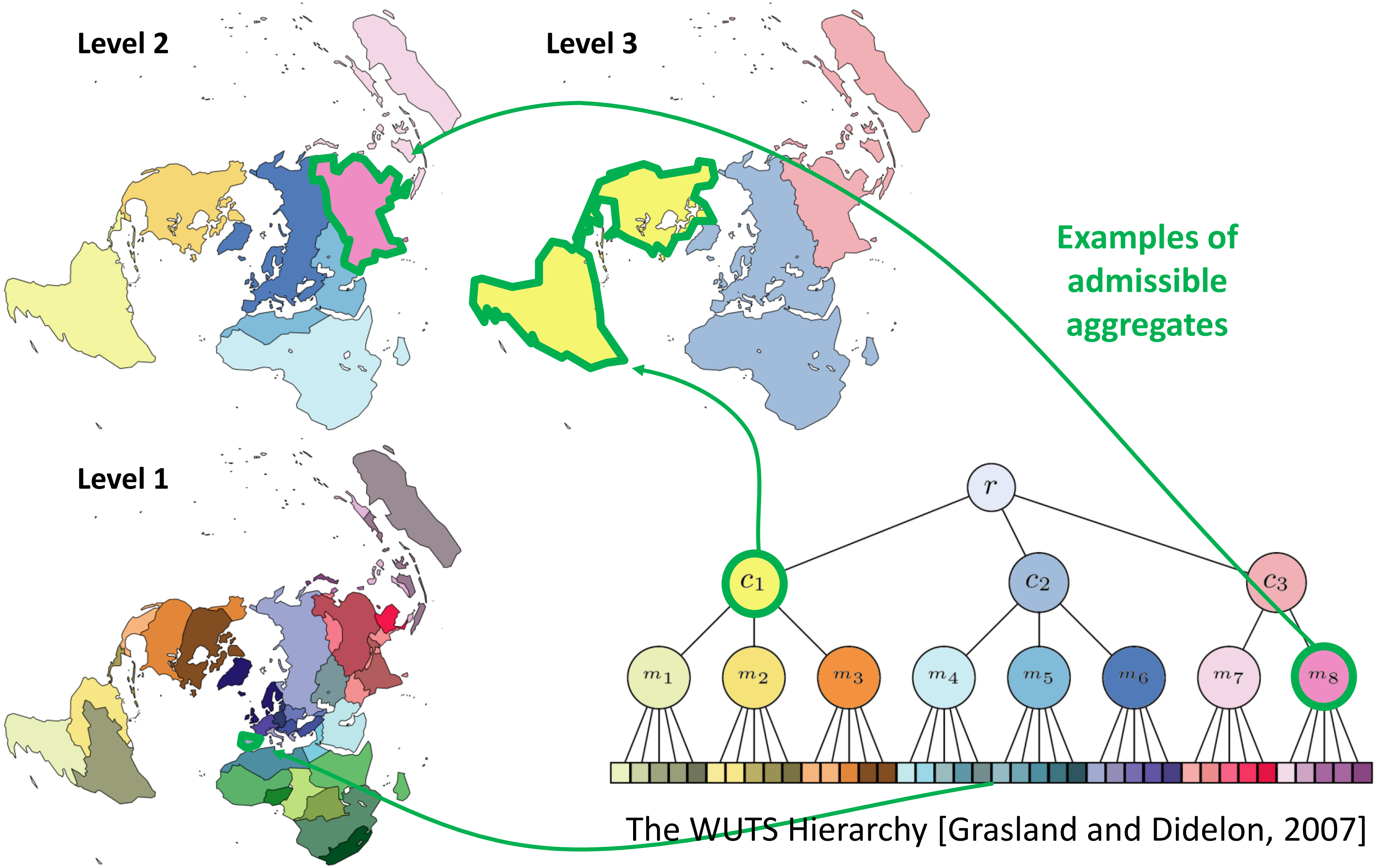


Preserving Social and Political Features



The WUTS Hierarchy [Grasland and Didelon, 2007]

Preserving Social and Political Features

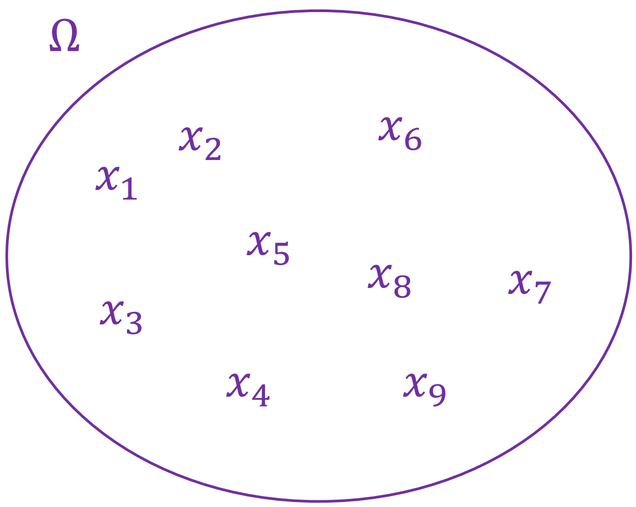


The WUTS Hierarchy [Grasland and Didelon, 2007]

The Set Partitioning Problem

Given:

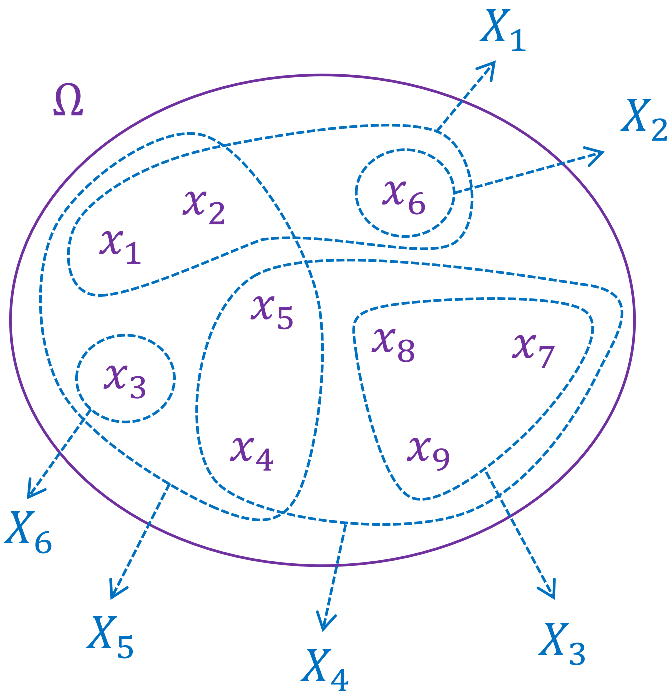
- a set of individuals $\Omega = \{x_1, \dots, x_n\}$



The Set Partitioning Problem

Given:

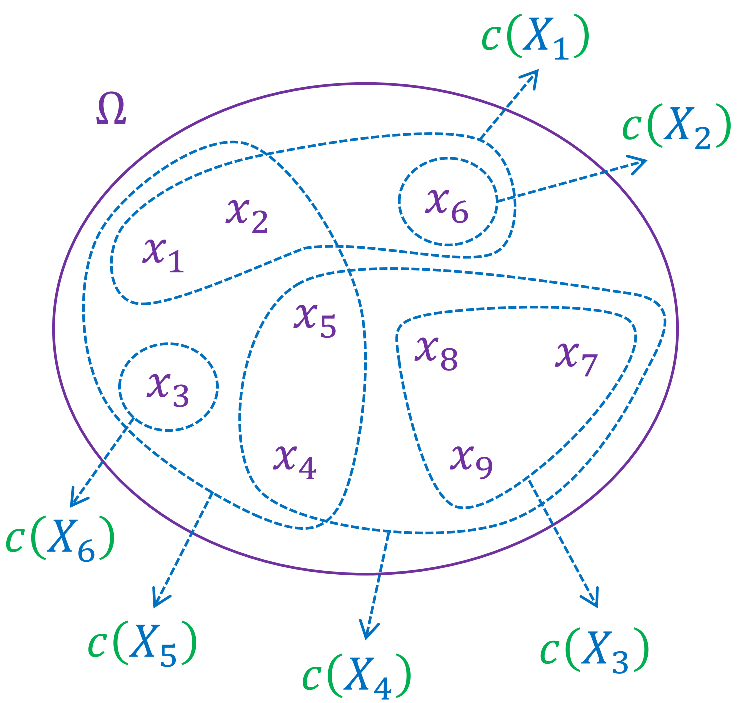
- a set of individuals $\Omega = \{x_1, \dots, x_n\}$
- a set of admissible parts $\mathcal{P} = \{X_1, \dots, X_m\} \subset 2^\Omega$



The Set Partitioning Problem

Given:

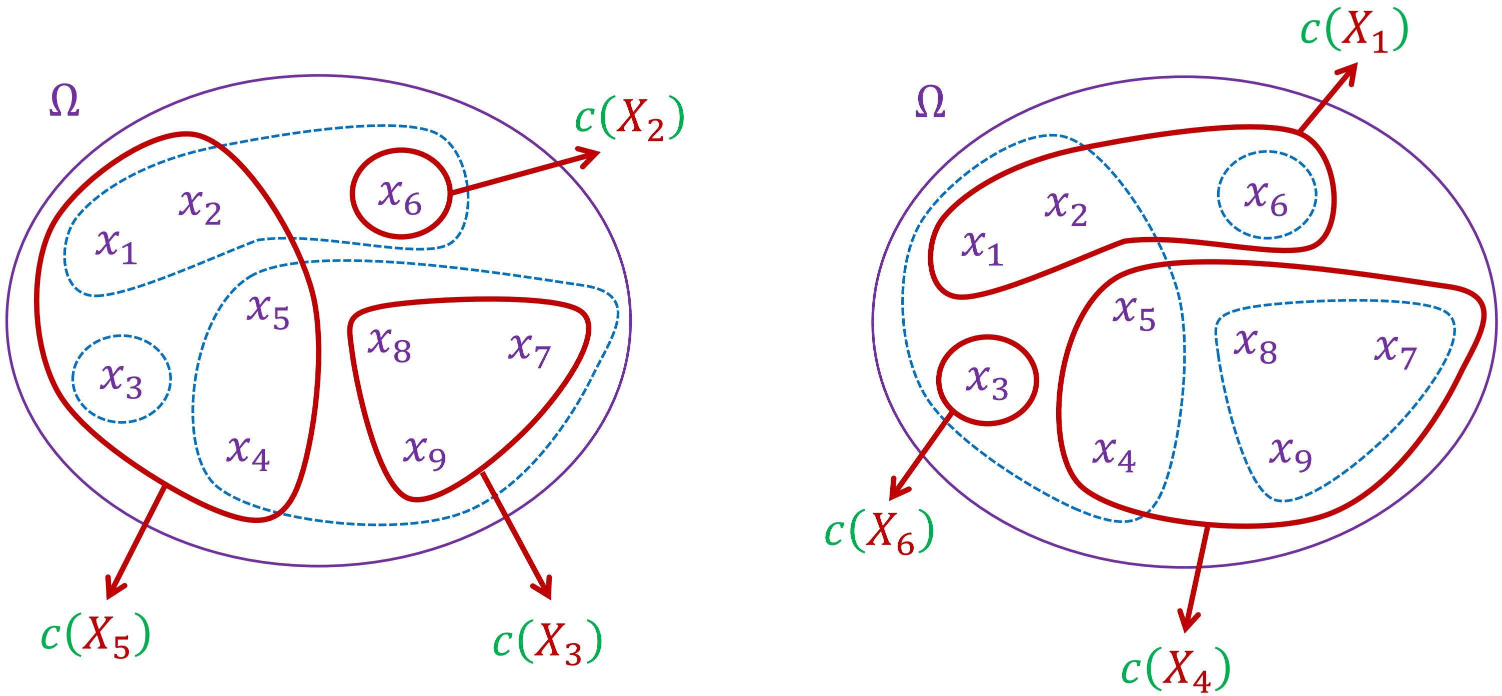
- a set of individuals $\Omega = \{x_1, \dots, x_n\}$
- a set of admissible parts $\mathcal{P} = \{X_1, \dots, X_m\} \subset 2^\Omega$
- a cost function $c : \mathcal{P} \to \mathbb{R}$



The Set Partitioning Problem

Given:

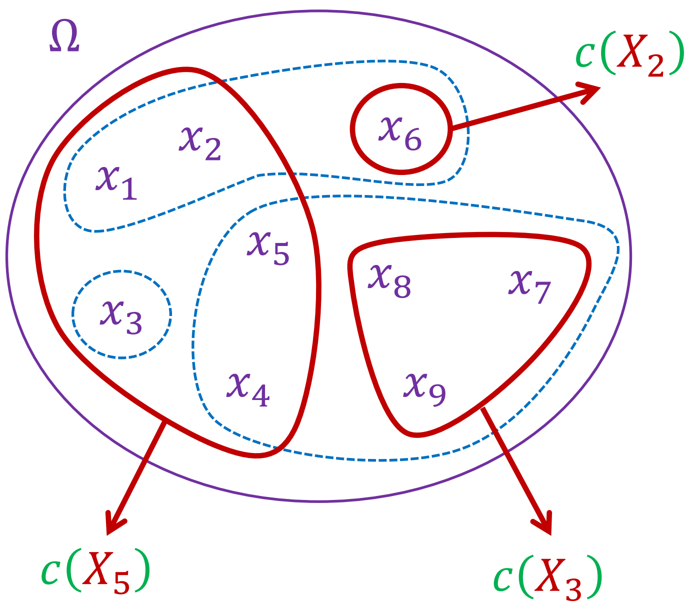
- a set of individuals $\Omega = \{x_1, \dots, x_n\}$
- a set of admissible parts $\mathcal{P} = \{X_1, \dots, X_m\} \subset 2^\Omega$
- a cost function $c : \mathcal{P} \to \mathbb{R}$
- the corresponding set of admissible partitions $\mathfrak{P} = \{\mathcal{X} \subset \mathcal{P}$ such that $\mathcal{X}$ is a partition of $\Omega\}$



The Set Partitioning Problem

Given:

- a set of individuals $\Omega = \{x_1, \ldots, x_n\}$
- a set of admissible parts $\mathcal{P} = \{X_1, \ldots, X_m\} \subset 2^{\Omega}$
- a cost function $c : \mathcal{P} \to \mathbb{R}$
- the corresponding set of admissible partitions $\mathfrak{P} = \{\mathcal{X} \subset \mathcal{P}$ such that $\mathcal{X}$ is a partition of $\Omega\}$



Problem: Find an admissible partition that minimizes the cost function:

$$\mathcal{X}^* = \underset{\mathcal{X} \in \mathfrak{P}}{\arg\min} \left(\sum_{X \in \mathcal{X}} c(X) \right)$$

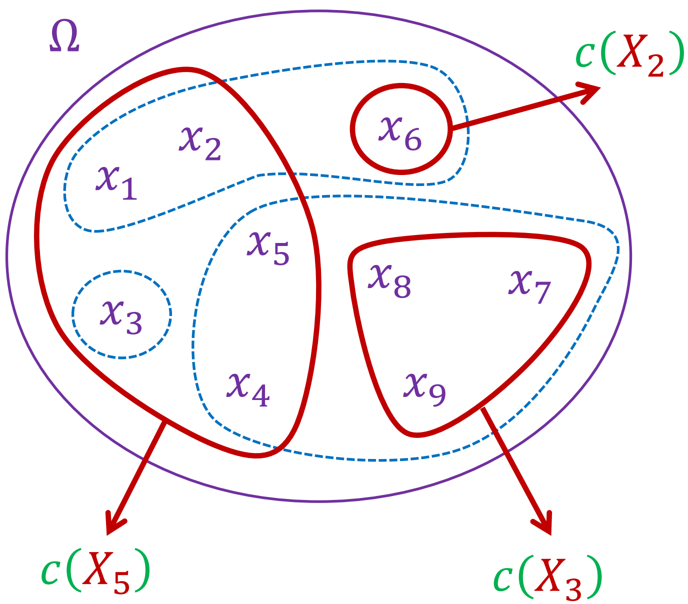
→ **NP-complete!**

The Set Partitioning Problem

Given:

- a set of individuals $\Omega = \{x_1, \dots, x_n\}$
- a set of admissible parts $\mathcal{P} = \{X_1, \dots, X_m\} \subset 2^\Omega$
- a cost function $c : \mathcal{P} \to \mathbb{R}$
- the corresponding set of admissible partitions $\mathfrak{P} = \{\mathcal{X} \subset \mathcal{P}$ such that $\mathcal{X}$ is a partition of $\Omega\}$

Additional assumptions



Problem: Find an admissible partition that minimizes the cost function:

$$\mathcal{X}^* = \arg\min_{\mathcal{X} \in \mathfrak{P}} \left(\sum_{X \in \mathcal{X}} c(X) \right)$$

→ NP-complete!

Applications

Multilevel Geographical Analysis

- $\Omega =$ territorial units
- $\mathcal{P} =$ admissible aggregates
- $c =$ compression rate
- $\mathfrak{P} =$ aggregated representations

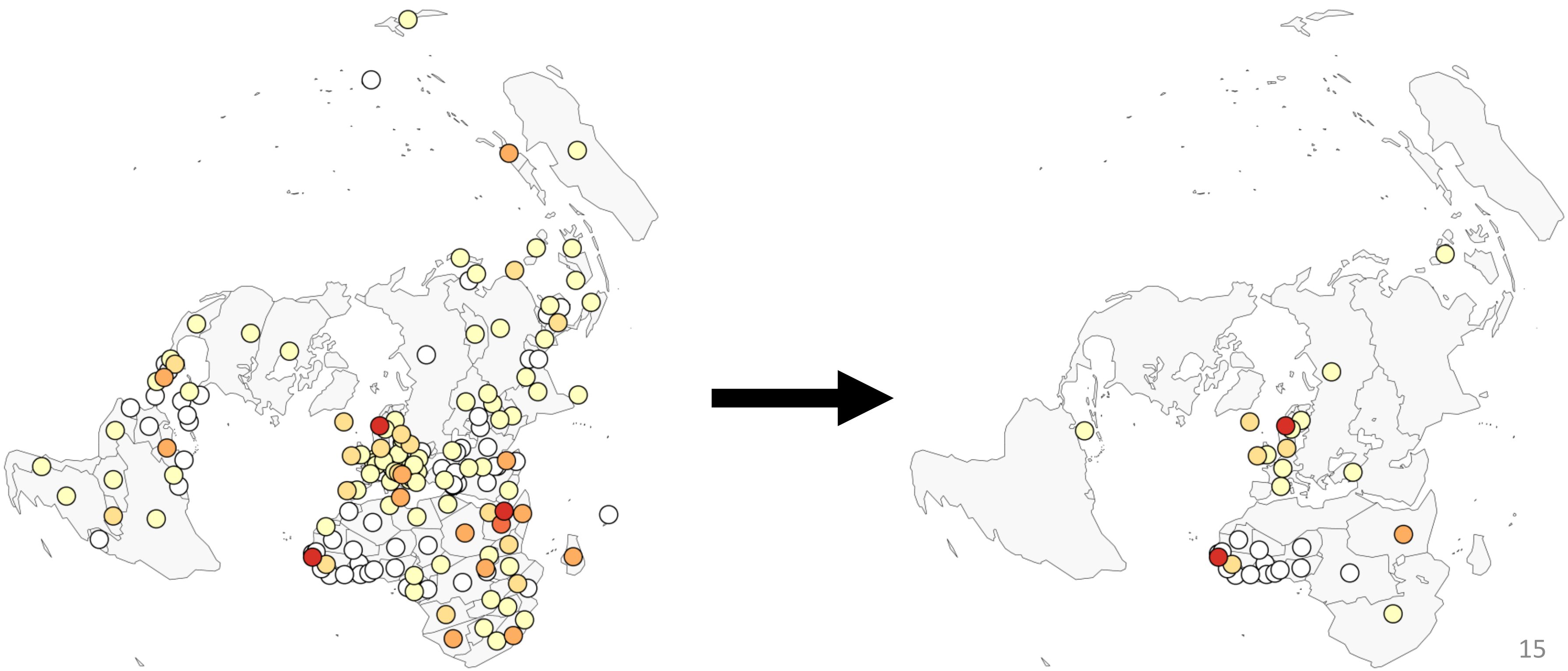
Special Versions

Hierarchical SPP

- Assumption: $\mathcal{P}$ forms a hierarchy
- Result: $\mathcal{O}(n)$ depth-first search
  [Pons *et al.*, 2011] [Lamarche-Perrin *et al.*, 2014]

Graph SPP

- Assumption: $\mathcal{P}$ are connected parts of a graph
- Result: NP-complete [Becker *et al.*, 1998]



Applications

Special Versions

Multilevel Geographical Analysis

Time Series Analysis

- $\Omega =$ ordered data points
- $\mathcal{P} =$ time intervals
- $c =$ compression rate
- $\mathfrak{P} =$ aggregated time series

Hierarchical SPP

- Assumption: $\mathcal{P}$ forms a hierarchy
- Result: $\mathcal{O}(n)$ depth-first search

  [Pons *et al.*, 2011] [Lamarche-Perrin *et al.*, 2014]

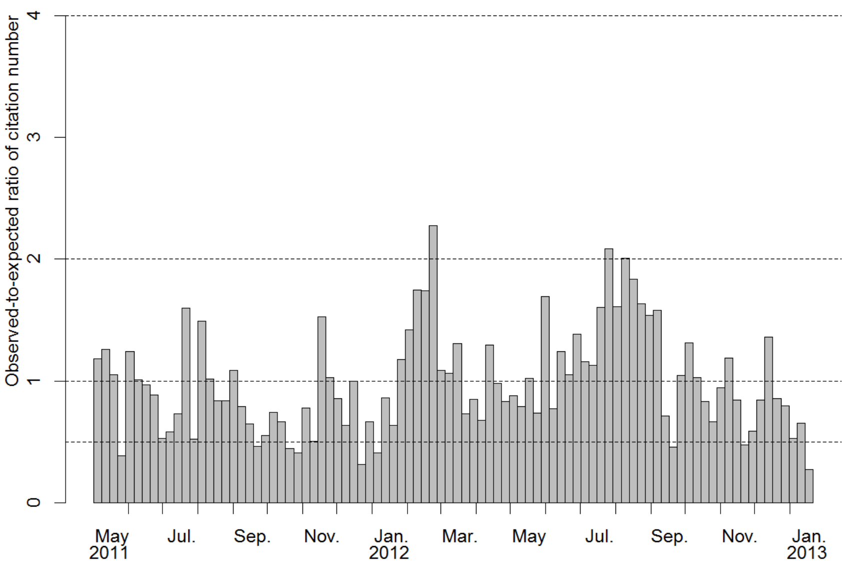
Graph SPP

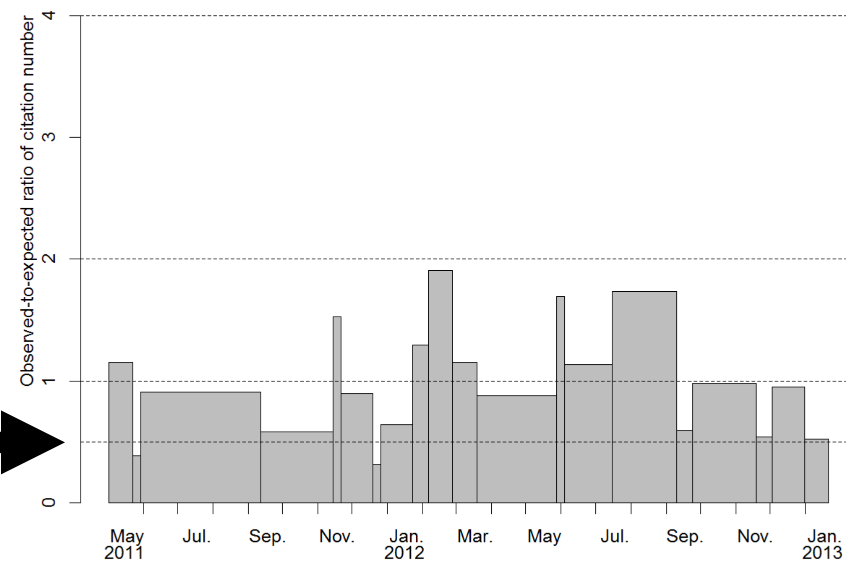
- Assumption: $\mathcal{P}$ are connected parts of a graph
- Result: NP-complete [Becker *et al.*, 1998]

Ordered SPP

- Assumption: $\mathcal{P}$ are intervals
- Result: $\mathcal{O}(n^2)$ dynamic programming

  [Anily *et al.*, 1991] [Jackson *et al.*, 2005]





Applications

Special Versions

Multilevel Geographical Analysis

Time Series Analysis

Coalition Structure Generation

- $\Omega =$ agents
- $\mathcal{P} =$ feasible teams
- $c =$ interaction costs
- $\mathfrak{P} =$ coalition structures

Hierarchical SPP

- Assumption: $\mathcal{P}$ forms a hierarchy
- Result: $\mathcal{O}(n)$ depth-first search

  [Pons *et al.*, 2011] [Lamarche-Perrin *et al.*, 2014]

Graph SPP

- Assumption: $\mathcal{P}$ are connected parts of a graph
- Result: NP-complete [Becker *et al.*, 1998]

Ordered SPP

- Assumption: $\mathcal{P}$ are intervals
- Result: $\mathcal{O}(n^2)$ dynamic programming

  [Anily *et al.*, 1991] [Jackson *et al.*, 2005]

Complete SPP

- Assumption: $\mathcal{P}$ contains all parts
- Result: $\mathcal{O}(3^n)$ dynamic programming

  [Yeh, 1986] [Lehmann *et al.*, 2006]

Applications

Multilevel Geographical Analysis

Time Series Analysis

Coalition Structure Generation

Community Detection

Distributed System Monitoring

Load Balancing Problem

Database Optimization

Image Processing

Combinatorial Auctions

Special Versions

Hierarchical SPP
- Assumption: $\mathcal{P}$ forms a hierarchy
- Result: $\mathcal{O}(n)$ depth-first search
  [Pons *et al.*, 2011] [Lamarche-Perrin *et al.*, 2014]

Graph SPP
- Assumption: $\mathcal{P}$ are connected parts of a graph
- Result: NP-complete [Becker *et al.*, 1998]

Ordered SPP
- Assumption: $\mathcal{P}$ are intervals
- Result: $\mathcal{O}(n^2)$ dynamic programming
  [Anily *et al.*, 1991] [Jackson *et al.*, 2005]

Complete SPP
- Assumption: $\mathcal{P}$ contains all parts
- Result: $\mathcal{O}(3^n)$ dynamic programming
  [Yeh, 1986] [Lehmann *et al.*, 2006]

Ordered x Hierarchical SPP [Dosimont *et al.*, 2014]

Array SPP [Muthukrishnan *et al.*, 2005]

SPP with Size Bounds [Rothkopf *et al.*, 1998]

Cyclic SPP [Rothkopf *et al.*, 1998]

A Lack of Unified Algorithmic Approaches

- The Ordered SPP has been solved at least 6 times in 30 years:

  [Chakravarty *et al.*, 1982] [Anily *et al.*, 1991] [Vidal, 1993] [Rothkopf *et al.*, 1998] [Jackson *et al.*, 2005] [Lamarche-Perrin *et al.*, 2013]

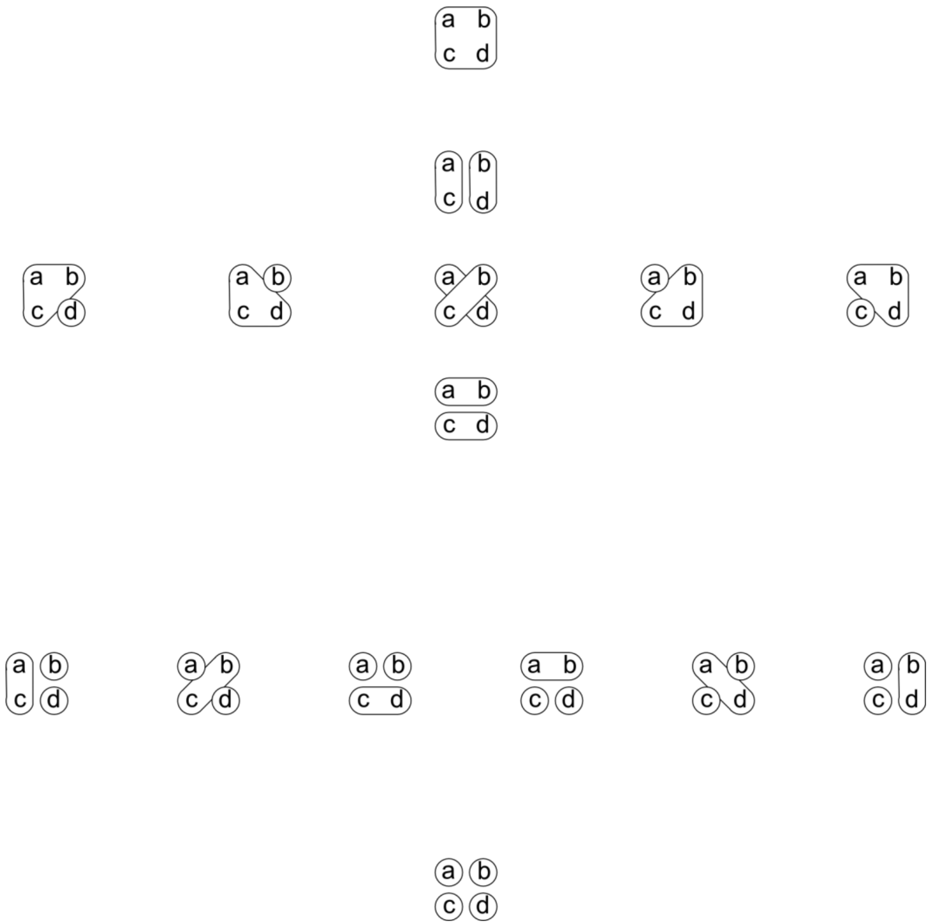
- Characterization of tractability based on general algebraic properties
  - Unimodularity of the integer matrix [Minoux, 1987]
  - Perfection of the intersection graph [Müller, 2006]

  → **Too general, and thus too weak in practice!**

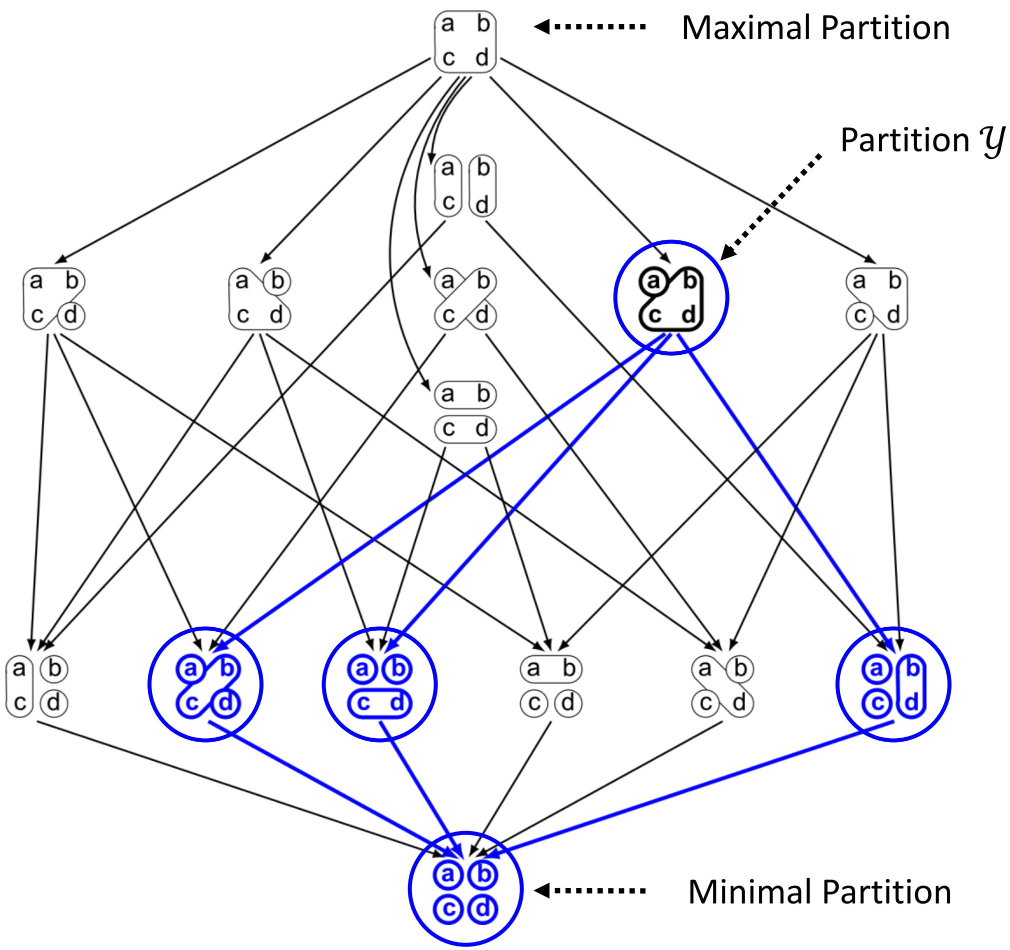
- Our contribution: a unified algorithmic framework

1. A proper understanding of the algebraic structure of the SPP
2. A generic algorithm exploiting this algebraic structure
3. Specialized implementations for versions of the SPP

The Poset of Partitions



The Poset of Partitions



Algebraic Structure

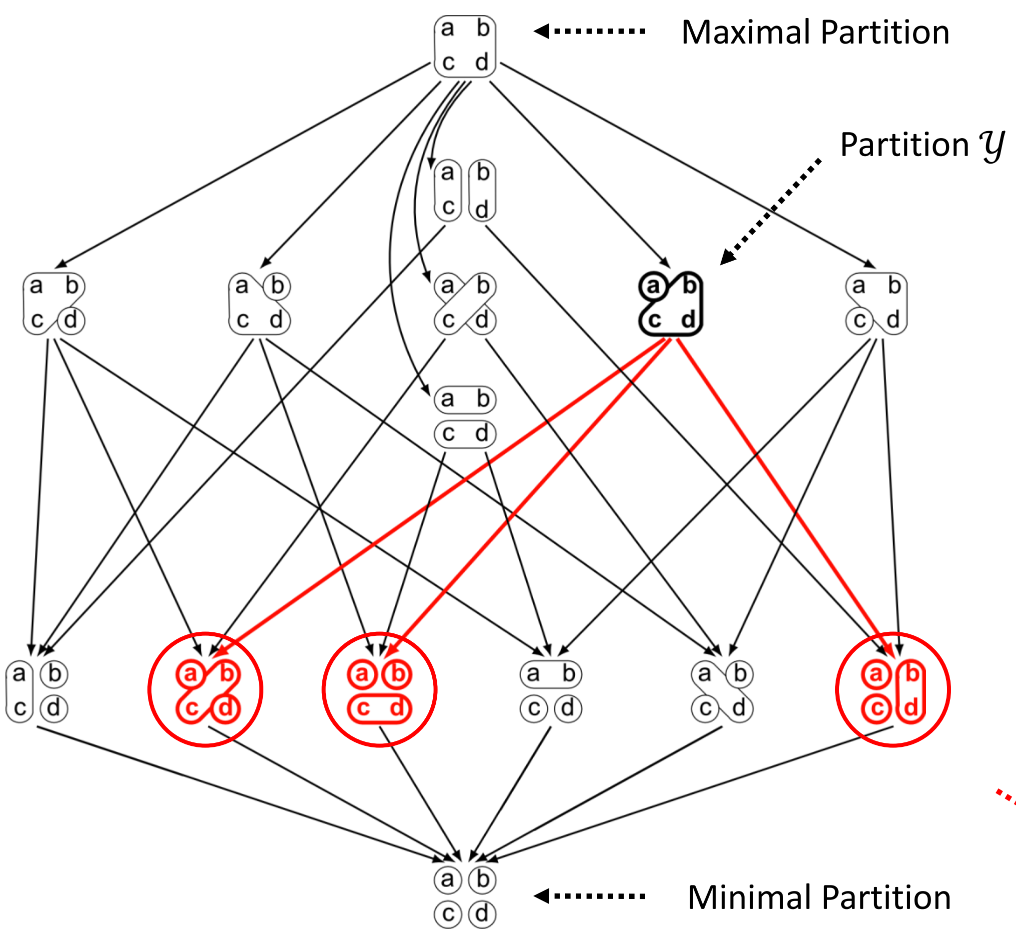
The *refinement relation* defines a partial order on the set of partitions:

$\mathcal{X}$ refines $\mathcal{Y}$

$$\Leftrightarrow \quad \forall X \in \mathcal{X},\ \exists Y \in \mathcal{Y},\ X \subset Y$$

$$\Re(\mathcal{Y}) = \{\mathcal{X} \text{ refining } \mathcal{Y}\}$$

The Poset of Partitions



Algebraic Structure

The *refinement relation* defines a partial order on the set of partitions:

$\mathcal{X}$ refines $\mathcal{Y}$

$$\Leftrightarrow \quad \forall X \in \mathcal{X},\ \exists Y \in \mathcal{Y},\ X \subset Y$$

The *covering relation* is the transitive reduction of the refinement relation:

$\mathcal{X}$ is covered by $\mathcal{Y}$

$\Leftrightarrow$ $\mathcal{X}$ is a "direct" refinement of $\mathcal{Y}$

$$\mathfrak{R}(\mathcal{Y}) = \{\mathcal{X} \text{ refining } \mathcal{Y}\}$$

$$\mathfrak{C}(\mathcal{Y}) = \{\mathcal{X} \text{ covering } \mathcal{Y}\}$$

Branching the Search Space

For any part $X \subset \Omega$, the partitions of X are either the maximal partition $\{X\}$ or a partition that refines a partition covered by $\{X\}$:

$$\mathfrak{P}(X) = \{\{X\}\} \cup \left(\bigcup_{\mathcal{Y} \in \mathfrak{C}(\{X\})} \mathfrak{R}(\mathcal{Y}) \right)$$

Partitions of X

Maximal partition

Partitions refining a partition covered by the maximal partition

$\{X\}$ (1 2 3 4 5)

Branching the Search Space

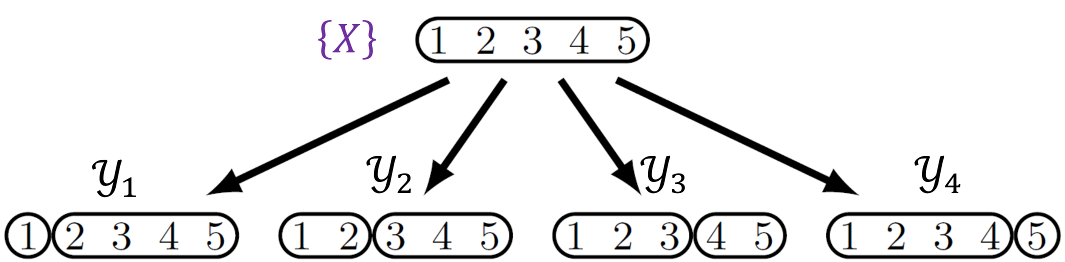
For any part $X \subset \Omega$, the partitions of X are either the maximal partition $\{X\}$ or a partition that refines a partition covered by $\{X\}$:

$$\mathfrak{P}(X) = \{\{X\}\} \cup \left(\bigcup_{\mathcal{Y}\in\mathfrak{C}(\{X\})} \mathfrak{R}(\mathcal{Y}) \right)$$

Partitions of X

Maximal partition

Partitions refining a partition covered by the maximal partition



Branching the Search Space

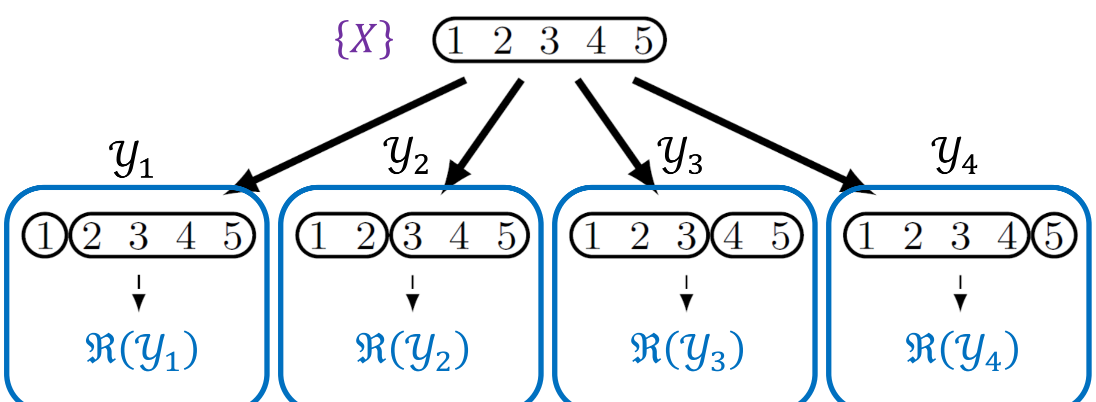
For any part $X \subset \Omega$, the partitions of X are either the maximal partition $\{X\}$ or a partition that refines a partition covered by $\{X\}$:

$$\mathfrak{P}(X) = \{\{X\}\} \cup \left(\bigcup_{\mathcal{Y} \in \mathfrak{C}(\{X\})} \mathfrak{R}(\mathcal{Y}) \right)$$

Partitions of X

Maximal partition

Partitions refining a partition covered by the maximal partition



Branching the Search Space

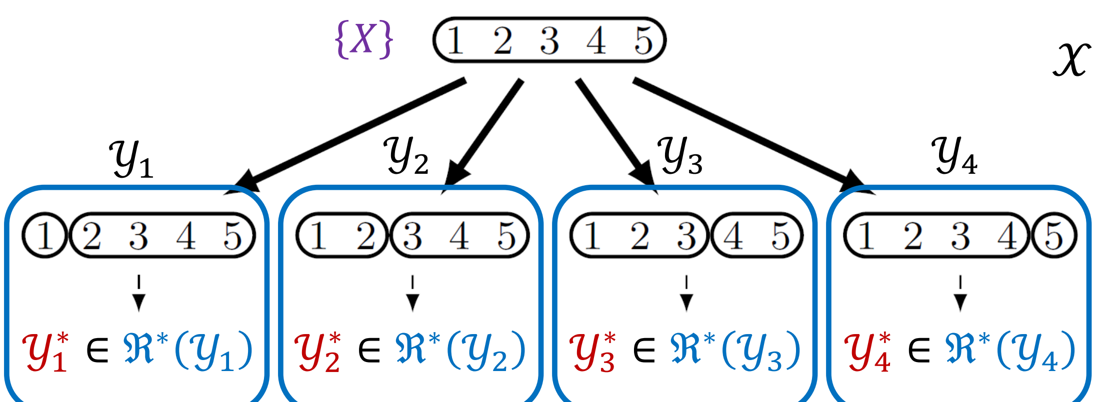
For any part $X \subset \Omega$, the partitions of X are either the maximal partition $\{X\}$ or a partition that refines a partition covered by $\{X\}$:

$$\mathfrak{P}(X) = \{\{X\}\} \cup \left(\bigcup_{\mathcal{Y}\in\mathfrak{C}(\{X\})} \mathfrak{R}(\mathcal{Y}) \right)$$

Partitions of X

Maximal partition

Partitions refining a partition covered by the maximal partition



$$\mathcal{X}^* = \operatorname*{arg\,min}_{\mathcal{X}\in\{\{X\},\mathcal{Y}_1^*,\mathcal{Y}_2^*,\mathcal{Y}_3^*,\mathcal{Y}_4^*\}} \left(\sum_{X\in\mathcal{X}} c(X)\right)$$

New search space

Principle of Optimality

For any partition $\mathcal{Y}$ of Ω, the union of optimal partitions of the parts of $\mathcal{Y}$ is optimal among the refinements of $\mathcal{Y}$:

$$\forall Y \in \mathcal{Y}, \quad \mathcal{Y}_Y^* \in \mathfrak{P}^*(Y) \qquad \Rightarrow \qquad \left(\bigcup_{Y \in \mathcal{Y}} \mathcal{Y}_Y^* \right) \in \mathfrak{R}^*(\mathcal{Y})$$

Locally-optimal partitions of the parts of $\mathcal{Y}$

Optimal partition among the refinements of $\mathcal{Y}$

$\mathcal{Y}$ $\quad (1 \ 2)(3 \ 4 \ 5)$

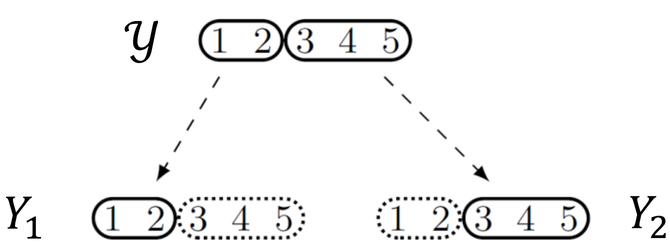
Principle of Optimality

For any partition $\mathcal{Y}$ of Ω, the union of optimal partitions of the parts of $\mathcal{Y}$ is optimal among the refinements of $\mathcal{Y}$:

$$\forall Y \in \mathcal{Y}, \quad \mathcal{Y}_Y^* \in \mathfrak{P}^*(Y) \qquad \Rightarrow \qquad \left(\bigcup_{Y \in \mathcal{Y}} \mathcal{Y}_Y^* \right) \in \mathfrak{R}^*(\mathcal{Y})$$

Locally-optimal partitions of the parts of $\mathcal{Y}$

Optimal partition among the refinements of $\mathcal{Y}$



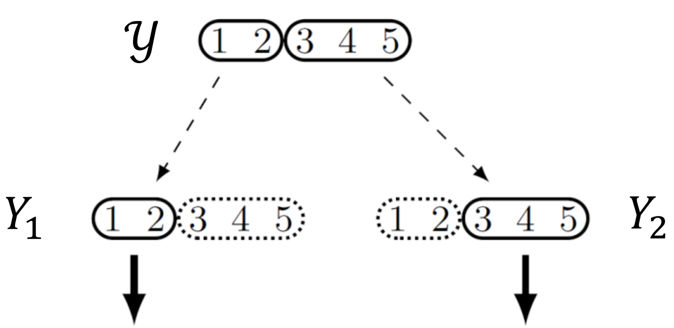
Principle of Optimality

For any partition $\mathcal{Y}$ of Ω, the union of optimal partitions of the parts of $\mathcal{Y}$ is optimal among the refinements of $\mathcal{Y}$:

$$\forall Y \in \mathcal{Y}, \quad \mathcal{Y}_Y^* \in \mathfrak{P}^*(Y) \qquad \Rightarrow \qquad \left(\bigcup_{Y \in \mathcal{Y}} \mathcal{Y}_Y^* \right) \in \mathfrak{R}^*(\mathcal{Y})$$

Locally-optimal partitions of the parts of $\mathcal{Y}$

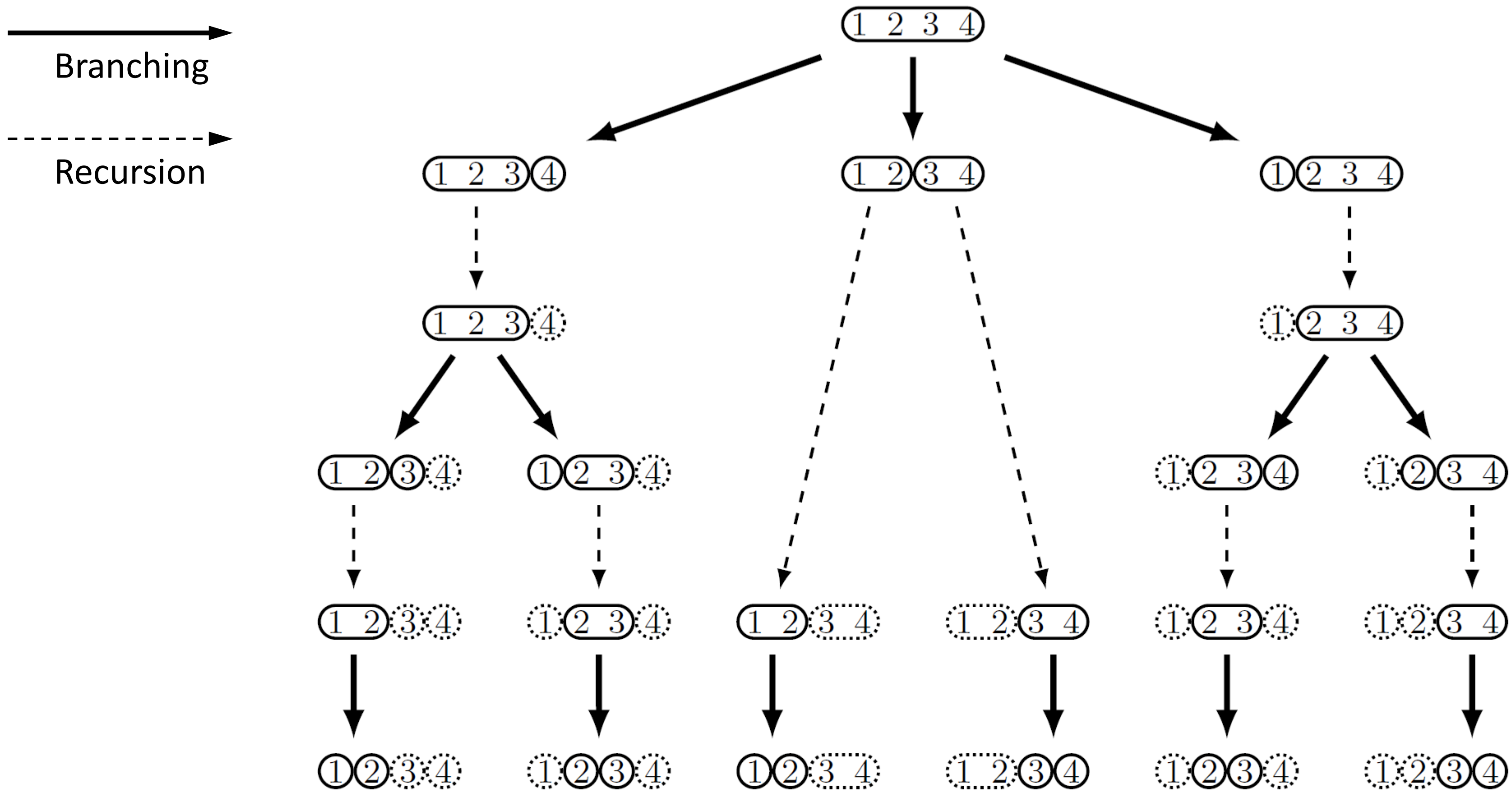
Optimal partition among the refinements of $\mathcal{Y}$



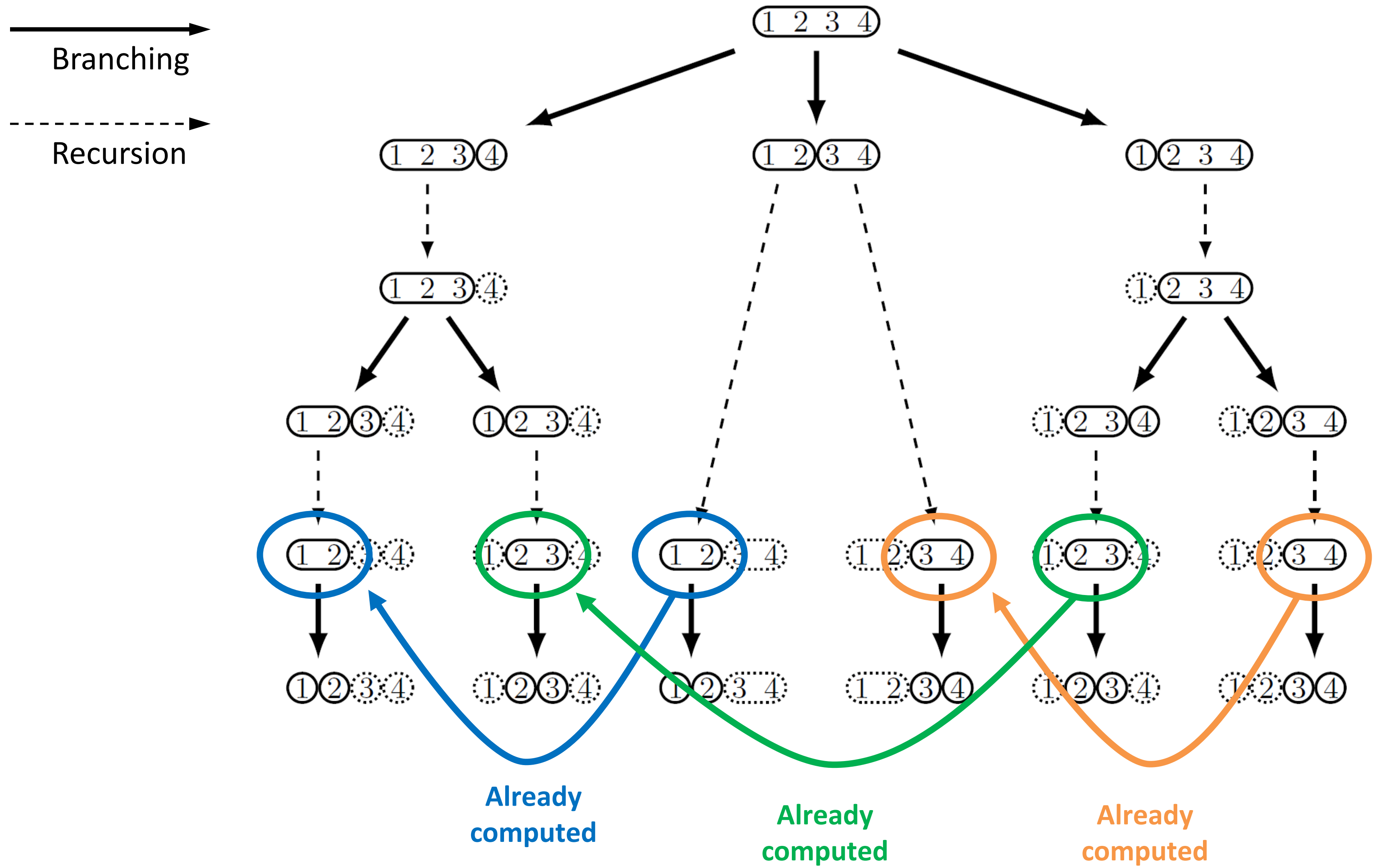
$$\mathcal{Y}_{Y_1}^* \in \mathfrak{P}^*(Y_1) \qquad \mathcal{Y}_{Y_2}^* \in \mathfrak{P}^*(Y_2) \qquad \Rightarrow \qquad \mathcal{Y}_{Y_1}^* \cup \mathcal{Y}_{Y_2}^* \in \mathfrak{R}^*(\mathcal{Y})$$

Execution of the Generic Algorithm

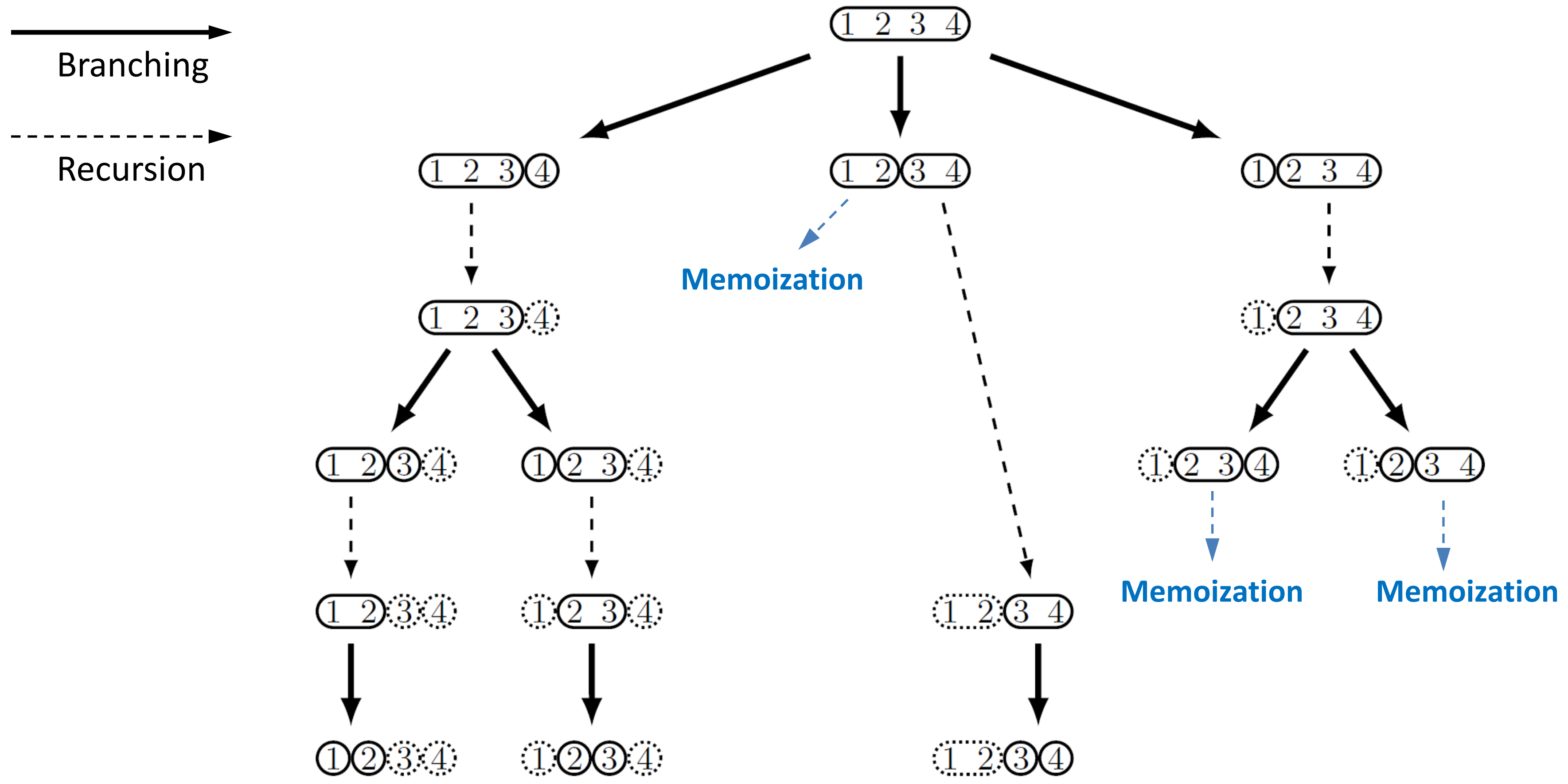
Ordered SPP on a Population of Size 4



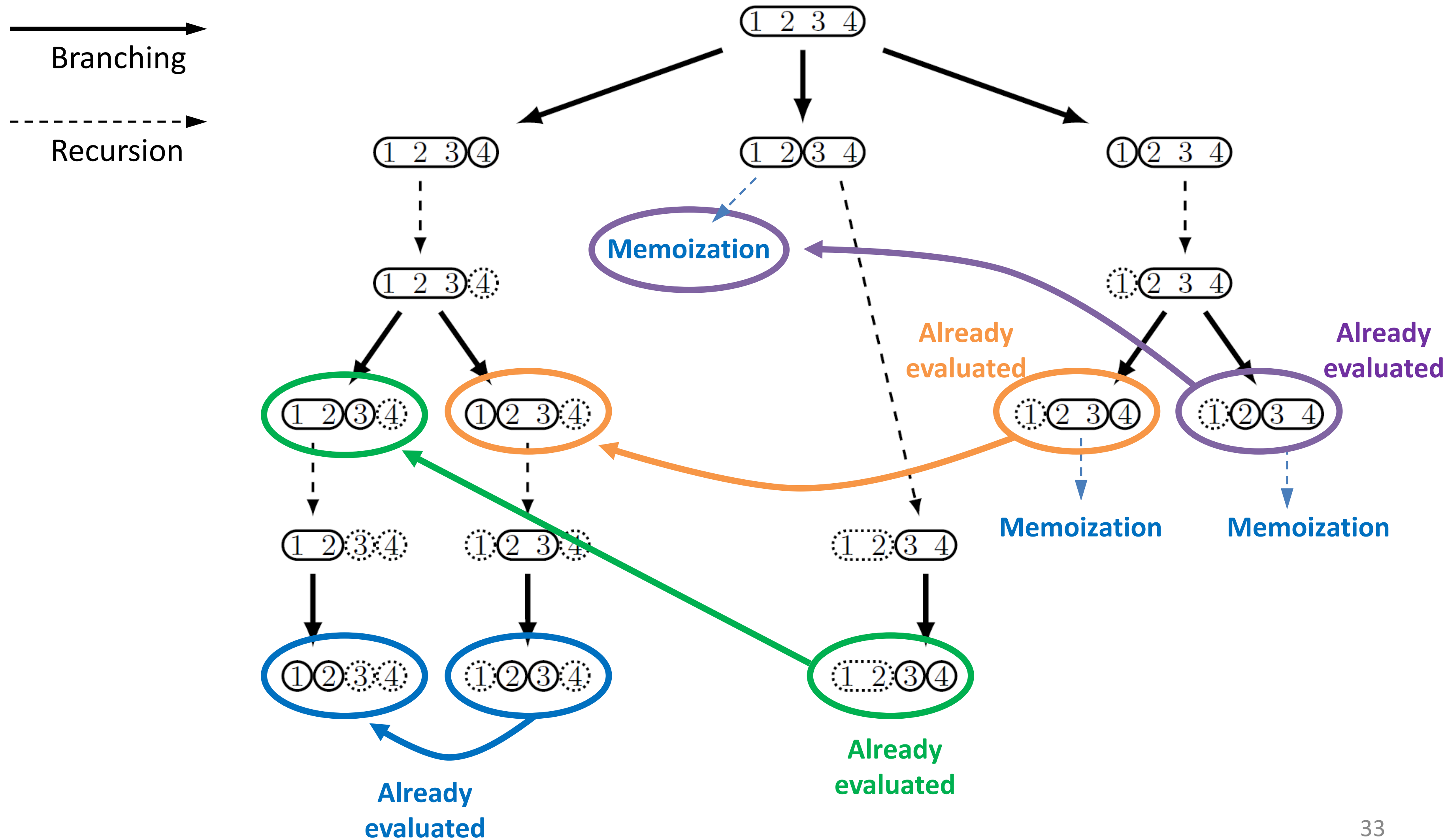
Memoization



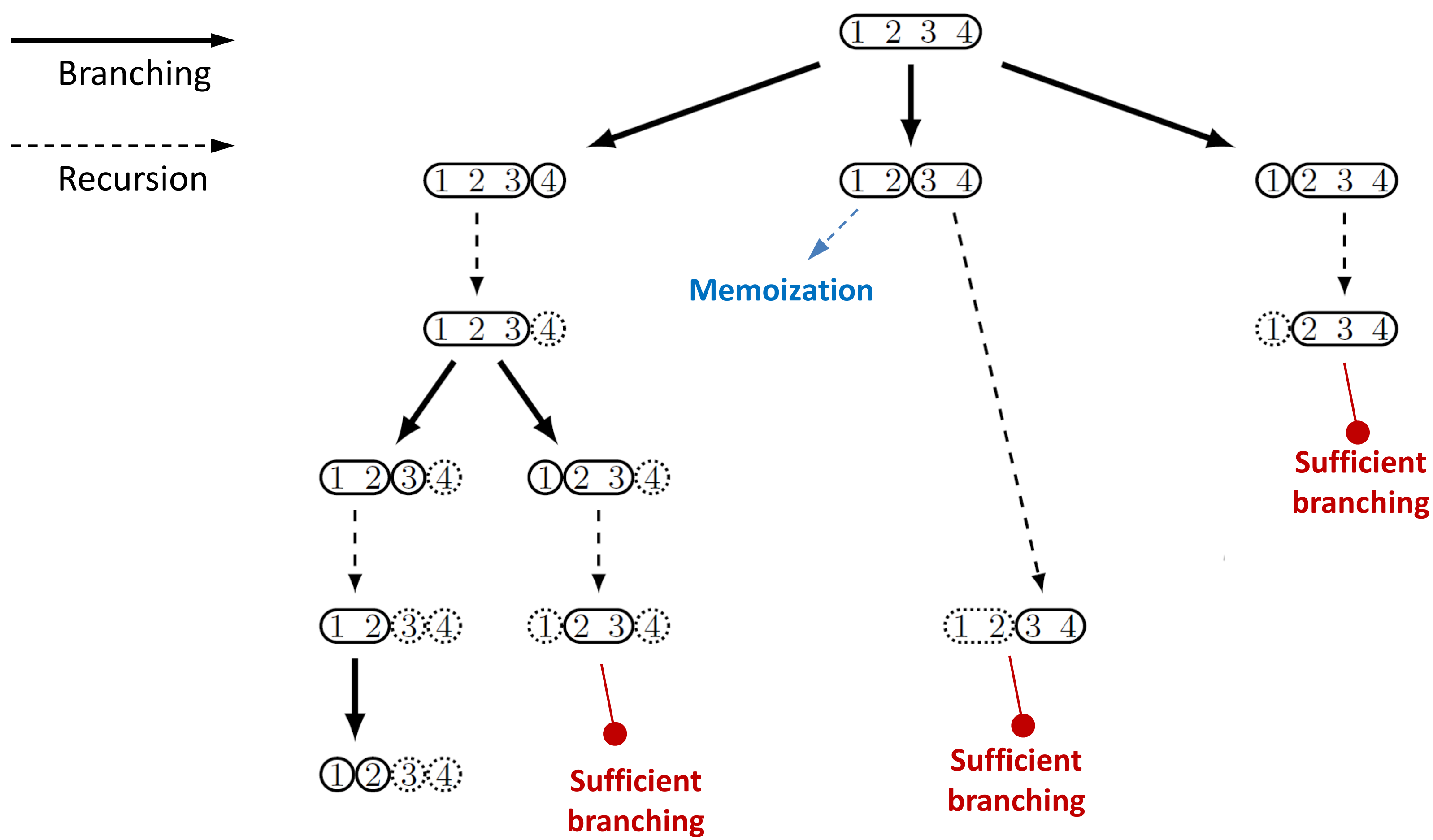
Memoization



Non-redundant Branching



Non-redundant Branching



The Generic Algorithm

A Generic Algorithm to Solve the SPP

Global Inputs:

c a cost function;

$\mathcal{P}$ a set of admissible parts defining admissible partitions;

$\mathfrak{L}$ a set of locally-optimal admissible partitions of parts on which the algorithm has already been applied.

Local Inputs:

X an admissible part;

$\overline{\mathcal{X}}$ the complementary partition of X inherited from the "higher" call ($\overline{\mathcal{X}}$ is a partition of $\Omega \backslash X$);

$\mathfrak{D}$ the set of admissible partitions which refinements have already been evaluated during "higher" calls.

Output:

$\mathcal{X}^*$ a locally-optimal admissible partition of X.

- If the algorithm has already been applied to part X, return the locally-optimal partition recorded in $\mathfrak{L}$.
- Initialization: $\mathcal{X}^* \leftarrow \{\{X\}\}$ and $\mathfrak{D}' \leftarrow \mathfrak{D}$.
- For each $\mathcal{Y} \in \mathfrak{C}(\{X\})$ such that $\overline{\mathcal{X}} \cup \mathcal{Y}$ does not refine any partition in $\mathfrak{D}$, do the following:
  - For each part $Y \in \mathcal{Y}$, call the algorithm with local inputs $X \leftarrow Y$, $\overline{\mathcal{X}} \leftarrow \overline{\mathcal{X}} \cup \mathcal{Y} \backslash \{Y\}$, and $\mathfrak{D} \leftarrow \mathfrak{D}'$ to compute a locally-optimal partition $\mathcal{Y}_Y^* \in \mathfrak{P}^*(Y)$.
  - $\mathcal{Y}^* \leftarrow \bigcup_{Y \in \mathcal{Y}} \mathcal{Y}_Y^*$.
  - If $c(\mathcal{Y}^*) > c(\mathcal{X}^*)$, then $\mathcal{X}^* \leftarrow \mathcal{Y}^*$.
  - $\mathfrak{D}' \leftarrow \mathfrak{D}' \cup \{\mathcal{Y}\}$.
- Return $\mathcal{X}^*$ and record this result in $\mathfrak{L}$.

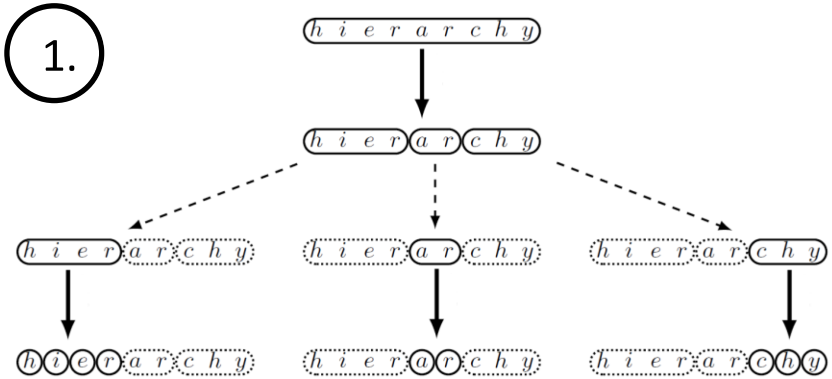
Generic: solve any instance of the SPP
→ but inefficient for special versions

Designing dedicated implementations:

1. Analysing the generic execution
2. Building appropriate data structures
3. Deriving a specialized algorithm

Application to the Hierarchical SPP

1.



2.

Data Structure

- Set of parts: rooted tree
- Optimal partition: cut of the tree
- Algorithm: depth-first search

3.

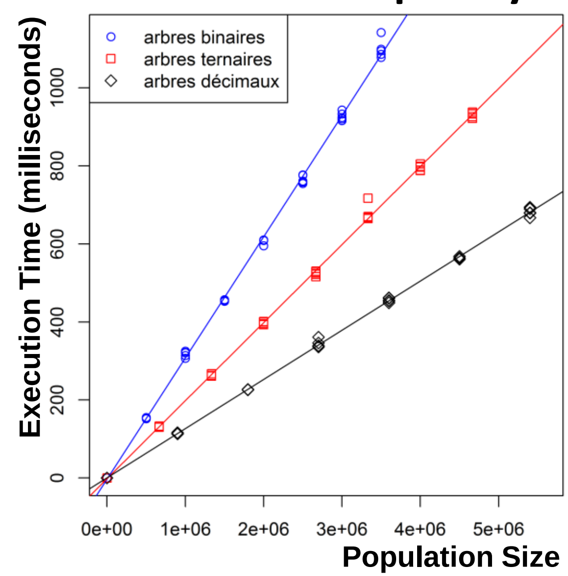
Algorithm 1 for the HSPP

Require: A tree with a label $cost$ on each node representing the cost of the corresponding admissible part.

Ensure: Each node of the tree has a Boolean label $optimalCut$ representing an optimal partition (see above).

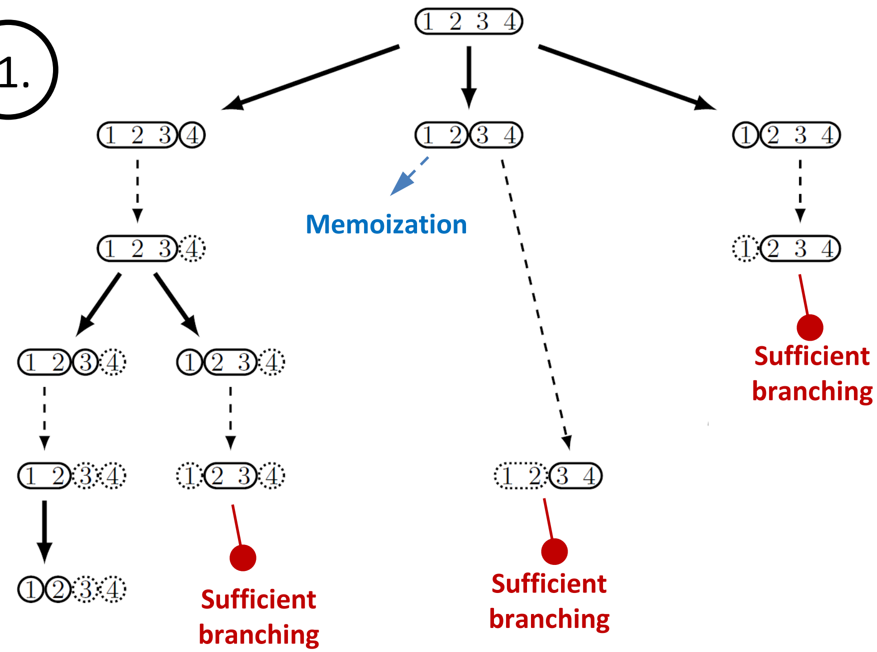
procedure SOLVEHSPP($node$)
  if $node$ has no child **then**
    $node.optimalCost \leftarrow node.cost$
    $node.optimalCut \leftarrow true$
  else
    $MCost \leftarrow node.cost$
    $\mu Cost \leftarrow 0$
    for each $child$ of $node$ **do**
      SOLVEHSPP($child$)
      $\mu Cost \leftarrow \mu Cost + child.optimalCost$
    $node.optimalCost \leftarrow \max(\mu Cost, MCost)$
    $node.optimalCut \leftarrow (\mu Cost < MCost)$

Linear Complexity



Application to the Ordered SPP

1.

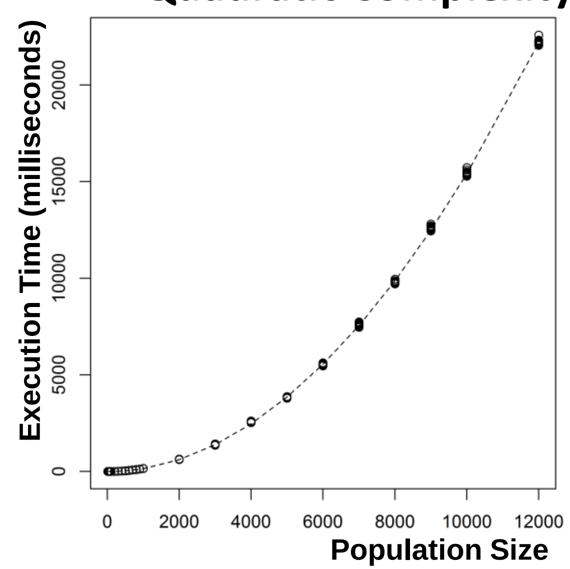


2.

Data Structure

- Set of parts: triangular matrix
- Optimal partition: array of cuts
- Algorithm: dynamic programming

Quadratic Complexity



3.

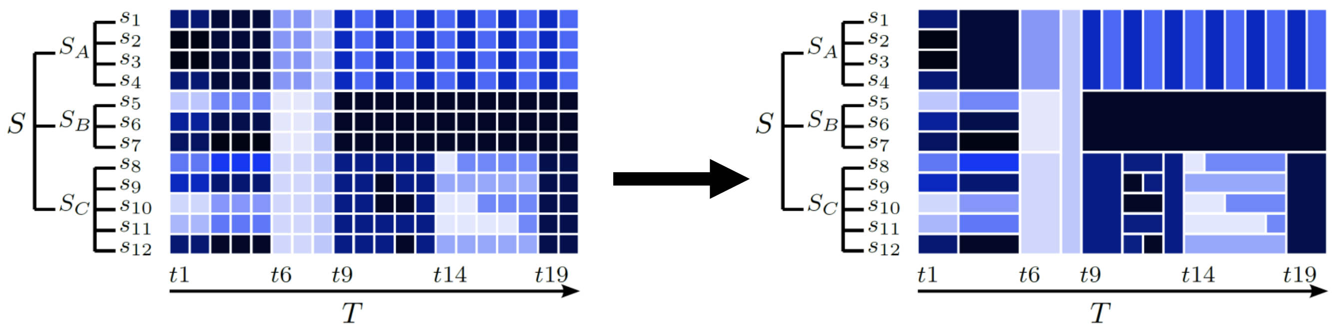
Algorithm 2 for the OSPP

Require: A matrix *cost* recording the costs of intervals.
Ensure: The vector $optimalCut$ represents an optimal partition (see text above).

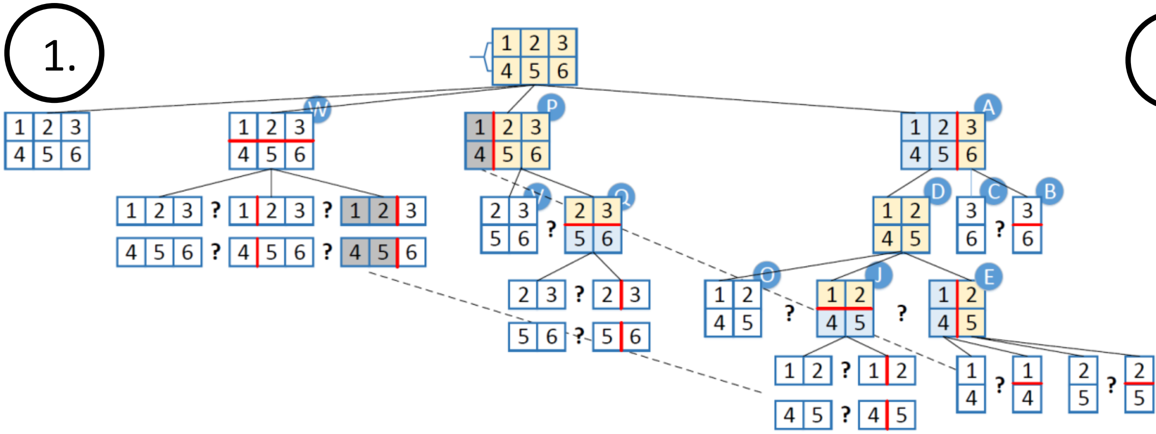
for $j \in [\![1, n]\!]$ **do**
  $optimalCost[j] \leftarrow cost[1, j]$
  $optimalCut[j] \leftarrow 1$
  for $cut \in [\![2, j]\!]$ **do**
    $\mu Cost \leftarrow optimalCost[cut - 1] + cost[cut, j]$
    if $\mu Cost > optimalCost[j]$ **then**
      $optimalCost[j] \leftarrow \mu Cost$
      $optimalCut[j] \leftarrow cut$

Application to a Multidimensional SPP

[Dosimont *et al.*, CLUSTER 2014]



1.



3.

Algorithm 1 computes a hierarchy-and-order-consistent partition that maximizes the parametrized information criterion

procedure *node*.COMPUTEOPTIMALPARTITION(p)

- **for each** *child* **do** ▷ Recursion
  - *child*.COMPUTEOPTIMALPARTITION(p)
- **for** $i = |T| - 1, \dots, 0$ **do** ▷ Iteration
  - **for** $j = i, \dots, |T| - 1$ **do**
    - $cut[i, j] \leftarrow j$ ▷ No cut
    - $pIC[i, j] \leftarrow p.gain[i, j] - (1 - p).loss[i, j]$
    - **if** has children **then** ▷ Spatial cut?
      - $pIC_s \leftarrow 0$
      - **for each** *child* **do**
        - $pIC_s \leftarrow pIC_s + child.pIC[i, j]$
      - **if** $pIC_s > pIC[i, j]$ **then**
        - $cut[i, j] \leftarrow -1$
        - $pIC[i, j] \leftarrow pIC_s$
    - **for** $cut_t = i, \dots, j - 1$ **do** ▷ Temporal cut?
      - $pIC_t \leftarrow pIC[i, cut] + pIC[cut + 1, j]$
      - **if** $pIC_t > pIC[i, j]$ **then**
        - $cut[i, j] \leftarrow cut_t$
        - $pIC[i, j] \leftarrow pIC_t$

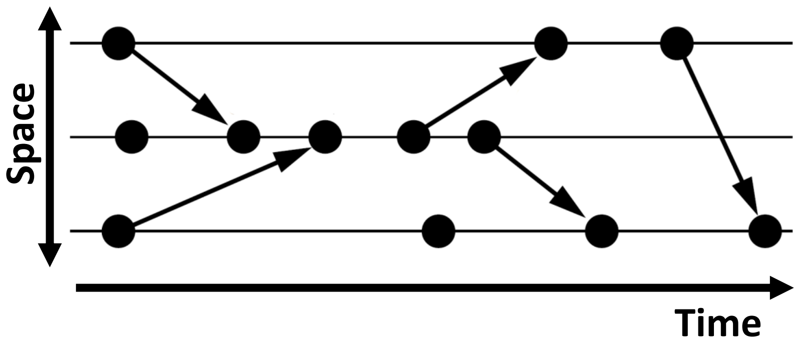
2.

Data Structure

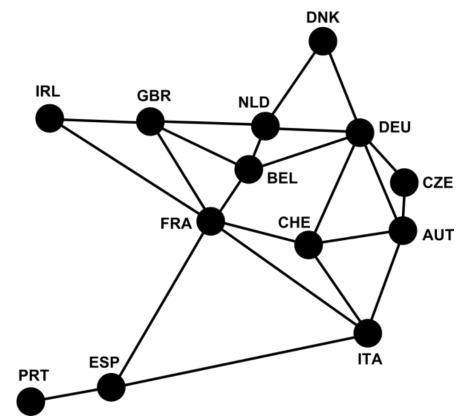
- Set of parts: rooted tree of triangular matrices
- Optimal partition: cut of the tree and arrays of cuts
- Algorithm: depth-first search and dynamic programming

Application Perspectives

Partitioning of Interaction Diagrams [Mattern, 1989]



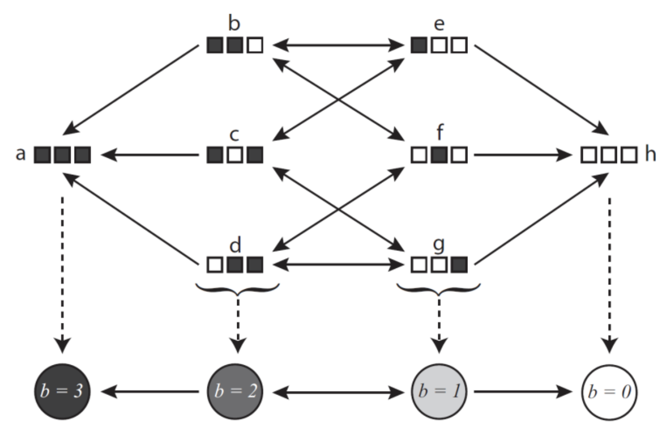
Partitioning of Graphs



Partitioning of Interaction Matrices

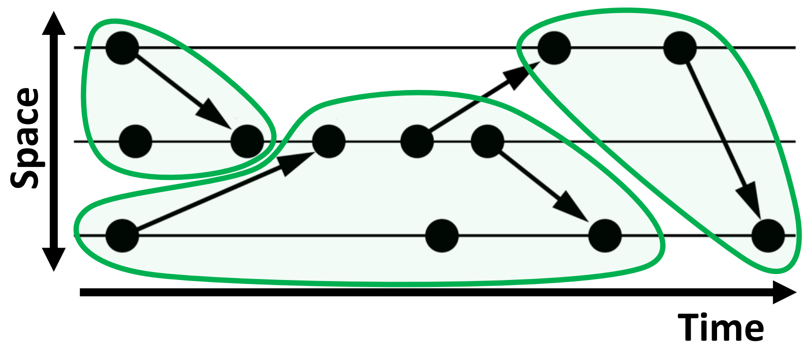
	ESP	FRA	GBR	BEL	CHE
ESP	X	12	11	10	4
FRA	14	X	12	12	5
GBR	20	11	X	6	9
BEL	15	9	6	X	5
CHE	10	16	17	9	X

Partitioning the State Space of Dynamical Systems [Banisch *et al.*, 2013]

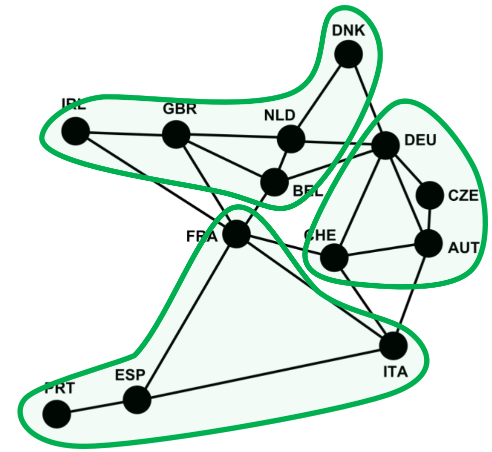


Application Perspectives

Partitioning of Interaction Diagrams [Mattern, 1989]



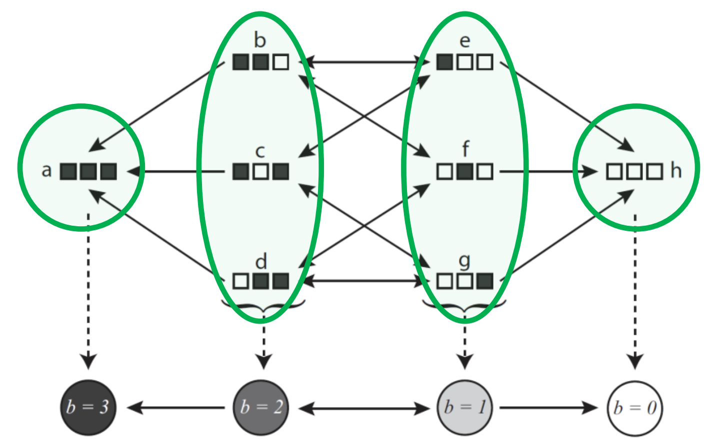
Partitioning of Graphs



Partitioning of Interaction Matrices

	ESP	FRA	GBR	BEL	CHE
ESP	X	12	11	10	4
FRA	14	X	12	12	5
GBR	20	11	X	6	9
BEL	15	9	6	X	5
CHE	10	16	17	9	X

Partitioning the State Space of Dynamical Systems [Banisch *et al.*, 2013]



THANK YOU FOR YOUR ATTENTION

Email: Robin.Lamarche-Perrin@mis.mpg.de
Page: www.mis.mpg.de/jjost/members/robin-lamarche-perrin.html