Proceedings SIAM International Conference on Data Mining SDM'08, Atlanta, USA, April 24-26, 2008.

Data-Peeler: Constraint-Based Closed Pattern Mining in n-ary Relations

Loïc Cerf* Jérémy Besson† Céline Robardet‡ Jean-François Boulicaut§

Abstract

Set pattern discovery from binary relations has been extensively studied during the last decade. In particular, many complete and efficient algorithms which extract frequent closed sets are now available. Generalizing such a task to n-ary relations ($n \geq 2$) appears as a timely challenge. It may be important for many applications, e.g., when adding the time dimension to the popular $objects \times features$ binary case. The generality of the task — no assumption being made on the relation arity or on the size of its attribute domains — makes it computationally challenging. We introduce an algorithm called Data-Peeler. From a n-ary relation, it extracts all closed n-sets satisfying given piecewise (anti)-monotonic constraints. This new class of constraints generalizes both monotonic and anti-monotonic constraints. Considering the special case of ternary relations, Data-Peeler outperforms the state-of-the-art algorithms CubeMiner and Trias by orders of magnitude. These good performances must be granted to a new clever enumeration strategy allowing an efficient closeness checking. An original application on a real-life 4-ary relation is used to assess the relevancy of closed n-sets constraint-based mining.

1 Introduction

Constraint-based mining has become a popular framework for supporting pattern discovery tasks. First, it enables to provide more interesting patterns when the analyst can specify a priori relevancy by means of constraints. Next, this has been identified as a key issue to achieve the tractability of many data mining tasks: useful constraints can be deeply pushed into the extraction process such that it is possible to get complete (every pattern which satisfies the user-defined constraint is computed) though efficient algorithms.

In this paper, we focus on patterns that hold in 0/1 data sets. In a popular setting, such data sets generally correspond to relations between two attributes only, e.g., $transactions \times items$ or $objects \times features$. Frequent itemset mining or formal concept mining are typical data mining tasks in such binary relations. Frequent itemset mining has been introduced by Agrawal et al. [2, 3]. To tackle difficult cases, one major breakthrough has been the study of (frequent) closed set mining, formal concepts being the mapping between closed sets of items (or features) and their supporting sets of transactions (or objects) [4, 7, 9, 17, 18, 19].

We address here the more general problem of closed pattern mining in n-ary discrete-valued relations. Hereafter, such patterns are called closed n-sets. When $n = 2$, this task turns to be the classical closed set mining from a binary relation. Mining n-ary relations with $n > 2$ is clearly useful across multiple application domains. For example, in the context of sale data analysis, we can easily have relations crossing items, customers, dates, and regions. We may want to extract maximal associations between such attributes for business decision making. Another typical (generic) application domain concerns the numerous situations where object properties can be recorded as features for a collection of objects over time. This typically provides ternary relations.

A formal concept associates a closed set of transactions with a closed set of items. This mapping is bijective. In this perspective, a formal concept is a maximal pattern w.r.t. the two sets. This definition, used in FCA (Formal Concept Analysis), is "generalizable" to n-ary relations. However, the bijection is not between two sets anymore. Each $(n-1)$-sets can be associated with the remaining one.

Ideas aiming at directly reusing FCA on preprocessed n-ary relations do not seem to work. The issue is to find a bijection between n-ary relations and binary ones on which the extraction of formal concepts provides the closed n-sets of the original relation. Such a transformation would certainly lead to a combinatorial explosion of the number of attributes. Indeed the attributes of the binary relation should combine several elements of the different sets to encompass the n-ary relation.

The main challenge of constraint-based closed n-set mining in n-ary relations relies on the ability to push constraints during the extraction and to handle an im-

*INSA-Lyon, LIRIS UMR5205, F-69621 Villeurbanne, France.
†INSA-Lyon, LIRIS UMR5205, F-69621 Villeurbanne, France.
‡INSA-Lyon, LIRIS UMR5205, F-69621 Villeurbanne, France.
§INSA-Lyon, LIRIS UMR5205, F-69621 Villeurbanne, France.

portant amount of data. This is especially difficult when no assumption is made on the arity of the relation and their attribute domain sizes. The pattern enumeration strategy becomes even more important than for itemset and/or formal concept extraction. Indeed, one cannot enumerate anymore one attribute domain (usually items) and compute the rest of the pattern thanks to a Galois connection. In the general case, $n-1$ attributes are needed to determine the remaining attribute.

Furthermore, the enumeration strategy has a major impact on the class of constraints which can be efficiently pushed. To achieve tractability, this is especially crucial to perform an efficient closeness constraint checking. Algorithms TRIAS [11] and CUBEMINER [12] have been recently proposed to compute closed 3-sets in ternary relations. They have different enumeration strategies. TRIAS basically relies on formal concept mining (i.e., closed 2-set mining) from two different binary relations that are projections of the original ternary relation. It works well if we assume that at least one attribute has a small domain size. CUBEMINER uses a ternary enumeration that recursively splits the data set into smaller pieces. Unfortunately, several additional checks must be performed to ensure the unicity of the extracted patterns.

We propose a new algorithm called DATA-PEELER. It is inspired by D-MINER [4] which computes complete collections of closed 2-sets that satisfy minimal size constraints in binary relations. DATA-PEELER extracts closed n-sets in n-ary relations when $n \geq 2$. It is based on an original enumeration process which considers any attribute domain on a same basis (i.e., no selection of a particular one is done a priori). It enables to push a large class of constraints called *piecewise (anti)-monotonic constraints*. In particular, this class includes the classes of monotonic and anti-monotonic constraints. Furthermore, DATA-PEELER efficiently checks the closeness constraint in such a way that there is no need to store previously computed patterns. DATA-PEELER can exploit new constraints like the promising *isolated constraint*: it enables to compute only patterns containing elements which are "significantly different from the elements "outside" it".

The rest of the paper is organized as follows. We formalize the mining task in Section 2 and we discuss the type of constraints our algorithm handles. In Section 3, we present the DATA-PEELER algorithm which extracts closed n-sets under constraints in n-ary relations. Implementation issues are discussed in Section 4. Section 5 studies space complexity. Experimental results are provided in Section 6. Finally, related work is discussed in Section 7 and Section 8 briefly concludes.

	A	B	C	A	B	C	A	B	C
1	1	1	1	1	1	1	1	1	
2	1	1		1			1	1	
3		1				1	1		1
4			1	1		1	1	1	1
	α			β			γ		

Figure 1: Boolean representation of the relation $\mathcal{R}_E \subseteq \{A, B, C\} \times \{1, 2, 3, 4\} \times \{\alpha, \beta, \gamma\}$

2 Problem Setting

Let $A^1, \cdots, A^n$ be n discrete-valued attributes whose domains are respectively $D^1, \cdots, D^n$. $\mathcal{R}$ is a n-ary relation on these attributes, i.e., $\mathcal{R} \subseteq D^1 \times \cdots \times D^n$. n-sets are elements of $2^{D^1} \times \cdots \times 2^{D^n}$. We now provide the formal definitions of closed n-sets and the new class of piecewise (anti)-monotonic constraints. We use the $\sharp$ operator to denote set cardinality.

2.1 Closed n-sets Closed n-sets are a generalization of formal concepts or closed itemsets to n-ary relations. Intuitively, a n-set $H = \langle X^1, \cdots, X^n \rangle$ s.t. $X^i \subseteq D^i$ is a closed n-set iff (a) all elements of each set X^i are in relation with all the other elements of the other sets in $\mathcal{R}$, and (b) X^i sets cannot be enlarged without violating (a). Formally, H is a closed n-set iff it satisfies both the constraints $\mathcal{C}_{connected}$ and $\mathcal{C}_{closed}$:

DEFINITION 1. *($\mathcal{C}_{connected}$) Pattern H satisfies $\mathcal{C}_{connected}$ iff $\forall U = (x^1, \cdots, x^n) \in X^1 \times \cdots \times X^n$, $U \in \mathcal{R}$.*

DEFINITION 2. *($\mathcal{C}_{closed}$) A pattern H that satisfies $\mathcal{C}_{connected}$ is closed iff $\forall j = 1 \cdots n$, $\forall x^j \in D^j \setminus X^j$, $\langle X^1, \cdots, X^j \cup \{x^j\}, \cdots, X^n \rangle$ does not satisfy $\mathcal{C}_{connected}$.*

In binary relations, bi-sets $\langle X^1, X^2 \rangle$ satisfying $\mathcal{C}_{connected} \wedge \mathcal{C}_{closed}$ are formal concepts (i.e., X^1 and X^2 are closed sets).

EXAMPLE 1. *Figure 1 provides a ternary relation $\mathcal{R}_E$. $\langle (A, B), (1, 2), (\alpha, \gamma) \rangle$ and $\langle (C), (4), (\alpha, \beta, \gamma) \rangle$ are examples of 3-sets in $\mathcal{R}_E$. The 3-set $\langle (A, B), (1, 2, 3), (\alpha, \gamma) \rangle$ violates $\mathcal{C}_{connected}$ because $(A, 3, \alpha) \notin \mathcal{R}_E$ or $(B, 3, \gamma) \notin \mathcal{R}_E$. $\langle (C), (3, 4), (\beta) \rangle$ satisfies $\mathcal{C}_{connected}$ but not $\mathcal{C}_{closed}$ because $(C, 1, \beta) \in \mathcal{R}_E$ or $(C, 3, \gamma) \in \mathcal{R}_E \wedge (C, 4, \gamma) \in \mathcal{R}_E$.*

2.2 Piecewise (anti)-monotonic constraints Enabling user-defined constraints is extremely useful to support subjective interestingness and thus the relevancy of the extracted collections. It is also well-known

that the active use of constraints (i.e., "pushing" them into the extraction phase) is a key issue to achieve extraction tractability (i.e., working on large domain sizes and/or a high density of related elements). For example, we may ask for patterns with a minimal number of elements in some domains (i.e., a counterpart of the classical minimal frequency constraint on itemsets) and/or patterns covering at least a given number of elements of $\mathcal{R}$ (i.e., some kind of minimal area or volume constraint). We now define the monotonicity property of constraints in the context of n-set mining.

DEFINITION 3. *(Monotonicity) Let us consider a constraint $\mathcal{C}$ taking m sets $P_1, \cdots, P_m$ as arguments. Each argument is a subset of an attribute domain. $\mathcal{C}$ is monotonic on its i^{th} argument iff $\forall P_1, \cdots, P_m$ and $\forall E_1, E_2$ s.t. $E_1 \subset E_2$, $\mathcal{C}(P_1, \cdots, P_{i-1}, E_2, P_{i+1}, \cdots, P_m) \Rightarrow \mathcal{C}(P_1, \cdots, P_{i-1}, E_1, P_{i+1}, \cdots, P_m)$.*

When we have $\mathcal{C}(P_1, \cdots, P_{i-1}, E_1, P_{i+1}, \cdots, P_m) \Rightarrow \mathcal{C}(P_1, \cdots, P_{i-1}, E_2, P_{i+1}, \cdots, P_m)$, constraint $\mathcal{C}$ is said anti-monotonic.

This definition is a straightforward extension of the monotonicity as defined on binary relations. If a constraint is (anti)-monotonic on each of its arguments, then it is (anti)-monotonic following the classical terminology in the itemset mining framework.

It is however possible to define a new and larger class of constraints which can be efficiently exploited. These constraints may have an argument occurring several times in their definitions. For instance, assume we want to compute each n-set having a mean above a threshold α on a criterion $Val^+ : D^i \rightarrow \mathbb{R}^+$:

$$\mathcal{C}_1(X^i) \equiv \frac{\sum_{x \in X^i} Val^+(x)}{\sharp X^i} \geq \alpha$$

The argument X^i appears twice in the expression of $\mathcal{C}_1$. To introduce the notion of piecewise monotonic constraint, we have to rewrite such a constraint by using a different argument for each occurrence of the same argument. Our example constraint can be rewritten as the new constraint $\mathcal{P}_{\mathcal{C}_1}$:

$$\mathcal{P}_{\mathcal{C}_1}(P_1, P_2) \equiv \frac{\sum_{x \in P_1} Val^+(x)}{\sharp P_2} \geq \alpha$$

A second example would be a constraint which specifies that each n-set has to contain a proportion of a given 2-set $\langle E, F \rangle$ larger than a threshold α:

$$\mathcal{C}_2(X^i, X^j) \equiv \frac{\sharp(X^i \cap E) \times \sharp(X^j \cap F)}{\sharp X^i \times \sharp X^j} \geq \alpha$$

Such a constraint can be rewritten as:

$$\mathcal{P}_{\mathcal{C}_2}(P_1^i, P_2^i, P_1^j, P_2^j) \equiv \frac{\sharp(P_1^i \cap E) \times \sharp(P_1^j \cap F)}{\sharp P_2^i \times \sharp P_2^j} \geq \alpha$$

We can now define the class of piecewise (anti)-monotonic constraints.

DEFINITION 4. *(Piecewise (anti)-monotonic constraint) A constraint $\mathcal{C}$ is piecewise (anti)-monotonic if its associated constraint $\mathcal{P}_{\mathcal{C}}$ is either monotonic or anti-monotonic on each of its arguments.*

Both $\mathcal{C}_{connected}$ and $\mathcal{C}_{closed}$ constraints are piecewise (anti)-monotonic. Some other examples of piecewise (anti)-monotonic constraints are:

- $\mathcal{C}_{\mu-size}(X) \equiv \sharp X \geq \mu$
- $\mathcal{C}_{\nu-volume}(X^1, \cdots, X^m) \equiv \prod_{i=1}^{m} \sharp X^i \geq \nu$
- $\mathcal{C}_{\epsilon-almost\ square}(X^k, X^l) \equiv \frac{\sharp X^k}{\sharp X^l} - \frac{\sharp X^l}{\sharp X^k} \leq \epsilon \wedge \frac{\sharp X^l}{\sharp X^k} - \frac{\sharp X^k}{\sharp X^l} \leq \epsilon$
- $\mathcal{C}_{diffval}(X) \equiv \sum_{x \in X} Val_1^+(x) - \sum_{x \in X} Val_2^+(x) \geq 0$, where Val_1^+ and Val_2^+ are positive functions.

Among the piecewise (anti)-monotonic constraints, let us discuss a promising one when considering pattern relevancy. For a given closed n-set, nothing enforces the outside elements to be different enough from the inside elements. In other terms, an element outside of the n-set may be in relation with almost all the elements of the n-set. To avoid the extraction of such n-sets, we propose a new constraint, namely $\mathcal{C}_{\delta-isolated}$, defined as follows:

DEFINITION 5. *($\mathcal{C}_{\delta-isolated}$) A n-set $H = \langle X^1, \cdots, X^n \rangle$ is isolated w.r.t. the attribute X^i, denoted by $\mathcal{C}_{\delta-isolated}(H, i)$, iff $\forall x \in D^i \setminus X^i$,*

$$\sharp(K \setminus \mathcal{R}) > \delta \times \sharp K$$

where $K = X^1 \times \cdots \times \{x\} \times \cdots X^n$ and $\delta \in [0, 1]$ is a user-defined parameter.

In other words, if this $\mathcal{C}_{\delta-isolated}$ constraint is satisfied, each $x \in D^i$ that is outside the n-set must have a density (in terms of relative number of elements in $\mathcal{R}$) on the elements inside the n-set lower than $1-\delta$. When $\delta = 1$, any element of D^i outside the n-set must not be in relation with elements from the $(D^j)_{j \neq i}$ contained in the n-set. When $\delta = 0$, the $\mathcal{C}_{0-isolated}$ constraint is equivalent to the $\mathcal{C}_{closed}$ constraint on dimension i.

EXAMPLE 2. *In our running example $\mathcal{R}_E$, for $H = \langle (A, B, C), (1), (\alpha, \beta) \rangle$, $\mathcal{C}_{0.4-isolated}(H, 2)$ is true but $\mathcal{C}_{0.5-isolated}(H, 2)$ is not because of the element 2 or the element 4.*

To summarize, the data mining task considered in this paper is the extraction of closed n-sets that satisfy piecewise (anti)-monotonic constraints and, in particular, the $\mathcal{C}_{\delta-isolated}$ constraint.

3 The DATA-PEELER Algorithm

3.1 Enumeration strategy The enumeration of all the patterns by materializing and traversing all possible n-sets is in practice not feasible. Therefore, we look for a decomposition of the original search space into smaller pieces such that each portion can be independently studied in main memory and such that the union of the closed n-sets extracted from each portion is the whole collection of closed n-sets. Thus, n-sets are explored in a depth-first search manner.

DATA-PEELER uses a binary enumeration. Each node N in the enumeration tree is a pair (U, V) where U and V are two n-sets. N *represents* all the n-sets containing all the elements of U and a subset of the elements of V. In other words, this is the search space of n-sets $\langle X^1, \cdots, X^n\rangle$ s.t. $\forall i = 1 \cdots n$, $U^i \subseteq X^i \subseteq U^i \cup V^i$. Notice that the root node $(\langle\emptyset, \cdots, \emptyset\rangle, \langle D^1, \cdots, D^n\rangle)$ represents all possible n-sets. In the contrary, nodes such that $\forall i \in 1 \ldots n$, $V^i = \emptyset$ represent a single n-set $\langle U^1, \ldots, U^n\rangle$.

EXAMPLE 3. *The node* $E = (U, V) = (\langle(B), (1,2), (\alpha)\rangle, \langle(A), (3), \emptyset\rangle)$ *represents the 3-sets* $\langle(B), (1,2), (\alpha)\rangle, \langle(A,B), (1,2), (\alpha)\rangle, \langle(B), (1,2,3), (\alpha)\rangle$ *and* $\langle(A,B), (1,2,3), (\alpha)\rangle$.

At a node $N = (U, V)$, DATA-PEELER recursively selects an element p from V (see below for the selection criterion) and generates two new nodes $N_L = (U_L, V_L) = (U \cup \{p\}, V \setminus \{p\})$ and $N_R = (U_R, V_R\rangle = \langle U, V \setminus \{p\})$. N_L (respectively N_R) represents the n-sets of N which contain (resp. do not contain) p.

EXAMPLE 4. *Considering the node E from Example 3, the selection of element* 3 *of V leads to the two nodes* $E_L = (\langle(B), (1,2,3), (\alpha)\rangle, \langle(A), \emptyset, \emptyset\rangle)$ *and* $E_R = (\langle(B), (1,2), (\alpha)\rangle, \langle(A), \emptyset, \emptyset\rangle)$.

3.2 Checking $\mathcal{C}_{connected}$ It is possible to exploit constraint $\mathcal{C}_{connected}$ to reduce the size of V_L and then cut down the number of candidates to be considered. Indeed elements of V that cannot be added to U_L without violating $\mathcal{C}_{connected}$ can be safely removed from V_L.

More formally, let $U = \langle U^1, \cdots, U^n\rangle$, $V = \langle V^1, \cdots, V^n\rangle$ and $N = (U, V)$. If the selected element is $p^j \in V^j$, then $N_L = (U_L, V_L)$ and $N_R = (U_R, V_R)$, returned by $Children(N, p^j) = (N_L, N_R)$, are such that:

- $U_L = \langle U^1, \cdots, U^j \cup \{p^j\}, \cdots, U^n\rangle$
- $V_L = \langle V'^1, \cdots, V'^n\rangle$ such that $V'^i = V^i \setminus \{v \in V^i \mid \neg\, \mathcal{C}_{connected}(\langle U^1, \cdots, \{p^j\}, \cdots, \{v\}, \cdots, U^n\rangle\}$ when $i \neq j$ and $V'^j = V^j$.
- $U_R = U$
- $V_R = \langle V^1, \cdots, V^j \setminus \{p^j\}, \cdots, V^n\rangle$

U_L now contains p^j meaning that p^j belongs to all the n-sets represented by N_L. All the elements of $V \setminus \{p^j\}$ that, once added to U_L, lead to unconnected n-sets, have been removed from V_L. For N_R, p^j is simply removed from V_R and then N_R does not contain n-sets with p^j anymore. Thanks to this enumeration strategy, all the $\mathcal{C}_{connected}$ n-sets are extracted once and only once.

EXAMPLE 5. *In our running example, when enforcing* $\mathcal{C}_{connected}$*, we eventually obtain* $E_L = (\langle(B), (1,2,3), (\alpha)\rangle, \langle\emptyset, \emptyset, \emptyset\rangle)$. *Indeed, element A cannot be added to* $U_L = \langle(B), (1,2,3), (\alpha)\rangle$ *to form a n-set satisfying* $\mathcal{C}_{connected}$ *because* $(A, 3, \alpha) \notin \mathcal{R}$. *Thus A is absent from* V_L.

Until now, we discussed how to extract all n-sets satisfying $\mathcal{C}_{connected}$ in n-ary relations. We now need to enforce the closeness property.

3.3 Checking $\mathcal{C}_{closed}$ We want to exploit the closeness constraint during the enumeration process (i.e., giving rise to safe pruning) rather than in a post-processing phase. Basically, if there exists an element p^j such that $p^j \in D^j \setminus (U^j \cup V^j)$ and $\mathcal{C}_{connected}(\langle U^1 \cup V^1, \cdots, \{p^j\}, \cdots, U^n \cup V^n\rangle)$ is satisfied, then all the n-sets represented by N can be extended with p^j to form a larger n-set satisfying $\mathcal{C}_{connected}$. In other terms, they are not closed. In that case, the n-sets are closed in the data set formed by elements of U and V but not in the whole data set. Therefore, such a node can be safely pruned.

Thanks to our enumeration strategy, we do not have to check every element of $D^1 \times \cdots \times D^n \setminus (U \cup V)$. Indeed, elements which have been removed from V when applying $\mathcal{C}_{connected}$ do not need to be checked. By definition, they have been removed because they cannot be used to form any connected n-set with the elements of U. Only the elements selected and removed during the enumeration, that is to say when N_R is built, have to be checked. We use a stack denoted $\mathcal{S}$ in which we store such elements.

More formally, the closeness constraint is defined in the node space as follows: $\mathcal{C}_{closed}((U, V), \mathcal{S}) \equiv \forall e \in \mathcal{S}, \neg\mathcal{C}_{connected}(\langle U^1 \cup V^1, \cdots, \{e\}, \cdots, U^n \cup V^n\rangle)$.

EXAMPLE 6. *Referring to the running example, assuming that* $\gamma \in \mathcal{S}$ *at the node E, then* E_L *satisfies* $\mathcal{C}_{closed}$*, whereas* E_R *does not (*3 *can extend it).*

3.4 Piecewise (anti)-monotonic constraints Let us now study how DATA-PEELER can take advantage

of piecewise (anti)-monotonic constraints, i.e., how it can prune a node as early as possible without missing any closed n-set. The idea is to define a new constraint $\mathcal{M}od_{\mathcal{C}}$ in the node space such that, for all H represented by N, $\neg\mathcal{M}od_{\mathcal{C}}(N) \Rightarrow \neg\mathcal{C}(H)$. In other words, if a node does not satisfy $\mathcal{M}od_{\mathcal{C}}$ then no n-set it represents satisfies $\mathcal{C}$. When $V^i = \emptyset$, for all i, we have $\neg\mathcal{M}od_{\mathcal{C}}(N) \Leftrightarrow \neg\mathcal{C}(H)$

DEFINITION 6. *Let $\mathcal{C}$ a piecewise (anti)-monotonic constraint. Let us denote by $\mathcal{M}(\mathcal{P}_{\mathcal{C}})$ (respectively $\mathcal{A}(\mathcal{P}_{\mathcal{C}})$) the set of parameters of $\mathcal{P}_{\mathcal{C}}$ (see Definition 3) which are monotonic (resp. anti-monotonic). By definition, $\mathcal{A}(\mathcal{P}_{\mathcal{C}}) \cup \mathcal{M}(\mathcal{P}_{\mathcal{C}})$ contains all the parameters of $\mathcal{P}_{\mathcal{C}}$.*

We can now define $\mathcal{M}od(\mathcal{C})$:

$$\mathcal{M}od_{\mathcal{C}}((U,V)) \equiv \mathcal{P}_{\mathcal{C}}(P_1, \cdots, P_m)$$

where $\forall j = 1 \ldots m$, $P_j = U^i$ if parameter P_j belongs to $\mathcal{M}(\mathcal{P}_{\mathcal{C}})$ and is related to attribute domain i or $P_j = U^i \cup V^i$ if parameter P_j belongs to $\mathcal{A}(\mathcal{P}_{\mathcal{C}})$ and is related to attribute domain i.

EXAMPLE 7. *Let $\mathcal{C}_1(\langle X^1, X^2\rangle) \equiv \sharp X^1 \times \sharp X^2 \geq 10$ and $\mathcal{C}_2(\langle X^1, X^2, X^3\rangle) \equiv \sharp X^1 / \sharp X^2 \geq 8 \wedge a \notin X^3$ be two piecewise (anti)-monotonic constraints. The related constraints in the node space are:*

- $\mathcal{M}od_{\mathcal{C}_1}((U,V)) \equiv \mathcal{C}_1(U^1 \cup V^1, U^2 \cup V^2)$
- $\mathcal{M}od_{\mathcal{C}_2}((U,V)) \equiv \mathcal{C}_2(U^1 \cup V^1, U^2, U^3)$

In Example 3, $\mathcal{M}od_{\mathcal{C}_1}((\langle\langle(A,B),(1,2)\rangle, \langle(C),(4)\rangle\rangle)) \equiv \sharp\{A,B,C\} \times \sharp\{1,2,4\} \geq 10$ is false. This node can be safely pruned.

Figure 2 depicts a part of the enumeration tree of DATA-PEELER on $\mathcal{R}_E$. It illustrates Examples 4 to 6. Every closed 3-set $\langle X^1, X^2, X^3\rangle$ satisfying $\mathcal{C}_{5-volume}$ is to be extracted. The bold circled leaf is a closed n-set. The crossed nodes are pruned.

4 Implementation

4.1 Algorithm DATA-PEELER is a depth-first search algorithm. It takes two arguments: the current node N and its related stack $\mathcal{S}$. It starts with the root node $N_0 = (\langle\emptyset, \cdots, \emptyset\rangle, \langle D^1, \cdots, D^n\rangle)$ and an empty stack $\mathcal{S} = \emptyset$. Its major steps are presented in Algorithm 1. First of all, the closeness property is checked (see Section 3.3) as well as a user-defined piecewise (anti)-monotonic constraint $\mathcal{C}$ (see Section 3.4). If they are both satisfied, either the n-set U is output if there is no element to be enumerated anymore, or the enumeration process keeps going by splitting the current node N into two new nodes. In that case, an element p of V to be enumerated is selected (see below Section 4.2). Then, the two new nodes are built with the function $Children(N,p)$ described in Section 3.2. Finally, DATA-PEELER$(N_L, \mathcal{S})$ and DATA-PEELER$(N_R, \mathcal{S} \cup \{p\})$ are recursively called. Notice here that the stack $\mathcal{S}$ of N_R now contains p. Indeed, p has been removed from N_R by the enumeration and not by the enforcement of $\mathcal{C}_{connected}$.

Algorithm 1 DATA-PEELER

```
Input: A node N = (U, V) and a stack S
Output: Closed n-sets satisfying C
if C_closed(N, S) ∧ Mod_C(N) then
  if Is_empty(V) then
    output U = ⟨U^1, ··· , U^n⟩
  else
    p ← Select(V)
    (N_L, N_R) ← Children(N, p)
    DATA-PEELER(N_L, S)
    DATA-PEELER(N_R, S ∪ p)
  end if
end if
```

4.2 Selecting the element to be enumerated As we saw in Section 3.1, an element p of V is selected to proceed with the enumeration, i.e., to generate two new nodes partitioning the original one. Our selection strategy is to select p to maximize the number of elements that $\mathcal{C}_{connected}$ enforcement can potentially remove from V. First, DATA-PEELER sorts elements of every attribute domain in increasing order w.r.t. their density in $\mathcal{R}$. We use the fact that the less elements are connected in $\mathcal{R}$, the more likely $\mathcal{C}_{connected}$ will help to remove elements from the current node during the enumeration phase.

So far, the choice of the attribute domain, on which the element is to be enumerated, remains open. Elements of an attribute may be removed only when (a) they are in V and (b) elements from the $n-1$ other attributes are in U. The following formula gives the maximum number of elements of $\mathcal{R}$ which are browsed when enforcing $\mathcal{C}_{connected}$ after an element from V^d is enumerated:

$$\sum_{k \neq d} \left(\sharp V^k \times \prod_{l \notin \{d,k\}} \sharp U^l \right)$$

We choose to enumerate an element on the attribute domain d maximizing this formula. The experiment in Section 6.2 empirically shows that the proposed selection criteria outperforms other sensible criteria.

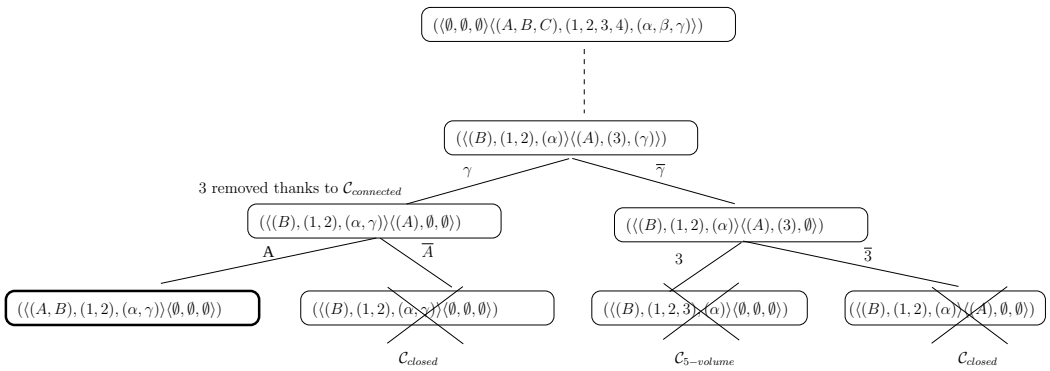

Figure 2: Part of the enumeration tree on $\mathcal{R}_E$

5 Space Complexity

The size (in bits) of an element ID is denoted a and the size (in bits) of a pointer is denoted b.

5.1 Storing the data set Unlike for binary relation mining algorithms, it is not possible to speed up the extraction by storing for each element e of $D^1, \ldots, D^n$ the projection of the input data set $\mathcal{R}$ on e (usually called "tidset"). The use of sophisticated data structures like FP-Trees [10] remains an open problem because of the multiple attributes to be considered and the need to use each one during the enumeration.

The whole data set must be stored in main memory to check both $\mathcal{C}_{connected}$ and $\mathcal{C}_{closed}$. Two classes of data structures were investigated, namely a bitset-based structure, and a list-based structure.

In both cases, the data set is stored in a complete prefix tree of height $n-1$ corresponding to the $n-1$ first attributes. The nodes at depth $i \in 0 \ldots n-2$ always have $\sharp D^{i+1}$ children, one for every element of D^{n+1}. From depth 0 to $n-2$, the edges binding a node to its children are pointers. Each leaf stands for a prefix of size $n-1$ of every element of $D^1 \times \cdots \times D^{n-1}$. The difference between the two studied structures resides in how the last attribute elements are stored.

5.1.1 The bitset-based structure In such a structure, every leaf of the prefix tree points to a bitset representing the last attribute elements. A "0" (respectively "1") in the bitset stands for the absence (respectively the presence) of the related element of $\mathcal{R}$. The presence of such an element is tested in constant time. The space occupied by the data set is:

$$\underbrace{b\sum_{i=0}^{n-1}\prod_{j=1}^{i}\sharp D^i}_{\text{the depths from 0 to } n-1} + \underbrace{\prod_{j=1}^{n}\sharp D^j}_{\text{the bitsets}}$$

5.1.2 List-based structure Here, every leaf points to a list of IDs of elements of D^n. Each of them represents an element of $\mathcal{R}$.

The presence of such an element is tested in $\mathcal{O}(\log \sharp D^n)$. Choosing D^n as the smaller attribute domain minimizes the access time.

If $d = \frac{\sharp\mathcal{R}}{\prod_{i=1}^{n}\sharp D^i}$ denotes the density of the data set, the space requirement is:

$$\underbrace{b\sum_{i=0}^{n-1}\prod_{j=1}^{i}\sharp D^j}_{\text{the depths from 0 to } n-1} + \underbrace{a \times d\prod_{j=1}^{n}\sharp D^j}_{\text{the lists}}$$

Compared to the bitset-based structure, a space gain occurs if and only if $d < \frac{1}{a}$. Taking $a = 64$ (size of an integer on modern hardware), the density of the data set must be under 1.56% for the list-based structure to present a space advantage over the bitset-based structure. Thus, the bitset-based structure is always better in data access time and, in most cases, in space requirement too. Therefore, we choose this structure for our implementation.

Notice that other sparser structures were theoretically investigated. They consist in using an incomplete prefix tree. Of course, the time access cost increases ($\mathcal{O}(\sum_{i=1}^{n}\log\sharp D^i)$ for a totally sparse tree). Furthermore, the space requirement can be greater since we need to add an element ID to each node. Indeed the child node addressed by a pointer cannot be identified from the position of the child in the list of children (some are "missing"). It can be shown that a space gain occurs only when, in average, a node at depth i has less than $\frac{b}{a+b}\sharp D^{i+1}$ children. Unless the data set is very sparse and/or non-homogeneous, even depth $n-2$ does not satisfy such a property. This justifies the fact that we focused on the *list-based structure* where only the deepest level is sparse.

5.2 Storing the nodes of the enumeration tree Both U and $\mathcal{S}$ can be statically stored. At every recursive call, one single element is pushed in either U (when constructing N_L) or $\mathcal{S}$ (when constructing N_R) and popped once this recursive call is completed.

Any element of V can be removed when $\mathcal{C}_{connected}$ is enforced. As a result, V cannot be statically stored. The construction of the enumeration tree being depth-first, the worst case is bound to reaching the deepest node. At worst, the depth of the enumeration tree is $\sum_{i=1}^{n} \sharp D^i$ where each recursive call removes only one element from V. In this case, the required space to store V is:

$$a \sum_{i=1}^{\sum_{j=1}^{n} \sharp D^j} i = \frac{a}{2} \sum_{j=1}^{n} \sharp D^j \times (\sum_{j=1}^{n} \sharp D^j - 1)$$

5.3 Space complexity Combining the results from Section 5.1 and Section 5.2, the space complexity of Data-Peeler is:

- $\mathcal{O}((\sharp D^1 + \sharp D^2)^2)$ if $n = 2$ (the space requirement for the V set predominates)
- $\mathcal{O}(\prod_{i=1}^{n} \sharp D^i)$ if $n > 2$ (the space requirement for the data set predominates)

6 Experimental Results

Every experiment described here has been performed on a GNU/Linux system equipped with a AMD 2600+ processor and 512 Mo of RAM. Data-Peeler is implemented in C++ and compiled with GCC 4.1.2. Every plotted curve uses a logarithmic scale for its time axis.

6.1 Presentation of the data sets

6.1.1 Synthetic data sets To study the behavior of Data-Peeler and to compare it to competitors in different situations, we have used the IBM Quest data generator [3]. Various "basket data"-like data sets with predefined attributes and densities have been generated. Three attributes are considered, namely the customers, the bought items, and the time periods (in months).

6.1.2 Real data sets To assess the added-value of Data-Peeler both in terms of the relevancy of the extracted n-sets and its performance w.r.t. competitors, we have been working on logs from the `DistroWatch.com` website. This popular website gathers comprehensive information about GNU/Linux and BSD distributions. Every distribution being described on a separate page, a visitor loading such a page is considered "interested" in the distribution. Every IP address is analyzed to identify the country it comes from. Timestamps allow to study the evolution of the interest granted to the different distributions along time. The whole data set gathers 36 months, 243 countries and 538 distributions. Two different data sets have been derived from it. In both cases, data have been normalized so that every country and every time period has the same importance. They have been transformed in 0/1 data in the following way: for each distribution, we kept the elements of $\mathcal{R}$ containing this distribution and such that its normalized interest exceeds a threshold equals to one quarter of the maximal normalized interest for this distribution in $\mathcal{R}$.

6.2 Impact of the enumeration strategy Let us first empirically compare the enumeration strategy presented in Section 4.2 with two other sensible strategies. For each node (U, V),

- the enumerated attribute j is chosen such that it has the smallest non-empty $\sharp V^j$. Among all the elements of V^j, the element with the smallest density in $\mathcal{R}$ is selected.
- the enumerated element $p^j \in \cup_{i=1}^{n} V^i$ is chosen such that $(D^1 \times D^2 \times \ldots \times \{p^j\} \times \ldots \times D^n) \setminus \mathcal{R}$ has the largest cardinality.

The first strategy enumerates every element of the $n-2$ attributes with the smallest cardinalities. Then, when enumerating elements from the two remaining attributes, $\mathcal{C}_{connected}$ may finally succeed in reducing the V set.

The second strategy globally sorts the elements of the attributes altogether. If every attribute domain has the same cardinality, this order follows a growing density. Otherwise, an element p^j from a small attribute domain size is usually preferred since $D^1 \times \ldots \times \{p^j\} \times \ldots \times D^n$ is larger.

Tests have been performed on the data sets generated by the Quest data generator. Whereas the chosen enumeration strategy of Data-Peeler scales very well, the other strategies force us to choose small size attributes to be able to plot results: 36 customers buying in average 6 items out of 18 (density of about 33%) per month. The number of months has been varying from 6 to 36 and we enforced the constraint that every closed 3-set had to contain at least 3 customers, 2 items and 3 months.

Results are represented in Figure 3. The enumeration strategy of Data-Peeler largely outperforms the two other strategies. The performances of Enumeration 1 mainly depend on the size of the smallest attribute domain (above 18 months, the smallest attribute domain becomes the set of items which is constant).

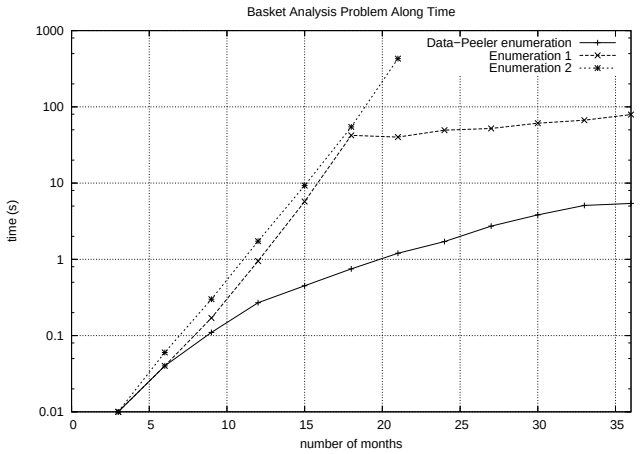

Figure 3: Comparing Data-Peeler enumeration with two other sensible strategies

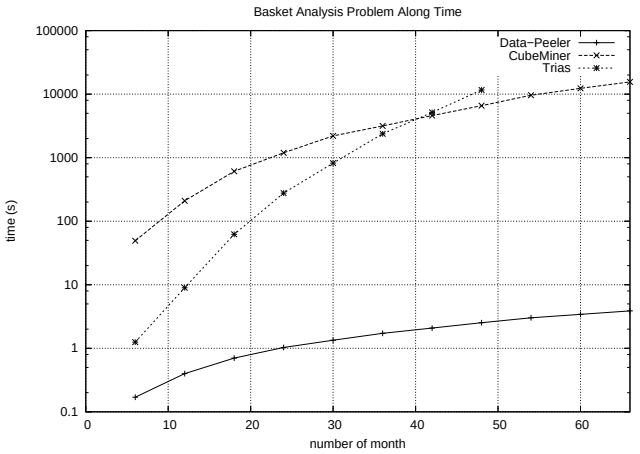

Figure 4: Comparison w.r.t. CubeMiner and Trias

This behavior, as mentioned earlier, is due to the complete enumeration of the smallest domain. The performance of Enumeration 2 emphasizes the need, when selecting the element to be enumerated, to take into account the characteristics of the current node.

6.3 Comparisons with competitors Data-Peeler is compared with both CubeMiner [12] and Trias [11] on 3-ary relations. We have been using the implementations provided by their respective authors.

6.3.1 Mining synthetic data sets Comparisons with CubeMiner and Trias are achieved on synthetic data sets provided by the Quest data generator. 144 customers buying in average 6 items out of 72 (density of about 8.3%) per month have been generated. We make the number of months vary from 6 to 66 and we constrain every closed 3-set to involve at least 2 customers, 2 items and 2 months.

The results are represented in Figure 4. Data-Peeler outperforms its competitors by several orders of magnitude. The growing number of months (the smallest domain) significantly alters the performance of Trias, whereas it has less effect on CubeMiner.

For example, considering data along 48 months, to extract all the 5801 closed n-sets, CubeMiner takes 1 hour and 50 minutes, Trias 3 hours and 14 minutes, whereas Data-Peeler only needs 2.5 seconds. Unlike its competitors, even with 600 months, Data-Peeler is still able to extract all closed n-sets in a reasonable time, i.e., 1 minute and 21 seconds for 431892 closed n-sets.

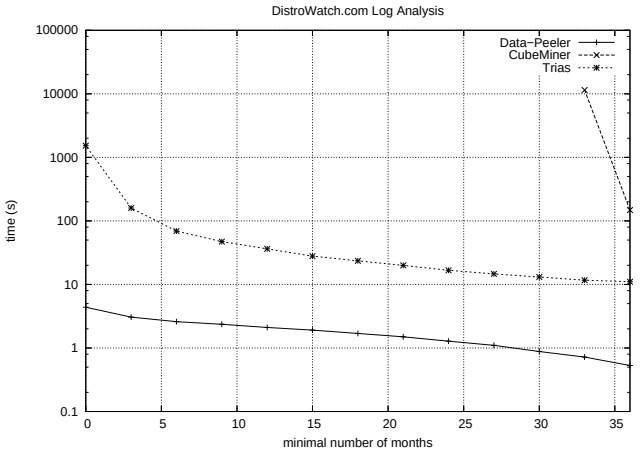

Figure 5: Comparison w.r.t. CubeMiner and Trias (real data)

6.3.2 Empirical evaluation on a real data set A ternary relation has been derived from the logs of `DistroWatch.com`. It indicates, month after month, whether visitors from a country look interested in a distribution. All data (36 months, 243 countries and 538 distributions) have been kept. Compared to the synthetic data sets considered above, this one is relatively large even if it has one small attribute domain. It is also much sparser since its density is 0.55%. We have constrained every closed 3-set to involve at least 2 countries and 2 distributions. The minimal number of months a closed 3-set must contain has been varying from 0 to 36.

Results are represented in Figure 5. Data-Peeler outperforms its competitors by several orders of magnitude. Thanks to the small number of months in the data set, Trias succeeds in extracting the closed 3-sets even

without any constraint on the time attribute. CubeMiner suffers a lot from the global size of the data set. It is unable to perform the extraction under a size constraint of 33/36 months.

With a requirement of at least 6 months out of 36, Data-Peeler needs 2.6 seconds and Trias 69.3 seconds to extract all the 87 patterns.

Without any minimal size constraint on the number of months, 10658 closed n-sets are computed in 4.4 seconds by Data-Peeler and in 1531 seconds by Trias. In both cases, CubeMiner can not perform the task.

6.4 A qualitative feedback

6.4.1 Data and problem setting We have derived an interesting 4-ary relation from the logs of `DistroWatch.com`. It takes in consideration visitors (identified with their IP addresses) who have loaded at least two different distribution pages the same day. It is assumed that this is a sign of a common interest in the visited distributions. The less relevant countries and distributions have been removed. The days have been aggregated in semesters (the release period of many distributions). In the end, we have a 4-ary relation $\mathcal{R}_{DW}(D_1, D_2, S, C)$ indicating that people from country C (among 39) show a common interest in distributions D_1 and D_2 (among 323) during the semester S (among 6). This relation covers 1.7% of the possible associations between attributes. We aim at extracting all the maximal sets of distributions which are simultaneously interesting for people from a maximal set countries during a maximal set of semesters. To obtain them, we need to extract the closed 4-sets $\langle X_1, X_2, X_3, X_4\rangle \in 2^{D_1} \times 2^{D_2} \times 2^S \times 2^C$ of $\mathcal{R}_{DW}$ such that $X_1 = X_2$. This constraint is anti-monotonic. However, handling it by a modification of the enumeration strategy is much more efficient: whenever a distribution is chosen to be enumerated (either moved from V to U or from V to $\mathcal{S}$), the same distribution in the other domain is moved in the same manner. In the following, all the extracted n-sets will satisfy this constraint.

6.4.2 Minimal size and volume constraints Data-Peeler extracts every closed 4-set from the data set in 229 seconds. Since it is impossible to inspect all the 602290 closed 4-sets, minimal sizes are enforced on every set: we constrain every closed 4-set to involve at least 2 semesters, 2 countries and 3 distributions. After 20 seconds, Data-Peeler provides 17196 closed 4-sets. Again, it prevents from a systematic interpretation of each closed 4-set.

Therefore, we have enforced a volume constraint to keep only the largest patterns. We considered the new constraint $\mathcal{C}_{v-weighted\,volume} \equiv \sum_{d=1}^{n} (\sharp X^d)^{\alpha(d)} \geq v$ where $\alpha(d) = \frac{n \sum_{i=1}^{n} \sharp X^i}{\sharp X^d}$. It is a generalization of a minimal volume constraint where every attribute is weighted w.r.t. its cardinality. The function α is such that elements from an attribute domain which is twice smaller will "count" twice more in the weighted volume. $\mathcal{C}_{v-Weighted\,Volume}$ is anti-monotonic. It reduces the computation to 14 seconds and the number of extracted closed 4-sets to 352.

Given such a collection of 352 patterns, it has been possible to manually inspect them and assess their relevancy.

- 94 closed 4-sets have on the distribution domains a subset of {Fedora, FreeBSD, Debian, Ubuntu, Gentoo, MEPIS, Slackware, Yellow Dog, Mandriva, openSUSE}. Considering all of them, every semester is mentioned. All these distributions are mainstream general-purpose distributions. These closed 4-sets involve many countries all over the world. However Great Britain shows off by being present in all these n-sets but one.

- 64 closed 4-sets have on the distribution domains a subset of {Astaro, ClarkConnect, IPCop, m0n0wall, Devil, SmoothWall, CensorNet}. Every semester is involved. These seven distributions are meant to serve a common interest: they are all specifically designed to act as a firewall. These closed 4-sets involve countries from every continent. Australia (closely followed by Belgium) is the most present country.

- 80 closed 4-sets have on the distribution domains a subset of {dyne:bolic, ArtistX, AGNULA, MoviX, GeeXboX}. Every semester is involved. These five distributions are meant to serve a common interest: they are all specifically designed to manipulate movies and music. These 4-sets mainly contain occidental countries but India is very present too. Switzerland belongs to all these 4-sets. GNU/Linux is obviously a popular choice among Swiss artists.

- 83 closed 4-sets have on the distribution domains a subset of {dyne:bolic, ArtistX, AGNULA, MoviX, GeeXboX} $\cup$ {Ubuntu, Damn Small, KNOPPIX, MEPIS, PCLinuxOS, Xandros}. It could be seen as a "collision" between two separate interests. Nevertheless, the distributions from the second set being primarily designed for desktop use, they are also suited to play movies and music. Furthermore, every distribution from these two sets uses the APT package management system (if any).

The few remaining closed 4-sets are interesting too. Among the distributions which appear once in the returned closed 4-sets, Gentoox and GentooTH form with Gentoo a closed 4-set running along the last four semesters (GentooTH did not exist before) in 11 countries. Their common point is obvious from their names: they are all based on Gentoo.

In the same way, Knopperdisk and Feather are mentioned in one single closed 4-set where they are associated with Damn Small along the last five semesters (Knopperdisk did not exist before). These distributions have a strong common point: all of them are KNOPPIX light derivatives aimed at being installed on a USB pen drive.

6.4.3 δ-isolated constraint Instead of searching for large closed patterns, we now focus on extracting closed 4-sets which are isolated in the country domain by enforcing the $\mathcal{C}_{\delta-isolated}$ constraint.

We first set $\delta = 0.75$ on the country attribute. Keeping the size constraints from the previous section, Data-Peeler returns one single closed 4-set. It involves, during two semesters, the distributions B2D, Linpus and PUD for Taiwan and Hong Kong. These three distributions are Taiwanese and insist on the direct support of traditional Chinese. Traditional Chinese characters are almost exclusively used in Taiwan, Hong Kong and Macao (which was not kept among the 39 countries in the data set). Indeed, the impact of this particularity is revealed thanks to the $\mathcal{C}_{0.75-isolated}$ constraint.

When lowering δ to 0.7, Data-Peeler returns two additional closed 4-sets. Both of them refer to Russia and Ukraine during 2006 and involve ALT, ASP and one mainstream distribution (either Ubuntu or Mandriva). Both ALT and ASP are Russian distributions. Again, the particularity of the Russian alphabet as the default character encoding is captured thanks to the $\mathcal{C}_{\delta-isolated}$ constraint.

7 Related Work

We do not consider that pattern discovery in binary relations (e.g., mining closed 2-sets) has to be studied in details here. Instead, we focus on n-ary relation data mining where $n \geq 3$. A couple of analysis techniques for real valued matrices with n dimensions are mentioned too.

7.1 Real-valued matrices Considering kinetic microarray data, Jiang et al. [14] have proposed an ad-hoc multi steps algorithm to compute maximal sets of genes which are coherent on a subset of samples during the whole time series. For each gene and each pair of samples, it computes the Pearson's correlation coefficient between these two time series. A user defined threshold is then used to capture strong correlations. Then a $samples \times samples$ matrix is constructed for each gene from which maximal cliques are computed. They correspond to maximal sets of samples coherent for the corresponding gene. Finally, for each sample set, the corresponding maximal gene set is computed, i.e., a gene g is associated with a sample set S if there exists a maximal coherent sample set S_g (a clique associated with g) such that $S \subseteq S_g$. Such a processing is more efficiently achieved with Data-Peeler. Furthermore, the time dimension is considered at a global scale. Thus the method is unable to find temporal trends applicable to only a subset of the time points.

Considering the same targeted application, [20] proposes to mine different types of clusters in ternary relations ($genes \times samples \times timepoints$). It first computes a range multigraph for each time slice. Vertices stand for biological samples and there is an edge for each gene set having a similar expression ratio. On this graph, maximal cliques are computed and post-processed to produce bi-clusters: the set of samples associated with the vertices and the set of genes obtained by intersecting the sets associated with the edges. Finally, a new multigraph is computed where each time point is a vertex and each pair of highly overlapping bi-clusters (gene-set, samples) forms an edge between two time steps. Tri-clusters are obtained by computing maximal cliques on such a graph. By constructing a ternary relation $genes \times samples \times timepoints$, Data-Peeler can compute similar patterns in a single step.

Sun et al. [16] propose to extend Principal Component Analysis to sequences of tensors (data cubes with $M \geq 3$ attribute domains). Each tensor gathers the measurements for a time step. The so called tensor analysis consists in computing M orthogonal matrices such that the reconstruction error is minimized. Each projection matrix crosses one attribute domain of the data with syntactical variables summarizing it. Like PCA, the results of tensor analysis may be difficult to interpret since all the data coordinates participate in the linear combination that may mix both positive and negative weights, which might partly cancel each other and turn the result to be difficult to understand.

7.2 Logic minimization Data-Peeler extracts patterns from a n-ary relation between finite sets. Considering these sets as the domains of multiple-valued variables, the relation can be seen as the truth table of a multiple-valued logic function with a range {'0', '1'}. Boolean functions are a specialization of this framework where every set gathers two elements (usually bound to

the semantics "true" and "false").

The *Karnaugh map* [13] is a tool to minimize such Boolean functions. This method is to be applied by hand ("by eye" would be more correct since it exploits the human capability to discern geometrical patterns). For this reason, it works well for up to four variables, and it becomes unpractical for more than six variables. It relies on organizing the truth table in such a way that every maximal rectangle of '1' gives a "prime implicant" (a disjunction of conjunctions) minimizing the part of the Boolean function responsible for the '1's of the rectangle. Once every '1' is "covered" by at least one prime implicant, the disjunction of the prime implicants is a minimization of the original Boolean function. Later, the *QuineMcCluskey algorithm* [15] was designed to deal with more variables. The procedure basically remains the same. However, the organization of the truth table, used in the *Karnaugh map*, is substituted by a tabular form which better suits computers' way of processing data. This algorithm always returns the minimal form of the Boolean function to the cost of finding all prime implicants.

The Espresso algorithm [6] uses a different approach. The returned function is not always the minimal form (but close to it) and the computation is reduced (in both memory and time) by orders of magnitude. It is still heavily used, in particular in the design of logic circuits. Espresso was adapted to deal with multiple-valued logic functions too.

In this perspective, Data-Peeler is a prime implicant extractor for multiple-valued logic functions. However, it has been designed for the extraction of patterns under constraints, whereas Espresso (or its predecessors) outputs a tiling of the data set. Furthermore Data-Peeler can handle huge data sets without the time explosion occurring with Espresso.

7.3 Mining multi-relational data Multi-relational data mining leads to further extensions of pattern mining in binary relations. New problems arise as how to preserve the monotonic properties of the new patterns and how to check pattern closeness defined w.r.t. the relations. Afrati et al. [1] have proposed different algorithms and conditions under which the Apriori framework can be used to preserve the monotonicity properties. Garriga et al [8] have proposed, within an inductive logic programming framework, a clever way to unify several pattern mining problems, e.g., itemset and graph mining. Intuitively, patterns are sets of elements connected from sets of relations and closed w.r.t. these connections. Using two popular subsumptions and two interpretations, they propose different algorithms to deal with multi-relational data sets.

7.4 3-set mining Let us now consider the two direct competitors w.r.t. our proposal, namely CubeMiner [12] and Trias [11]. We already illustrated in Section 6.3 that our algorithm significantly outperforms these state-of-the-art algorithms.

First, in [12], Ji et al. have proposed two algorithms to compute closed 3-sets for 3-ary relations. The first one, called *Representative slice mining* consists in computing all subsets of the smallest attribute domain, and for each subset constructing the corresponding 2D Boolean matrix using the bitwise AND operator between elements of this attribute. Then, a frequent pattern algorithm is used on each Boolean matrix and a post-processing phase removes 3-sets which are not closed.

The second one, called CubeMiner, directly operates on the ternary relation. It consists in using cubes $X \times Y \times Z$ called cutters such that none of their elements are in relation with each others, i.e., generalizing the cutters introduced for constraint-based mining of formal concepts in [4]. These cutters are recursively applied to generate candidates containing less 0 values than their parents: for each cutter, 3 candidates are constructed, one without elements from X, a second one without elements from Y and a third one without elements of Z. Several checks must be performed on each candidate to ensure their closeness and their unicity. The unicity is obtained by making sure that removed elements are not included in a previously applied cutters on this branch. It requires to intersect the current cutter with all those which were previously used. The closeness is evaluated by making sure that there is no previously used cutters such that their elements could be added to the current candidates. Consequently, each candidate must be compared two times to the elements of the cutter list. The enumeration process of Data-Peeler does not require to check the unicity of each candidate.

In [11] , Jaschke et al. have proposed the Trias that also computes closed 3-sets in a ternary relation. It basically relies on formal concept computation on two different binary relations. To compute all closed n-sets on a relation $\mathcal{R} \subseteq D^1 \times D^2 \times D^3$, Trias first constructs a new relation $\mathcal{R}_1 \subseteq (D^1, D^2 \times D^3)$ whose columns correspond to couples of elements from D^2 and D^3. Formal concepts are extracted in this relation. Each formal concept $\langle A, B \rangle$ is such that B contains couples from $D^2 \times D^3$ in relation with each of the elements of $A \subseteq D^1$. The set B stands for a relation which is not fully connected (i.e., there are false values in its Boolean representation). Thus, in a second step, Trias computes formal concepts in the relation generated by B and checks if the obtained formal concepts are closed w.r.t. D^1. This step is easily carried out by checking if its closure is equal to A.

8 Conclusion

We have proposed a new correct and complete algorithm called Data-Peeler and we have defined the class of constraints it can efficiently exploit. From a n-ary relation, Data-Peeler computes every closed n-set satisfying given piecewise (anti-)monotonic constraints.

Previous works mainly deal with the binary relation case (i.e., the popular 0/1 matrix case) and numerous algorithms have been proposed to compute collections of closed 2-sets, possibly constrained by user-defined properties. Recently, two algorithms were designed for the ternary relation case, namely CubeMiner and Trias. Although Data-Peeler is designed to handle relations of any arity, our empirical study has illustrated that, in various settings, Data-Peeler outperforms both CubeMiner and Trias by orders of magnitude.

A real-life 4-ary relation has been mined to assess the qualitative added-value of closed n-set mining. In this application, we have interpreted the relevancy of the extracted patterns under various piecewise (anti)-monotonic constraints. In particular, the introduced $\mathcal{C}_{\delta-isolated}$ constraint has been proved useful.

We look forward to experimenting Data-Peeler in different fields. So far, we have considered Basket Data Analysis and Web Usage Mining scenarios but we might investigate very soon its application for kinetic gene expression data analysis and dynamic graph mining. We also plan to tackle the abundance of very similar closed n-sets by clustering them, a simple idea that has been recently proved useful in the 2-dimensional case [5].

Acknowledgments We want to thank the authors of CubeMiner and Trias algorithms for providing their implementations and Ladislav Bodnar for sharing with us the `DistroWatch.com` logs. This work is partly funded by EU contract IST-FET IQ FP6-516169, INRA and ANR Bingo2 (MDCO 2007).

References

[1] F. Afrati, G. Das, A. Gionis, H. Mannila, T. Mielikainen and P. Tsaparas, *Mining Chains of Relations*, IEEE ICDM'05, pp. 553–556.

[2] R. Agrawal, T. Imielinski and A. Swami, *Mining Association Rules between Sets of Items in Large Databases*, ACM SIGMOD'93, pp. 207–216.

[3] R. Agrawal and R. Srikant, *Fast Algorithms for Mining Association Rules in Large Databases*, VLDB'94), Morgan Kaufmann, pp. 487–499 (Introduction to the Quest data generator).

[4] J. Besson, C. Robardet, J-F. Boulicaut and S. Rome. *Constraint-Based Formal Concept Mining and its Application to Microarray Data Analysis.* Intelligent Data Analysis 9(1), IOS Press (2005), pp. 59–82

[5] S. Blachon, R. G. Pensa, J. Besson, C. Robardet, J.F.Boulicaut and O. Gandrillon. *Clustering Formal Concepts to Discover Biologically Relevant Knowledge from Gene Expression Data*, In Silico Biology 7(0033) (2007).

[6] R. K. Brayton, C. T. McMullen, G. D. Hachtel and A. L. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*, Kluwer Academic Publishers (1984).

[7] A. Gely, *A Generic Algorithm for Generating Closed Sets of a Binary Relation*, IFCA'05, Springer LNCS 3403, pp. 223–234.

[8] G.C. Garriga, R. Khardon and Luc De Raedt, *On Mining Closed Sets in Multi-Relational Data*, IJCAI'07, pp. 804–809.

[9] B. Goethals and M. Zaki. *Advances in Frequent Itemset Mining Implementations: Report on FIMI'03*, SIGKDD'04 Explorations 6(1), pp. 109–117.

[10] J. Han, J. Pei, and Y. Yin. *Mining Frequent Patterns without Candidate Generation*, ACM SIGMOD'00, pp. 1–12.

[11] R. Jaschke, A. Hotho, C. Schmitz, B. Ganter and G. Stumme, *TRIAS: An Algorithm for Mining Iceberg Tri-Lattices*, IEEE ICDM'06, pp. 907–911.

[12] L. Ji, K-L. Tan and A. K. H. Tung, *Mining Frequent Closed Cubes in 3D Data Sets*, VLDB'06, Morgan Kaufmann, pp. 811–822.

[13] M. Karnaugh, *The Map Method for Synthesis of Combinational Logic Circuits*, Transactions of American Institute of Electrical Engineers part I (November 1953), pp. 593–599.

[14] D. Jiang, J. Pei, M. Ramanathan, C. Tang and A. Zhang, *Mining Coherent Gene Clusters from Gene-Sample-Time Microarray Data*, ACM SIGKDD'04, pp. 430–439.

[15] E. L. McCluskey, *Minimization of Boolean Functions*, Bell System Technical Journal (April 1959), pp. 149–175.

[16] J. Sun, D. Tao and C. Faloutsos, *Beyond Streams and Graphs: Dynamic Tensor Analysis*, ACM SIGKDD'06, pp. 374–383.

[17] T. Uno, M. Kiyomi and H. Arimura, *LCM ver.3: Collaboration of Array, Bitmap and Prefix Tree for Frequent Itemset Mining*, ACM OSDM'05, pp. 77–86.

[18] J. Wang, J. Han and J. Pei, *CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets*, ACM SIGKDD'03, pp. 236–245.

[19] M. J. Zaki and C-J. Hsiao, *CHARM: An Efficient Algorithm for Closed Itemset Mining*, SIAM SDM'02.

[20] L. Zhao and M. J. Zaki, *TriCluster: An Effective Algorithm for Mining Coherent Clusters in 3D Microarray Data*, ACM SIGMOD'05, pp. 694–705.