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

Robin Lamarche-Perrin¹, Yves Demazeau², and Jean-Marc Vincent²

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]
The WUTS Hierarchy [Grasland and Didelon, 2007]

Preserving Social and Political Features
The Set Partitioning Problem

Given:
- a set of individuals $\Omega = \{x_1, \ldots, x_n\}$
The Set Partitioning Problem

Given:
- a set of individuals \( \Omega = \{x_1, \ldots, x_n\} \)
- a set of admissible parts \( \mathcal{P} = \{X_1, \ldots, X_m\} \subset 2^\Omega \)
The Set Partitioning Problem

Given:
- a set of individuals $\Omega = \{x_1, ..., x_n\}$
- a set of admissible parts $P = \{X_1, ..., X_m\} \subset 2^\Omega$
- a cost function $c : P \rightarrow \mathbb{R}$
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 $\mathfrak{P} = \{\mathcal{X} \subset \mathcal{P} \text{ such that } \mathcal{X} \text{ is a partition of } \Omega\}$
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 $\mathfrak{P} = \{\mathcal{X} \subset \mathcal{P} \text{ such that } \mathcal{X} \text{ is a partition of } \Omega\}$

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

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

$\Rightarrow$ NP-complete!
The Set Partitioning Problem

Given:
- a set of individuals \( \Omega = \{x_1, ..., x_n\} \)
- a set of admissible parts \( \mathcal{P} = \{X_1, ..., X_m\} \subset 2^\Omega \)
- a cost function \( c : \mathcal{P} \rightarrow \mathbb{R} \)
- the corresponding set of admissible partitions \( \mathfrak{P} = \{\mathcal{X} \in \mathcal{P} \text{ such that } \mathcal{X} \text{ is a partition of } \Omega\} \)

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

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

\( \rightarrow \) NP-complete!
Applications

Multilevel Geographical Analysis
- $\Omega = \text{territorial units}$
- $\mathcal{P} = \text{admissible aggregates}$
- $c = \text{compression rate}$
- $\Psi = \text{aggregated representations}$

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

Multilevel Geographical Analysis

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

Special Versions

Hierarchical SPP
- Assumption: $\mathcal{P}$ forms a hierarchy
- Result: $O(n)$ 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: $O(n^2)$ dynamic programming [Anily et al., 1991] [Jackson et al., 2005]
Coalition Structure Generation
- \( \Omega = \) agents
- \( \mathcal{P} = \) feasible teams
- \( c = \) interaction costs
- \( \mathfrak{P} = \) coalition structures

Time Series Analysis

Multilevel Geographical Analysis

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

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

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

Complete SPP
- Assumption: \( \mathcal{P} \) contains all parts
- Result: \( O(3^n) \) dynamic programming
  \([\text{Yeh, 1986}] [\text{Lehmann et al., 2006}]\)
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: $\mathcal{O}(n^2)$ dynamic programming

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

- 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

\[
\begin{array}{cccc}
\text{a} & \text{b} \\
\text{c} & \text{d} \\
\text{a} & \text{b} \\
\text{c} & \text{d} \\
\text{a} & \text{b} \\
\text{c} & \text{d} \\
\text{a} & \text{b} \\
\text{c} & \text{d} \\
\text{a} & \text{b} \\
\text{c} & \text{d} \\
\end{array}
\]
The Poset of Partitions

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

\[ \mathcal{X} \text{ refines } \mathcal{Y} \iff \forall X \in \mathcal{X}, \exists Y \in \mathcal{Y}, X \subset Y \]

\( \mathcal{R}(\mathcal{Y}) = \{X \text{ refining } \mathcal{Y}\} \)
The Poset of Partitions

**Algebraic Structure**

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

\[ \mathcal{X} \text{ refines } \mathcal{Y} \iff \forall X \in \mathcal{X}, \exists Y \in \mathcal{Y}, X \subset Y \]

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

\[ \mathcal{X} \text{ is covered by } \mathcal{Y} \iff \mathcal{X} \text{ is a “direct” refinement of } \mathcal{Y} \]

\[ \mathcal{R}(Y) = \{ \mathcal{X} \text{ refining } Y \} \]

\[ \mathcal{C}(Y) = \{ \mathcal{X} \text{ covering } Y \} \]
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 $\{X\}$:

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

- **Partitions of $X$**
- **Maximal partition**
- **Partitions refining a partition covered by the maximal partition**

\[
\{X\} \quad 1 \ 2 \ 3 \ 4 \ 5
\]
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 $\{X\}$:

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

- Partitions of $X$
- Maximal partition
- Partitions refining a partition covered by the maximal partition
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 $\{X\}$:

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

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

Partitions of \( X \)  
Maximal partition  
Partitions refining a partition covered by the maximal partition

\[
x^* = \arg\min_{x \in \{X, y_1^*, y_2^*, y_3^*, y_4^*\}} \left( \sum_{x \in x} c(X) \right)
\]

New search space
Principle of Optimality

For any partition \( \mathcal{Y} \) of \( \Omega \), the union of optimal partitions of the parts of \( \mathcal{Y} \) is optimal among the refinements of \( \mathcal{Y} \):

\[
\forall Y \in \mathcal{Y}, \quad y_Y^* \in \mathfrak{P}^*(Y) \quad \Rightarrow \quad \left( \bigcup_{Y \in \mathcal{Y}} y_Y^* \right) \in \mathcal{R}^*(\mathcal{Y})
\]

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

Optimal partition among the refinements of \( \mathcal{Y} \)

\( \mathcal{Y} \)
Principle of Optimality

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

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

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 $\Omega$, the union of optimal partitions of the parts of $\mathcal{Y}$ is optimal among the refinements of $\mathcal{Y}$:

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

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

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

$\mathcal{Y}$ \quad 1 \hspace{0.1cm} 2 \hspace{0.1cm} 3 \hspace{0.1cm} 4 \hspace{0.1cm} 5$

$Y_1$ \quad \underline{1 \hspace{0.1cm} 2 \hspace{0.1cm} 3 \hspace{0.1cm} 4 \hspace{0.1cm} 5}$ \quad $Y_2$

$y_{Y_1}^* \in \mathcal{P}^*(Y_1)$ \quad $y_{Y_2}^* \in \mathcal{P}^*(Y_2)$ \quad $y_{Y_1}^* \cup y_{Y_2}^* \in \mathcal{R}^*(Y)$
Execution of the Generic Algorithm
Ordered SPP on a Population of Size 4

Branching

Recursion
Memoization

Branching

Recursion

Already computed

Already computed

Already computed
Memoization

Branching

Recursion

Memoization

Memoization

Memoization
Non-redundant Branching

Branching

Recursion

Memoization

Already evaluated

Already evaluated
Non-redundant Branching

Branching → Memoization → Sufficient branching

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

- **Local Inputs:**
  - $X$ an admissible part;
  - $\overline{X}$ the complementary partition of $X$ inherited from the “higher” call ($\overline{X}$ is a partition of $\Omega \setminus X$);
  - $\mathcal{D}$ 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 $\mathcal{L}$.
- Initialization: $\mathcal{X}^* \leftarrow \{X\}$ and $\mathcal{D}' \leftarrow \mathcal{D}$.
- For each $\mathcal{Y} \in \mathcal{C}(\{X\})$ such that $\overline{X} \cup \mathcal{Y}$ does not refine any partition in $\mathcal{D}$, do the following:
  - For each part $Y \in \mathcal{Y}$, call the algorithm with local inputs $X \leftarrow Y$, $\overline{X} \leftarrow \overline{X} \cup \mathcal{Y}\setminus\{Y\}$, and $\mathcal{D} \leftarrow \mathcal{D}'$ to compute a locally-optimal partition $\mathcal{Y}_Y^* \in \mathcal{P}^*(Y)$.
  - $\mathcal{Y}^* \leftarrow \bigcup_{Y \in \mathcal{Y}} \mathcal{Y}_Y^*$.
  - If $c(\mathcal{Y}^*) > c(\mathcal{X}^*)$, then $\mathcal{X}^* \leftarrow \mathcal{Y}^*$.
  - $\mathcal{D}' \leftarrow \mathcal{D}' \cup \{\mathcal{Y}\}$.
- Return $\mathcal{X}^*$ and record this result in $\mathcal{L}$.

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. Algorithm 1 for the HSPP

   **Require**: A tree with a label \( \text{cost} \) on each node representing the cost of the corresponding admissible part.

   **Ensure**: Each node of the tree has a Boolean label \( \text{optimalCut} \) representing an optimal partition (see above).

   **procedure** `SOLVEHSPPP(node)`
   - if node has no child then
     - \( \text{node.optimalCost} \leftarrow \text{node.cost} \)
     - \( \text{node.optimalCut} \leftarrow \text{true} \)
   - else
     - \( M \text{Cost} \leftarrow \text{node.cost} \)
     - \( \mu \text{Cost} \leftarrow 0 \)
     - for each child of node do
       - `SOLVEHSPPP(child)`
       - \( \mu \text{Cost} \leftarrow \mu \text{Cost} + \text{child.optimalCost} \)
     - \( \text{node.optimalCost} \leftarrow \max(\mu \text{Cost}, M \text{Cost}) \)
     - \( \text{node.optimalCut} \leftarrow (\mu \text{Cost} < M \text{Cost}) \)

2. Data Structure
   - Set of parts: rooted tree
   - Optimal partition: cut of the tree
   - Algorithm: depth-first search

3. Linear Complexity

   ![Graph showing execution time (milliseconds) vs population size](image)
Application to the Ordered SPP

1. 

2. Data Structure
   - Set of parts: triangular matrix
   - Optimal partition: array of cuts
   - Algorithm: dynamic programming

3. Algorithm 2 for the OSPP
   \[
   \textbf{Require:} \quad \text{A matrix } \text{cost} \text{ recording the costs of intervals.}
   \]
   \[
   \textbf{Ensure:} \quad \text{The vector } \text{optimalCut} \text{ represents an optimal partition (see text above).}
   \]
   \[
   \begin{align*}
   \text{for } & j \in [1,n] \text{ do} \\
   \text{optimalCost}[j] & \leftarrow \text{cost}[1,j] \\
   \text{optimalCut}[j] & \leftarrow 1 \\
   \text{for } & \text{cut } \in [2,j] \text{ do} \\
   \mu\text{Cost} & \leftarrow \text{optimalCost}[\text{cut} - 1] + \text{cost}[	ext{cut}, j] \\
   \text{if } & \mu\text{Cost} > \text{optimalCost}[j] \text{ then} \\
   \text{optimalCost}[j] & \leftarrow \mu\text{Cost} \\
   \text{optimalCut}[j] & \leftarrow \text{cut}
   \end{align*}
   \]
Application to a Multidimensional SPP
[Dosimont et al., CLUSTER 2014]

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 Interaction Matrices

<table>
<thead>
<tr>
<th></th>
<th>ESP</th>
<th>FRA</th>
<th>GBR</th>
<th>BEL</th>
<th>CHE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESP</td>
<td>X</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>FRA</td>
<td>14</td>
<td>X</td>
<td>12</td>
<td>12</td>
<td>5</td>
</tr>
<tr>
<td>GBR</td>
<td>20</td>
<td>11</td>
<td>X</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>BEL</td>
<td>15</td>
<td>9</td>
<td>6</td>
<td>X</td>
<td>5</td>
</tr>
<tr>
<td>CHE</td>
<td>10</td>
<td>16</td>
<td>17</td>
<td>9</td>
<td>X</td>
</tr>
</tbody>
</table>

Partitioning of Graphs

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

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