Combinatorial local patterns in Boolean dataset
Plan

1. Introduction

2. Generic algorithm for local pattern mining in 0/1 data
   - Biset mining
   - Enumeration strategy
   - Constraint handling
   - Finding upper bounds

3. Data-Peeler: closed n-sets in n-ary relations
   - Closed $n$-sets
   - Applications
Introduction
Generic algorithm for local pattern mining in 0/1 data
Data-Peeler: closed n-sets in n-ary relations

Combinatorial local pattern extraction

Set pattern extraction

Declaratively defining interesting patterns:

- $\mathcal{L}$ is a language of patterns to be mined.
- $\mathbf{r}$ is a dataset
- $\mathcal{C}$ is a predicate used for selecting potentially interesting patterns.

A combinatorial local pattern extraction under constraints in $\mathbf{r}$ is

$$\text{Patterns} = \{ P \in \mathcal{L} \mid \mathcal{C}(P, \mathbf{r}) \}$$
Combinatorial local pattern extraction

Set pattern extraction

Declaratively defining interesting patterns:
- $\mathcal{L}$ is a language of patterns to be mined.
- $\mathbf{r}$ is a dataset
- $\mathcal{C}$ is a predicate used for selecting potentially interesting patterns.

A combinatorial local pattern extraction under constraints in $\mathbf{r}$ is:

$$\text{Patterns} = \{ P \in \mathcal{L} \mid \mathcal{C}(P, \mathbf{r}) \}$$

Such techniques are useful when
- the dataset $\mathbf{r}$ is large
- there is few knowledge on the studied phenomenon
Combinatorial local pattern extraction

Set pattern extraction

Declaratively defining interesting patterns:
- \( \mathcal{L} \) is a language of patterns to be mined.
- \( r \) is a dataset
- \( C \) is a predicate used for selecting potentially interesting patterns.

A combinatorial local pattern extraction under constraints in \( r \) is
\[
\text{Patterns} = \{ P \in \mathcal{L} \mid C(P, r) \}
\]

Main Principle

Some of the constraints are sufficiently tight to drastically reduce the search space and turn the computation to be feasible.
**Examples**

### 0/1 dataset

\[ \mathcal{L} = 2^\mathcal{I} \]

\[ \mathbf{r} = \{ \{a, b, d\}, \{b, c, d\}, \{a, d\}, \{b\} \} \]

\( \mathcal{C} : \text{Patterns of } \mathcal{L} \text{ that appear at least twice in } \mathbf{r} \text{ and of size at least 2.} \)

\( \mathcal{P}atterns = \{ \{a, d\}, \{b, d\} \} \)

### String dataset

\( \mathcal{L} \) is the set of finite ordered list of \( \mathcal{I} \).

\[ \mathbf{r} = \{cabbac, bba, ccabd, cdad\} \]

\( \mathcal{C} : \text{Patterns of } \mathcal{L} \text{ that appear at least twice in } \mathbf{r} \text{ and of size at least 3.} \)

Answer (with strings) : \( \mathcal{P}atterns\{bba, cab\} \)

Answer (with sequences) : \( \mathcal{P}atterns\{bba, cab, cad\} \)
Local pattern extraction and data experts

Complete solvers for local pattern extraction have reached a kind of maturity:

- numerous extractors
- efficient for various kinds of datasets
Local pattern extraction and data experts

Complete solvers for local pattern extraction have reached a kind of maturity:

- numerous extractors
- efficient for various kinds of datasets

⇒ Slow progress from Data Mining to Knowledge Discovery
Local pattern extraction and data experts

Complete solvers for local pattern extraction have reached a kind of maturity:

- numerous extractors
- efficient for various kinds of datasets

⇒ Slow progress from Data Mining to Knowledge Discovery

Still lack of:

- taking into account user’s expectations
  - increase the interestingness of extracted patterns
  - decrease the number of spurious patterns
- abstraction and generalization
  - new patterns, e.g., fault-tolerant and numerical patterns
  - new constraints
  - algorithm tuning for efficiency purpose
Local patterns in 0/1 datasets

<table>
<thead>
<tr>
<th></th>
<th>$i_1$</th>
<th>$i_2$</th>
<th>$i_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$t_2$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$t_3$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$t_4$</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Extraction task

A biset extraction task under-constraints in a 0/1 dataset $r$ can be seen as the computation of the following collection:

$$\mathcal{L} = 2^T \times 2^I$$

$$\mathcal{Patt} = \{ P = (X, Y) \in \mathcal{L} \mid C(P, r) \}$$
Formal concepts

<table>
<thead>
<tr>
<th></th>
<th>$i_1$</th>
<th>$i_2$</th>
<th>$i_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$t_2$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$t_3$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$t_4$</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Formal concept extraction

$$\mathcal{L} = 2^T \times 2^I$$

$$\mathcal{C} \equiv \mathcal{C}_{\text{linked}} \land \mathcal{C}_{\text{closed}}$$

$$\mathcal{C}_{\text{linked}}(T, I) = \forall i \in I \text{ and } \forall t \in T, (t, i) \in r$$

$$\mathcal{C}_{\text{closed}}(T, I) = \forall t \in T \setminus T, \neg\mathcal{C}_{\text{linked}}(T \cup \{t\}, I)$$

$$\forall i \in I \setminus I, \neg\mathcal{C}_{\text{linked}}(T, I \cup \{i\})$$

$$\mathcal{P} = \{({t_2, t_3}, \{i_1, i_2\}), ({t_3}, \{i_1, i_2, i_3\}), ({t_2, t_3, t_4}, \{i_2\}), ({t_1, t_3}, \{i_3\})\}$$
Local patterns in 0/1 datasets

<table>
<thead>
<tr>
<th></th>
<th>$i_1$</th>
<th>$i_2$</th>
<th>$i_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$t_2$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$t_3$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$t_4$</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

A gene (column) is over expressed in a biological situation (row): all the (maximal) groups of genes that are co-over-expressed in different biological situations.
Local patterns in 0/1 datasets

<table>
<thead>
<tr>
<th></th>
<th>$i_1$</th>
<th>$i_2$</th>
<th>$i_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$t_2$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$t_3$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$t_4$</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- A gene (column) is over expressed in a biological situation (row) : all the (maximal) groups of genes that are co-over-expressed in different biological situations.
- A user (row) has appropriate rights to access to a service (column) : all the (maximal) groups of users that have the same rights for some services.
Local patterns in 0/1 datasets

<table>
<thead>
<tr>
<th></th>
<th>i₁</th>
<th>i₂</th>
<th>i₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>t₂</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>t₃</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>t₄</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- A gene (column) is over expressed in a biological situation (row): all the (maximal) groups of genes that are co-over-expressed in different biological situations.
- A user (row) has appropriate rights to access to a service (column): all the (maximal) groups of users that have the same rights for some services.
- A person (row) isn’t interested in a topic (column): all groups of people who have common disinterests.
## Problem setting

### Data-Peeler

<table>
<thead>
<tr>
<th></th>
<th>$i_1$</th>
<th>$i_2$</th>
<th>$i_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$t_2$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$t_3$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$t_4$</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\[(t_1, t_2, i_1 i_2 i_3)\]

\[(t_2 t_3, i_1 i_2 i_3)\]

\[(t_2 t_3, i_1 i_2)\]

\[(t_3, i_1 i_2 i_3)\]

\[(t_1 t_3, i_3)\]

\[(t_1 t_2 t_3, \emptyset)\]

\[(t_1 t_2 t_3, i_1 i_2 i_3)\]

\[(t_1 t_2 t_3, i_2)\]
Generalization Problem

Problem

- a lot of enumeration strategies
- a lot of different ways of handling constraints
- a lot of "ad-hoc" data structures
- the fundamental ideas hidden in the middle of the haystack
Generalization Problem

Problem
- a lot of enumeration strategies
- a lot of different ways of handling constraints
- a lot of "ad-hoc" data structures
- the fundamental ideas hidden in the middle of the haystack

Goal
Abstracting all these principles while being:
- simple for depicting fundamental principals
- able to use the best ideas of the-state-of-the-art algorithms
Generalization Problem

Problem
- a lot of enumeration strategies
- a lot of different ways of handling constraints
- a lot of "ad-hoc" data structures
- the fundamental ideas hidden in the middle of the haystack

The only assumption
Let $SP$ be a search space. It exists $Upper(C)(SP)$ such that
\[
\forall (X, Y) \in SP \; \neg Upper(C)(SP) \Rightarrow \neg C(X, Y)
\]
Pattern domain

A biset is a couple \((X, Y)\) such that \(X \subseteq \mathcal{T}\) and \(Y \subseteq \mathcal{T}\)
Pattern domain

Search space

A biset search space $SP$ is defined by a lattice $\langle (\bot_T, \bot_I), (\top_T, \top_I) \rangle$. 
Pattern domain

Search space

A biset search space $SP$ is defined by a lattice $\langle (\perp_T, \perp_I), (\top_T, \top_I) \rangle$. 
Enumeration

Let $y$ belongs to $\top_I \setminus \bot_I$ or $\top_I \setminus \bot_I$:

$$\text{Enumeration}(SP, y) =$$

$$SP \setminus y = \langle (\bot_I, \bot_I), (\top_I, \top_I) \setminus \{y\} \rangle$$

$$SP \cup y = \langle (\bot_I, \bot_I) \cup \{y\}, (\top_I, \top_I) \rangle$$
Enumeration

Let $y$ belongs to $\top_T \setminus \bot_T$ or $\top_I \setminus \bot_I$:

$\text{Enumeration}(SP, y) =$

$SP \setminus y = \langle (\bot_T, \bot_I), (\top_T, \top_I) \setminus \{y\} \rangle$

$SP \cup y = \langle (\bot_T, \bot_I) \cup \{y\}, (\top_T, \top_I) \rangle$
Constraint handling

Checking

\[ \text{Checking}(SP, C) = \neg \text{Upper}(C)(SP) \]
Constraint handling

**Checking**

\[ \text{Checking}(SP, C) = \neg \text{Upper}(C)(SP) \]

**Example - Checking**

Extraction task: \( C(X, Y) \equiv C_{FC} \land |X| \times |Y| \geq 8. \)

\( \langle (t_3, i_2 i_3), (t_1 t_3 t_4, i_2 i_3) \rangle \) can be pruned.
Constraint handling

Checking

\[ \text{Checking}(\text{SP}, \mathcal{C}) = \neg \text{Upper}(\mathcal{C})(\text{SP}) \]

Propagation

\[ \text{Propagation}(\text{SP}, \mathcal{C}) = \langle (\perp'_\mathcal{T}, \perp'_\mathcal{I}), (\top'_\mathcal{T}, \top'_\mathcal{I}) \rangle \text{ s.t.} \]
\[ \perp'_\mathcal{T} = \perp_{\mathcal{T}} \cup \{ a \in \top_{\mathcal{T}} \setminus \perp_{\mathcal{T}} | \neg \text{Upper}(\mathcal{C}_{IP})(\text{SP} \setminus \{a\}) \} \]
\[ \perp'_\mathcal{I} = \perp_{\mathcal{I}} \cup \{ a \in \top_{\mathcal{I}} \setminus \perp_{\mathcal{I}} | \neg \text{Upper}(\mathcal{C}_{IP})(\text{SP} \setminus \{a\}) \} \]
\[ \top'_\mathcal{T} = \top_{\mathcal{T}} \setminus \{ a \in \top_{\mathcal{T}} \setminus \perp_{\mathcal{T}} | \neg \text{Upper}(\mathcal{C}_{IP})(\text{SP} \cup \{a\}) \} \]
\[ \top'_\mathcal{I} = \top_{\mathcal{I}} \setminus \{ a \in \top_{\mathcal{I}} \setminus \perp_{\mathcal{I}} | \neg \text{Upper}(\mathcal{C}_{IP})(\text{SP} \cup \{a\}) \} \]
Constraint handling

Checking

\[ \text{Checking}(SP, C) = \neg \text{Upper}(C)(SP) \]

Propagation

\[ \text{Propagation}(SP, C) = \langle (\bot'_T, \bot'_I), (\top'_T, \top'_I) \rangle \text{ s.t.} \]
\[ \bot'_T = \bot_T \cup \{ a \in \top_T \setminus \bot_T | \neg \text{Upper}(C_{IP})(SP \setminus \{a\}) \} \]
\[ \bot'_I = \bot_I \cup \{ a \in \top_I \setminus \bot_I | \neg \text{Upper}(C_{IP})(SP \setminus \{a\}) \} \]
\[ \top'_T = \top_T \setminus \{ a \in \top_T \setminus \bot_T | \neg \text{Upper}(C_{IP})(SP \cup \{a\}) \} \]
\[ \top'_I = \top_I \setminus \{ a \in \top_I \setminus \bot_I | \neg \text{Upper}(C_{IP})(SP \cup \{a\}) \} \]

Example - Propagation

Extraction task : \( C(X, Y) \equiv C_{FC} \land |X| \times |Y| < 6 \land X \cap \{t_4t_5\} \neq \emptyset. \)
\[ \text{Propagation}(\langle (t_3, i_2i_3), (t_1t_3t_4, i_2i_3) \rangle, C) = \langle (t_3t_4, i_2i_3), (t_3t_4, i_2i_3) \rangle. \]
Introduction
Generic algorithm for local pattern mining in 0/1 data
Data-Peeler: closed n-sets in n-ary relations

Biset mining
Enumeration strategy
Constraint handling
Finding upper bounds

Generic algorithm

\[ K = (G, M, I) \text{ a boolean context and } C_{EP} \text{ a constraint over } 2^T \times 2^T \]

Algorithm

\[ \text{Extract}(((\emptyset, \emptyset), (T, I))) \]

End Algorithm

\[ \text{Extract}(SP = ((\perp_T, \perp_I), (T_T, \top_T))) \]

\[ SP \leftarrow ((\perp_T, \perp_I), (T_T, \top_T)) \text{ where } \]
\[ \perp_T = \perp_T \cup \{a \in T_T \setminus \perp_T \mid \neg Upper(C_{EP})(SP \setminus \{a\})\} \]
\[ \perp_I = \perp_I \cup \{a \in T_I \setminus \perp_I \mid \neg Upper(C_{EP})(SP \setminus \{a\})\} \]
\[ T_T = T_T \setminus \{a \in T_T \setminus \perp_T \mid \neg Upper(C_{EP})(SP \cup \{a\})\} \]
\[ T_I = T_I \setminus \{a \in T_I \setminus \perp_I \mid \neg Upper(C_{EP})(SP \cup \{a\})\} \]
If \( \perp_T \subseteq T_T \land \perp_I \subseteq T_I \land Upper(C_{EP})(SP) \) Then
If \((\perp_T, \perp_I) \neq (T_T, T_I)\) Then
Let \( e \in T_T \setminus \perp_T \cup T_I \setminus \perp_I \)
\((SP_1, SP_2) \leftarrow (SP \cup \{e\}, SP \setminus \{e\})\)
\[ \text{Extract}(SP_1) \]
\[ \text{Extract}(SP_2) \]
Else Print\((\perp_T, \perp_I)\)
End If
End If
Piecewise anti-monotonic constraints

How to define constraint upper-bounds?

- Simple case: Monotonic and anti-monotonic constraints
- General case: given a constraint and a biset search space
**Definition**

\( \mathcal{C} \) is anti-monotone w.r.t. \( \subseteq \) iff \( P \subseteq P' \), \( \neg \mathcal{C}(P) \Rightarrow \neg \mathcal{C}(P') \)

\[
\text{Upper}(\mathcal{C})(SP) = \mathcal{C}(X_1, Y_1) \text{ where } \\
X_1 = \top_I \text{ if } \mathcal{C} \text{ is monotonic on } X \\
X_1 = \bot_I \text{ if } \mathcal{C} \text{ is anti-monotonic on } X \\
Y_1 = \top_I \text{ if } \mathcal{C} \text{ is monotonic on } Y \\
Y_1 = \bot_I \text{ if } \mathcal{C} \text{ is anti-monotonic on } Y
\]

**Examples**

- \( \mathcal{C}_{\text{division}} \equiv |X|/|Y| > \alpha : \)
  \[
  \text{Upper}(\mathcal{C}_{\text{division}})(SP) \equiv |\top_I|/|\bot_I| > \alpha
  \]

- \( \mathcal{C}_{\text{area}} \equiv |X| \times |Y| > \alpha : \)
  \[
  \text{Upper}(\mathcal{C}_{\text{area}}) \equiv |\top_I| \times |\top_I| > \alpha
  \]
### Piecewise anti-monotonic constraints

#### How to compute an upper-bound for the constraint $C_{\text{mean}}$?

$$C_{\text{mean}}(X) \equiv \frac{\sum_{x \in X} \text{Val}^+(x)}{\#X} > \alpha$$

$C_{\text{mean}}(X)$ can be rewritten as follows:

$$\mathcal{P}_{C_{\text{mean}}}(X_1, X_2) \equiv \frac{\sum_{x \in X_1} \text{Val}^+(x)}{\#X_2} > \alpha$$

Finally, we obtain:

$$\text{Upper}(C_{\text{mean}}) \equiv \sum_{i \in \mathcal{T}} \text{Val}^+(i)/|\bot_{\mathcal{T}}| > \alpha$$
Introduction
Generic algorithm for local pattern mining in 0/1 data
Data-Peeler: closed n-sets in n-ary relations

Biset mining
Enumeration strategy
Constraint handling
Finding upper bounds

Piecewise anti-monotonic constraints

How to compute an upper-bound for the constraint $C_{\text{whateveritis}}(E,F)$?

$$C_{\text{whateveritis}}(E,F)(X,Y) \equiv \frac{|X \cap E| \times |Y \cup F|}{|X| \times |Y|} > \alpha$$

We rewrite $C_{\text{whateveritis}}(E,F)(X,Y)$ as follows:

$$P_{C_{\text{whateveritis}}(E,F)}(X_1, X_2, Y_1, Y_2) \equiv \frac{|X_1 \cap E| \times |Y_1 \cup F|}{|X_2| \times |Y_2|} > \alpha$$

Finally, we obtain:

$$\text{Upper}(C_{\text{whateveritis}}(E,F)) \equiv \frac{|\top \cap T \cap E| \times |\top \cap T \cup F|}{|\bot \cap T| \times |\bot \cap T|} > \alpha$$

Data-Peeler
Genericity of the algorithm

The constraints that define formal concepts, itemsets or fault-tolerant patterns are special cases of piecewise (anti)-monotonic constraints. All this patterns can be extracted by means of this framework.
Introduction to Data-Peeler

Let us consider the following tasks:

**Data - Gene expression**

The gene expression levels of a set of genes in different situations along time (for different timestamps).

**Question**

What are the groups of genes that are co-expressed in different situations during a lot of timestamps?
Introduction to Data-Peeler

Let us consider the following tasks:

**Data - Log**
Users from several countries loading web pages about GNU/Linux distributions along time.

**Question**
What are the sets of countries for which their citizens are "interested in" the same distributions along time.
N-ary relations

In a binary relation $\mathcal{T} \times \mathcal{I}$, a formal concept binds a subset of $\mathcal{T}$ with a subset of $\mathcal{I}$ s.t. both are closed w.r.t. each other.

In a $n$-ary relation $A_1 \times \cdots \times A_n$, a $n$-sets generalizes a formal concept: it binds subsets of every $A_i$ s.t. each of them is closed w.r.t. all the others.
Example

$n$-ary boolean relation

\[ \mathcal{L} = 2^{A_1} \times \cdots \times 2^{A_n} \]

Dataset:

Pattern:
Example

$n$-ary boolean relation

\[ \mathcal{L} = 2^{A_1} \times \cdots \times 2^{A_n} \]

Dataset:

Pattern:
### Example

**n-ary boolean relation**

\[ L = 2^{A_1} \times \cdots \times 2^{A_n} \]

**Dataset:**

![Dataset Image]

**Pattern:**

![Pattern Image]
Closed \( n \)-sets

Constraints

\( H = \langle X^1, \cdots, X^n \rangle \) is a closed \( n \)-set if it satisfies:

- \( C_{\text{linked}} \equiv \forall U = (x^1, \cdots, x^n) \in X^1 \times \cdots \times X^n, \ U \in r \).
- \( C_{\text{closed}} \equiv \forall j \in 1 \cdots n, \ \forall x^i \in A^i \setminus X^j, \ \langle X^1, \cdots, X^j \cup \{x^i\}, \cdots, X^n \rangle \) does not satisfy \( C_{\text{linked}} \).

Constraints handling

\[ \text{Propagation}(SP, C_{\text{linked}}) = \langle (\perp'_{A_1}, \cdots, \perp'_{A_n}), (\top'_{A_1}, \top'_{A_n}) \rangle \] \text{s.t.} \ y \in A_j \text{ has been enumerated and} \ i \neq j:

\[ \top'_{A_i} = \top_{A_i} \setminus \{v \in \top_{A_i} \mid \neg C_{\text{linked}}(\langle \perp_{A_1}, \cdots, \perp_{A_j} \cup \{y\}, \cdots, \{v\}, \cdots, \perp_{A_n} \rangle)\} \].
Closed $n$-sets

Constraints

$H = \langle X^1, \cdots, X^n \rangle$ is a closed $n$-set if it satisfies:

- $C_{\text{linked}} \equiv \forall U = (x^1, \cdots, x^n) \in X^1 \times \cdots \times X^n, U \in r$.
- $C_{\text{closed}} \equiv \forall j \in 1 \cdots n, \forall x^j \in A^j \setminus X^j$, 
  $\langle X^1, \cdots, X^j \cup \{x^j\}, \cdots, X^n \rangle$ does not satisfy $C_{\text{linked}}$.

Constraints handling

$\text{Checking}(SP, C_{\text{closed}}) \equiv \forall j, \forall y \in A_j \setminus T_{A_j}$,
$\neg C_{\text{linked}}(T_{A_1}, \cdots, \{y\}, \cdots, T_{A_n})$
Introduction
Generic algorithm for local pattern mining in 0/1 data

Data-Peeler : closed n-sets in n-ary relations

Data-Peeler

Difficulties

- Pattern enumeration strategy is not straightforward anymore: lots of degrees of freedom
- No Galois connection
- How to deal with constraints?
- A 4-ary relation where dimensions are of size 200 contains up to $1.6 \times 10^9$ tuples. Which data structure should we choose for $n$-sets?
- No dimension should be privileged

State-of-the-art

Two algorithms for 3-ary relations: Trias and CubeMiner
Data-Peeler

Overview

- At any recursive call, any element (from any set) can be enumerated
  - A clever enumeration strategy improves the running times by several orders of magnitude
- Piecewise (anti)-monotonic constraints can be pushed (extended to $n$-ary relations)
Data-Peeler

Results

- All competitors focusing on ternary relations are much slower
- New relevant constraints
Applications

Graph

How to extract cliques in graphs evolving through 2 dimensions?

Example

For each semester and each country, we have (DistroWatch.com) GNU/Linux distributions for which people have a common interest. How to extract groups of GNU/Linux distributions which are loaded together for several (consecutive) semesters and in several countries.
Graph Problem

Extracted patterns

- {Fedora, FreeBSD, Debian, Ubuntu, Gentoo, MEPIS, Slackware, Yellow Dog, Mandriva, openSUSE} - every semester - countries all over the world: general-purpose distributions.

- {Astaro, ClarkConnect, IPCop, m0n0wall, Devil, SmoothWall, CensorNet} - every semester - countries all over the world (but slightly different from the previous ones): firewall oriented distributions.

- {dynebolic, ArtistX, AGNULA, MoviX, GeeXboX} - Every semester - occidental countries and India: movies and music manipulating distributions.

- {B2D, Linpus and PUD for Taiwan and Hong} - two semesters - Taiwan and Hong Kong: traditional Chinese distributions.
Discretization

**Robustness w.r.t. binarization**

Let a matrix of dimension $n$, how to binarize such data set when several binarization methods are relevant? What about relevancy?

**Example**

For each semester and each country, we have (DistroWatch.com) GNU/Linux distributions for which people have a common interest. From initial numerical data set, one can apply different equally relevant discretization methods: "Per-distribution", "Per-semester", "Per-country" and "Global" binarization. How to make binarization more robust?
Solution

Add a dimension corresponding to the discretization methods.

<table>
<thead>
<tr>
<th>Minimal nb of binarizations</th>
<th>Nb of closed 4-sets</th>
<th>Time (in s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3580</td>
<td>18.93</td>
</tr>
<tr>
<td>4</td>
<td>1297</td>
<td>15.82</td>
</tr>
<tr>
<td>5</td>
<td>229</td>
<td>11.62</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>8.58</td>
</tr>
</tbody>
</table>

We constrained the minimal number of binarizations varying from 3 (presence in at least half of the binarizations) to 6 (presence in every binarization).