Closed Patterns Meet *n*-ary Relations

LOÏC CERF
INSA-Lyon
JÉRÉMY BESSON
Institute of Mathematics and Informatics
and
CÉLINE ROBARDET and JEAN-FRANÇOIS BOULICAUT
INSA-Lyon

3

Set pattern discovery from binary relations has been extensively studied during the last decade. In particular, many complete and efficient algorithms for frequent closed set mining 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, for example, 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 an n-ary relation, it extracts all closed n-sets satisfying given piecewise (anti) monotonic constraints. This new class of constraints generalizes both monotonic and antimonotonic 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 to efficiently enforce the closeness property. The relevance of the extracted closed n-sets is assessed on real-life 3-and 4-ary relations. Beyond natural 3-or 4-ary relations, expanding a relation with an additional attribute can help in enforcing rather abstract constraints such as the robustness with respect to binarization. Furthermore, a collection of closed n-sets is shown to be an excellent starting point to compute a tiling of the dataset.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems; H.2.8 [**Database Management**]: Database Applications—*Data mining*

General Terms: Algorithms

Additional Key Words and Phrases: Closed patterns, constraint-based mining, constraint properties, n-ary relations, tiling

This work is partly funded by EU contract IST-FET IQ FP6-516169, INRA and ANR BINGO2 (MDCO 2007).

Authors' addresses: L. Cerf, INSA-Lyon, LIRIS CNRS UMR5205, F 69621 Villeurbanne, France; email: Loic.cerf@liris.cnrs.fr; J. Besson, Institute of Mathematics and Informatics, LT-08663 Vilnius, Lithuania; C. Robardet, J.-F. Boulicaut, INSA-Lyon, LIRIS CNRS UMR5205, F 69621 Villeurbanne, France.

DOI 10.1145/1497577.1497580 http://doi.acm.org/10.1145/1497577.1497580

ACM Reference Format:

Cerf, L., Besson, J., Robardet, C., and Boulicaut, J. F. 2009. Closed patterns meet n-ary relations. ACM Trans. Knowl. Discov. Data. 3, 1, Article 3 (March 2009), 36 pages. DOI = 10.1145/1497577.1497580 http://doi.acm.org/10.1145/ 1497577.1497580

1. INTRODUCTION

Constraint-based mining has become a popular framework for supporting pattern discovery tasks (see, e.g., Boulicaut and Jeudy [2005]). 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 satisfying the user-defined constraint is computed) though efficient algorithms.

In this article, we focus on patterns that hold in 0/1 datasets. In a popular setting, such datasets generally correspond to relations between two attributes only, for example, *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. [1993] and Agrawal and Srikant [1994]. To tackle difficult cases, one major breakthrough has been the study of (frequent) closed itemset mining (see, e.g., Pasquier et al. [1999], Pei et al. [2000], Wang et al. [2003], Zaki and Hsiao [2002], Uno et al. [2005], Grahne and Zhu [2005], and Goethals and Zaki [2004] for an experimental survey). Formal concepts (see, e.g., Stumme et al. [2002], Besson et al. [2005], and Gély [2005]) are closed sets of items (or features) associated to their supporting sets of transactions (or objects).

We address here the more general problem of closed pattern mining in n-ary relations. Hereafter, such patterns are called closed n-sets. When $n = 2$, this task turns out to be the classical formal concept mining in binary relations. 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.

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 important amount of data. This is especially difficult when no assumption is made on the arity of the relation and the attribute domain sizes. The pattern enumeration strategy becomes even more important than for itemset or formal concept extraction. Indeed, it is no longer possible to enumerate one attribute domain (usually items) and compute the rest of the pattern thanks to a Galois connection. A closed set in one attribute domain is related to one and only one closed set of the other attribute domain. In n-ary relations, subsets of $n - 1$

attribute domains are needed to determine the subset of the remaining attribute domain. Indeed, a $n-2$-set does not define a unique closed n-set.

Furthermore, the enumeration strategy has a major impact on the class of constraints that can be efficiently pushed. To achieve tractability, it is especially crucial to efficiently enforce the closeness constraint. Algorithms TRIAS [Jaschke et al. 2006] and CUBEMINER [Ji et al. 2006] 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 dataset into smaller pieces. Unfortunately, several additional checks must be performed to ensure the unicity of the extracted patterns.

This article presents an algorithm, called DATA-PEELER, which has been introduced in Cerf et al. [2008]. The preliminary version focused on the theoretical basis of the algorithm. We provide here many more details about the algorithm itself, but also new and efficient optimizations. Moreover, we introduce here several original applications of closed n-set mining like robustness assessment with respect to binarization techniques or n-ary relation tiling. DATA-PEELER is inspired by D-MINER [Besson et al. 2005], namely an algorithm which computes complete collections of closed 2-sets satisfying minimal size and/or area constraints in binary relations. DATA-PEELER extracts closed n-sets in n-ary relations where $n \geq 2$. It is based on an original enumeration process which considers any attribute on a same basis. Thus, the order in which the elements are selected is performed along the enumeration time (not a priori). Our algorithm can efficiently exploit (i.e., "push" at extraction time) a broad class of constraints called piecewise (anti)-monotonic constraints. This class includes (but is not restricted to) monotonic and antimonotonic constraints. For instance, it can exploit the *isolated constraint* that enforces to compute only patterns containing elements that are significantly different from the elements "outside" it. Furthermore, DATA-PEELER enforces the closeness constraint in such a way that there is no need to store previously computed patterns.

The rest of the article is organized as follows. In Section 2, we formalize the mining task and discuss the type of constraints our algorithm handles. In Section 3, we present the DATA-PEELER algorithm that extracts closed n-sets under constraints. Implementation issues are discussed in Section 4. Section 5 studies space complexity. Experimental results are provided in Section 6. Section 7 focus on an original use of closed n-set mining to ensure robustness with respect to binarization. Section 8 details how to postprocess DATA-PEELER output to solve the tiling problem on n-ary relations. Finally, related work is discussed in Section 9, and Section 10 briefly concludes.

2. PROBLEM SETTING

Let $A^1, \ldots, A^n$ be n categorical attributes and assume their domains are respectively $D^1, \ldots, D^n$. Also $\mathcal{R}$ is an n-ary relation on these attributes, that is, $\mathcal{R} \subseteq D^1 \times \cdots \times D^n$. Moreover, n-sets are elements of $2^{D^1} \times \cdots \times 2^{D^n}$. In

	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		

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

this section, we provide the formal definitions of a *closed n-set* and of the new class of piecewise (anti)-monotonic constraints. We use the $\sharp$ operator to denote set cardinality. The so-called *n-sets* manipulated in this article are essentially n-tuples of sets. To make the notations easier to read, a 3-set denoted $\langle(\alpha, \gamma), (1, 2), (A, B)\rangle$ stands for the 3-tuple $(\{\alpha, \gamma\}, \{1, 2\}, \{A, B\})$. Furthermore, given an n-set $H = \langle X^1, \ldots, X^n\rangle$, we will write that $e \in H$ instead of $\exists i = 1 \ldots n$ such that $e \in X^i$. In the same spirit, given $e \in D^i$, $H \cup \{e\}$ (respectively, $H \setminus \{e\}$) will denote the n-set $\langle X^1, \ldots, X^i \cup \{e\}, \ldots, X^n\rangle$ (respectively,. $\langle X^1, \ldots, X^i \setminus \{e\}, \ldots, X^n\rangle$).

2.1 Closed *n*-Sets

Closed n-sets are a generalization of *formal concepts* to n-ary relations when $n > 2$. Intuitively, an n-set $H = \langle X^1, \ldots, X^n\rangle$ such that $\forall i = 1 \ldots n, X^i \subseteq D^i$ 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 2.1 ($\mathcal{C}_{connected}$). Pattern H satisfies $\mathcal{C}_{connected}$ in $\mathcal{R}$ iff

$$\forall u = (x^1, \ldots, x^n) \in X^1 \times \cdots \times X^n, u \in \mathcal{R}.$$

Definition 2.2 ($\mathcal{C}_{closed}$). Pattern H satisfies $\mathcal{C}_{closed}$ in $\mathcal{R}$ iff

$\forall j = 1 \ldots n, \forall x^j \in D^j \setminus X^j, \langle X^1, \ldots, X^j \cup \{x^j\}, \ldots, X^n\rangle$ does not satisfy $\mathcal{C}_{connected}$.

In binary relations, a closed 2-set is a 2-set $\langle X^1, X^2\rangle$ satisfying $\mathcal{C}_{connected} \wedge \mathcal{C}_{closed}$. In the literature, it has often been called the *formal concept* since Wille [1982]. It can also be defined as a closed itemset and its supporting set of objects/transactions.

Example 2.3. Figure 1 provides a ternary relation $\mathcal{R}_E \subseteq \{\alpha, \beta, \gamma\} \times \{1, 2, 3, 4\} \times \{A, B, C\}$. The relation could represent customers (1, 2, 3, and 4) buying items (A, B, and C) along three months (α, β, and γ).

$\langle(\alpha, \gamma), (1, 2), (A, B)\rangle$ and $\langle(\alpha, \beta, \gamma), (4), (C)\rangle$ are examples of closed 3-sets in $\mathcal{R}_E$. $\langle(\alpha, \gamma), (1, 2), (A, B)\rangle$ shows that the customers 1 and 2 buy both items A and B during the months α and γ ($\mathcal{C}_{connected}$). Moreover, it is closed with respect to every attribute ($\mathcal{C}_{closed}$).

—There is no other month during which these two customers buy these two items.

—No other customer buys these two items during these two months.

—No other item is simultaneously bought by these two customers during these two months.

The 3-set $\langle(\alpha, \gamma), (1, 2, 3), (A, B)\rangle$ violates $\mathcal{C}_{connected}$ because $(\alpha, 3, A) \notin \mathcal{R}_E$ or $(\gamma, 3, B) \notin \mathcal{R}_E$. The 3-set $\langle(\beta), (3, 4), (C)\rangle$ satisfies $\mathcal{C}_{connected}$ but not $\mathcal{C}_{closed}$ because $(\beta, 1, C) \in \mathcal{R}_E$ or $(\gamma, 3, C) \in \mathcal{R}_E \wedge (\gamma, 4, C) \in \mathcal{R}_E$.

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 2.4 (*Monotonicity*). Let us consider a constraint $\mathcal{C}$ taking m sets $P_1^{\sigma(1)}, \ldots, P_m^{\sigma(m)}$ as arguments. Each argument $P_i^{\sigma(i)}$ is a subset of the attribute domain $D^{\sigma(i)}$, σ being a mapping between the parameter indices and the attribute domain ones ($\sigma(i) = j$ iff $P_i^j \subseteq D^j$). Specifically, $\mathcal{C}$ is monotonic on its i^{th} argument iff $\forall P_1^{\sigma(1)}, \ldots, P_m^{\sigma(m)}$ and $\forall E, F$ such that $E \subseteq F \subseteq D^{\sigma(i)}$, $\mathcal{C}(P_1^{\sigma(1)}, \ldots, E, \ldots, P_m^{\sigma(m)}) \Rightarrow \mathcal{C}(P_1^{\sigma(1)}, \ldots, F, \ldots, P_m^{\sigma(m)})$. Dually, $\mathcal{C}$ is antimonotonic on its i^{th} argument iff $\mathcal{C}(P_1^{\sigma(1)}, \ldots, F, \ldots, P_m^{\sigma(m)}) \Rightarrow \mathcal{C}(P_1^{\sigma(1)}, \ldots, E, \ldots, P_m^{\sigma(m)})$.

This definition is a straightforward extension of the monotonicity as defined on binary relations. If a constraint is monotonic (respectively, antimonotonic) on each of its arguments, then it is monotonic (respectively, antimonotonic).

It is, however, possible to define a new and broader class of constraints that can be efficiently exploited. The so-called *piecewise (anti)-monotonic constraints* include classical constraints, such as monotonic and antimonotonic. In the expression of a piecewise (anti)-monotonic constraint every argument can occur several times. When this argument grows (with respect to the inclusion order), some of its occurrences tend to satisfy the constraint, whereas the other ones tend to violate it. Differentiating these two kinds of occurrences in two separate arguments makes the constraint monotonic or antimonotonic on each of them. In this way, they are easy to handle. We now provide a few examples of piecewise (anti)-monotonic constraints and we explain how their expressions can be transformed.

Assume that we want to extract the closed n-sets whose numbers of elements in the two first attributes are approximately the same. This approximation is tuned through a parameter $\epsilon \in \mathbb{R}^+$. The smaller ϵ is, the stronger the constraint

($\epsilon = 0$ forces $\sharp X^1 = \sharp X^2$). Here is how this constraint $\mathcal{C}_1$ can be formally defined.

$$\mathcal{C}_1(X^1, X^2) \equiv \frac{\sharp X^1}{\sharp X^2} - \frac{\sharp X^2}{\sharp X^1} \leq \epsilon \wedge \frac{\sharp X^2}{\sharp X^1} - \frac{\sharp X^1}{\sharp X^2} \leq \epsilon \wedge X^1 \neq \emptyset \wedge X^2 \neq \emptyset$$

$\mathcal{C}_1$ is neither monotonic nor antimonotonic, nor succinct (the itemsets satisfying a succinct constraint can be expressed by unions and differences of powersets of sets of all items having a nonempty support satisfying arbitrary predicates) [Ng et al. 1998]. It is not even convertible (i.e., monotonic or antimonotonic for a specific enumeration order) [Pei et al. 2001], but it is piecewise (anti)-monotonic. Intuitively, a constraint is piecewise (anti)-monotonic if it is either monotonic or antimonotonic on *every* occurrence of its arguments. For example, $\mathcal{C}_1$ has two arguments X^1 and X^2 and each of them occurs five times in the expression of the constraint. When rewritten with different arguments for each of these ten occurrences, the resulting new constraint $\mathcal{P}_{\mathcal{C}_1}$ is monotonic or antimonotonic on each of its ten arguments. Here is this constraint.

$$\begin{array}{rl} & \mathcal{P}_{\mathcal{C}_1}(P_1^1, P_2^1, P_3^1, P_4^1, P_5^1, P_6^2, P_7^2, P_8^2, P_9^2, P_{10}^2) \\ \equiv & \frac{\sharp P_1^1}{\sharp P_6^2} - \frac{\sharp P_7^2}{\sharp P_2^1} \leq \epsilon \wedge \frac{\sharp P_8^2}{\sharp P_3^1} - \frac{\sharp P_4^1}{\sharp P_9^2} \leq \epsilon \wedge P_5^1 \neq \emptyset \wedge P_{10}^2 \neq \emptyset \end{array}$$

It is easily shown that $\mathcal{P}_{\mathcal{C}_1}$ is monotonic on P_3^1, P_4^1, P_5^1, P_6^2, P_7^2, and P_{10}^2 and antimonotonic on P_1^1, P_2^1, P_8^2, and P_9^2.

Considering $\mathcal{R}_E \subseteq \{\alpha, \beta, \gamma\} \times \{1, 2, 3, 4\} \times \{A, B, C\}$, the constraint specifying that every n-set must contain a proportion of a given 2-set $\langle(\alpha, \gamma), (A, B)\rangle$ greater than 0.5 is another example of piecewise (anti)-monotonic constraint.

$$\mathcal{C}_2(X^1, X^3) \equiv \frac{\sharp(X^1 \cap \{\alpha, \gamma\}) \times \sharp(X^3 \cap \{A, B\})}{\sharp X^1 \times \sharp X^3} \geq 0.5$$

This constraint is rewritten as follows.

$$\mathcal{P}_{\mathcal{C}_2}(P_1^1, P_2^1, P_3^3, P_4^3) \equiv \frac{\sharp(P_1^1 \cap \{\alpha, \gamma\}) \times \sharp(P_3^3 \cap \{A, B\})}{\sharp P_2^1 \times \sharp P_4^3} \geq 0.5$$

Let us now define the class of piecewise (anti)-monotonic constraints.

Definition 2.5 (*Piecewise (anti)-Monotonic Constraint*). Let $\mathcal{C}$ be a constraint and $\mathcal{P}_{\mathcal{C}}$ its associated constraint, (i.e., a rewrite of $\mathcal{C}$ in which every occurrence of a same variable is replaced by a separate variable). $\mathcal{C}$ is piecewise (anti)-monotonic if $\mathcal{P}_{\mathcal{C}}$ is either monotonic or antimonotonic 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 as follows.

—$\mathcal{C}_{\mu-size}(X) \equiv \sharp X \geq \mu$ ($\mathcal{P}_{\mathcal{C}_{\mu-size}}(P) \equiv \sharp P \geq \mu$ is monotonic on P) $\mathcal{C}_{\mu-size}$ is the classical (absolute) frequency constraint in one attribute (*support attribute*).

—$\mathcal{C}_{\nu-volume}(X^1, \ldots, X^m) \equiv \prod_{i=1}^{m} \sharp X^i \geq \nu$ ($\mathcal{P}_{\mathcal{C}_{\nu-volume}}(P^1, \ldots, P^m) \equiv \prod_{i=1}^{m} \sharp P^i \geq \nu$ is monotonic on every P^i). $\mathcal{C}_{\nu-volume}$ endorses the role of restricting the collection of closed n-sets to the largest ones, thus finding significant associations.

—$\mathcal{C}_{v-avg}(X) \equiv \frac{\sum_{x\in X} Val^+(x)}{\sharp X} \geq v$, where Val^+ is a real-valued positive function. ($\mathcal{P}_{\mathcal{C}_{v-avg}}(P_1, P_2) \equiv \frac{\sum_{p\in P_1} Val^+(p)}{\sharp P_2} \geq v$ is monotonic on P_1 and antimonotonic on P_2). $\mathcal{C}_{v-avg}$ is extremely useful in many applications. For example, in a transactional data set, it enables the extraction of all the closed n-sets such that the average price of the items, in every closed n-set, is above a specified threshold.

—$\mathcal{C}_{diffval}(X) \equiv \sum_{x\in X} Val_1^+(x) - \sum_{x\in X} Val_2^+(x) \geq 0$, where Val_1^+ and Val_2^+ are real-valued positive functions ($\mathcal{P}_{\mathcal{C}_{diffval}}(P_1, P_2) \equiv \sum_{p\in P_1} Val_1^+(p) - \sum_{p\in P_2} Val_2^+(p) \geq 0$ is monotonic on P_1 and anti-monotonic on P_2). $\mathcal{C}_{diffval}$ is a very useful constraint, too. For example, in a transactional dataset, if Val_1^+ gives the prices of the items in a supermarket and Val_2^+ the prices of the same items in another supermarket, this constraint makes the extraction focus on closed n-sets, gathering items that are advantageously bought in the first supermarket.

Among the piecewise (anti)-monotonic constraints, let us discuss a promising one pertaining to pattern relevancy. For a given closed n-set, $\mathcal{C}_{closed}$ only forces the elements outside the pattern not to be connected to the inside elements. Some of them may be absent of the pattern because of one tuple absent from $\mathcal{R}$. In other terms, an element outside a closed n-set may be in relation with almost all the elements of the closed n-set. In $\mathcal{R}_E$, $\langle(\alpha, \beta, \gamma), (1), (A, B)\rangle$ is a closed 3-set. However the element 2 is "almost" identical to element 1 on $\{\alpha, \beta, \gamma\} \times \{A, B\}$ (five tuples out of six are in $\mathcal{R}_E$). To avoid the extraction of such closed n-sets, we propose a new constraint, namely $\mathcal{C}_{\delta-isolated}$. It is defined as follows.

Definition 2.6 ($\mathcal{C}_{\delta-isolated}$). An n-set $H = \langle X^1, \ldots, X^n\rangle$ is isolated with respect to the attribute X^i, denoted $\mathcal{C}_{\delta-isolated}(H, i)$, iff $\forall x \in D^i \setminus X^i$, $\sharp(K \setminus \mathcal{R}) > \delta \times \sharp K$, where $K = X^1 \times \cdots \times \{x\} \times \ldots X^n$ and $\delta \in [0, 1]$ is a user-defined parameter.

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

Example 2.7. In $\mathcal{R}_E$, let $H = \langle(\alpha, \beta), (1), (A, B, C)\rangle$. Then $\mathcal{C}_{0.4-isolated}(H, 2)$ is true whereas $\mathcal{C}_{0.5-isolated}(H, 2)$ is not because of elements 2 and 4.

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

3. THE DATA-PEELER ALGORITHM

3.1 Links with Closed-2-Set Mining

There does not seem to exist any "simple" bijection between n-ary relations and binary ones helping for the extraction of closed n-sets. Such a transformation

would certainly lead to a combinatorial explosion of the number of elements. Indeed, the attributes of the binary relation should combine several elements of the different sets to encompass the n-ary relation. Though it is trivial to transform an n-ary relation to n binary relations (give an id to every n-tuple and relate this id with each of the n elements), this leads to a multirelational data mining problem which is more general, therefore more difficult. Other attempts to reuse the knowledge on closed 2-set extraction include Representative Slice Mining and TRIAS (see Section 9). DATA-PEELER outperforms both of them by orders of magnitude. Even though DATA-PEELER does not rely on closed-2-set extractions, its underlying principles are highly similar to that of the D-MINER algorithm [Besson et al. 2005]. DATA-PEELER can even be considered as a generalization of D-MINER to n-ary relations.

3.2 Enumeration Strategy

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.

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 the n-sets $\langle X^1, \ldots, X^n\rangle$ such that $\forall i = 1 \ldots n, U^i \subseteq X^i \subseteq U^i \cup V^i$. The root node $(\langle\emptyset, \ldots, \emptyset\rangle, \langle D^1, \ldots, D^n\rangle)$ represents all possible n-sets. On the contrary, nodes such that $\forall i = 1 \ldots n, V^i = \emptyset$ represent a single n-set $\langle U^1, \ldots, U^n\rangle$. More generally, a node (U, V) represents $2^{\sum_{i=1}^{n} \sharp V^i}$ n-sets.

Example 3.1. The node $E = (U, V) = (\langle(\alpha), \emptyset, (C)\rangle, \langle(\gamma), (1, 4), (A, B)\rangle)$ represents 2^5 (i.e., 32) 3-sets. For instance, it represents $\langle(\alpha), \emptyset, (C)\rangle$, $\langle(\alpha), (4), (C)\rangle$ and $\langle(\alpha, \gamma), (1, 4), (A, B, C)\rangle$. In the contrary, it represents neither $\langle(\alpha), \emptyset, \emptyset\rangle$ (C *must* be in the 3-set) nor $\langle(\alpha, \beta, \gamma), (4), (C)\rangle$ (β *must not* be in the 3-set).

At a node $N = (U, V)$, DATA-PEELER recursively selects an element p from V (see Section 4.2 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 that contain (respectively, do not contain) p.

Example 3.2. Considering the node E of Example 3.1, the selection of element $4 \in V$ leads to the two nodes $E_L = (\langle(\alpha), (4), (C)\rangle, \langle(\gamma), (1), (A, B)\rangle)$ and $E_R = (\langle(\alpha), \emptyset, (C)\rangle, \langle(\gamma), (1), (A, B)\rangle)$ (see Figure 2).

3.3 Checking $\mathcal{C}_{connected}$

It is possible to exploit constraint $\mathcal{C}_{connected}$ to reduce the size of V_L and, as a consequence, to 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.

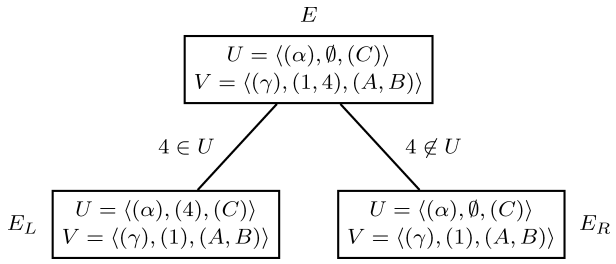


Fig. 2. Enumeration of the element $4 \in D^2$ from node E of Example 3.2.

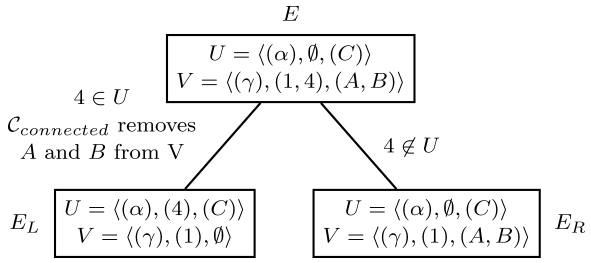


Fig. 3. Enumeration of the element $4 \in D^2$ from node E of Example 3.3.

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

—$U_L = \langle U^1, \ldots, U^j \cup \{p\}, \ldots, U^n \rangle$
—$V_L = \langle V'^1, \ldots, V'^n \rangle$ such that $\forall i = 1 \ldots n$,
$$V'^i = \begin{cases} V^j \text{ if i = j} \\ V^i \setminus \{v \in V^i | \neg \mathcal{C}_{connected}(\langle U^1, \ldots, \{p\}, \ldots, \{v\}, \ldots, U^n \rangle)\} \text{ otherwise.} \end{cases}$$
—$U_R = U$
—$V_R = \langle V^1, \ldots, V^j \setminus \{p\}, \ldots, V^n \rangle$.

U_L now contains p, meaning that p belongs to all the n-sets represented by N_L. All the elements of the $(V^i)_{i \neq j}$ that, once added to U_L, lead to unconnected n-sets, are absent from V_L. For N_R, p is simply removed from V_R. Hence, N_R represents no n-set containing p anymore. Thanks to this enumeration strategy, every n-set satisfying $\mathcal{C}_{connected}$ is browsed once and only once.

Example 3.3. In our running example, the elements A and B cannot be added to $U_L = \langle (\alpha), (4), (C) \rangle$ to form a 3-set satisfying $\mathcal{C}_{connected}$. Indeed, $(\alpha, 4, A) \notin \mathcal{R}$ and $(\alpha, 4, B) \notin \mathcal{R}$. As a consequence, $\mathcal{C}_{connected}$ removes those two elements from V^3: We finally obtain $E_L = (\langle (\alpha), (4), (C) \rangle, \langle (\gamma), (1), \emptyset \rangle)$ (see Figure 3).

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.

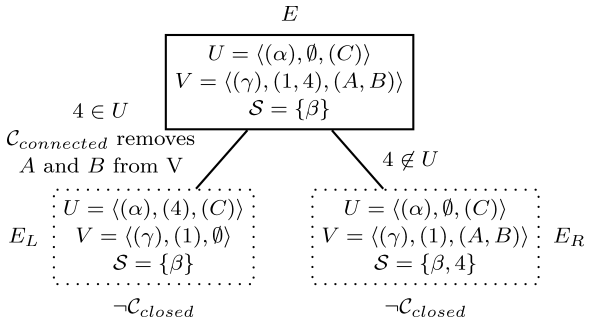


Fig. 4. Enumeration of the element $4 \in D^2$ from node E of Example 3.4.

3.4 Checking $\mathcal{C}_{closed}$

For better performance, the closeness constraint must be handled during the enumeration process (safe pruning) rather than in a postprocessing phase. Basically, if there exists an element p such that $p \in D^j \setminus (U^j \cup V^j)$ and $\mathcal{C}_{connected}(\langle U^1 \cup V^1, \ldots, \{p\}, \ldots, U^n \cup V^n\rangle)$ is satisfied, then every n-set represented by N can be extended with p to form a larger n-set satisfying $\mathcal{C}_{connected}$. In other terms, it is not closed. Therefore, N can be safely pruned. Its closure is computed and output in another part the enumeration tree (where $p \in U$).

Thanks to our enumeration strategy, we do not have to check every element of $(D^1 \times \cdots \times D^n) \setminus (U \cup V)$. Elements that have been removed from V when applying $\mathcal{C}_{connected}$ cannot be used to form any n-set connected with the elements of U. Indeed, this is the reason why they were removed from V. On the contrary, the closeness with respect to an element p that was selected and removed during the enumeration (*Children*(N, p) was called) must be checked on N_R and its descendants. A stack, denoted $\mathcal{S}$, is in charge of storing such elements. More formally, the closeness constraint is defined in the node space as follows: $\mathcal{C}_{closed}((U, V), \mathcal{S}) \equiv \forall e \in \mathcal{S}, \neg\mathcal{C}_{connected}(\langle U^1 \cup V^1, \ldots, \{e\}, \ldots, U^n \cup V^n\rangle)$.

Example 3.4. Let us refer to the running example and assume that $\mathcal{S} = \{\beta\}$ for E. Neither E_L nor E_R satisfies $\mathcal{C}_{closed}$ (see Figure 4). Indeed $\langle(\beta), (1, 4), (C)\rangle$ is connected and so is $\langle(\beta), (1), (A, B, C)\rangle$. Hence, among the 32 3-sets represented by E, none is both connected and closed.

3.5 Piecewise (anti)-Monotonic Constraints

Let us now study how DATA-PEELER can take advantage of piecewise (anti)-monotonic constraints, that is, 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 n-sets H represented by N, $\neg\mathcal{C}(H) \Leftrightarrow \neg\mathcal{M}od_{\mathcal{C}}(N)$. In other words, a node does not satisfy $\mathcal{M}od_{\mathcal{C}}$ iff none of the n-sets it represents satisfies $\mathcal{C}$.

Definition 3.5. Let $\mathcal{C}$ be a piecewise (anti)-monotonic constraint.

$$\mathcal{M}od_{\mathcal{C}}((U, V)) \equiv \mathcal{P}_{\mathcal{C}}\left(P_1^{\sigma(1)}, \ldots, P_m^{\sigma(m)}\right)$$

Input: A node $N = (U, V)$ and a stack $\mathcal{S}$
Output: Closed n-sets represented by N and satisfying $\mathcal{C}$

if $\mathcal{C}_{closed}(N, \mathcal{S}) \wedge \mathcal{M}od_{\mathcal{C}}(N)$ **then**
  if $Is\_empty(V)$ **then**
    output $U = \langle U^1, \ldots, U^n \rangle$
  else
    $p \leftarrow Select(V)$
    $(N_L, N_R) \leftarrow Children(N, p)$
    DATA-PEELER$(N_L, \mathcal{S})$
    DATA-PEELER$(N_R, \mathcal{S} \cup p)$
  end if
end if

Fig. 5. The DATA-PEELER algorithm.

where $\forall j = 1 \ldots m$, $P_j^{\sigma(j)} = U^{\sigma(j)} \cup V^{\sigma(j)}$ if $\mathcal{C}$ is monotonic on $P_j^{\sigma(j)}$ or $P_j^{\sigma(j)} = U^{\sigma(j)}$ if $\mathcal{C}$ is antimonotonic on P_j.

Example 3.6. As explained in Section 2.2, the piecewise (anti)-monotonic constraint $\mathcal{C}_2(X^1, X^3) \equiv \frac{\sharp(X^1 \cap \{\alpha,\gamma\}) \times \sharp(X^3 \cap \{A,B\})}{\sharp X^1 \times \sharp X^3} \geq 0.5$ is rewritten so that it is monotonic or (anti)-monotonic on each of its arguments: $\mathcal{P}_{\mathcal{C}_2}(P_1^1, P_2^1, P_3^3, P_4^3) \equiv \frac{\sharp(P_1^1 \cap \{\alpha,\gamma\}) \times \sharp(P_3^3 \cap \{A,B\})}{\sharp P_2^1 \times \sharp P_4^3} \geq 0.5$. Here is how it is enforced at extraction time (see Definition 3.5): $\mathcal{M}od_{\mathcal{C}_2}((U, V)) \equiv \mathcal{P}_{\mathcal{C}_2}(U^1 \cup V^1, U^1, U^3 \cup V^3, U^3)$. Thus, $\mathcal{C}_2$ does not prune the node of Example 3.1 since $\mathcal{M}od_{\mathcal{C}_2}((\langle(\alpha), \emptyset, (C)\rangle, \langle(\gamma), (1, 4), (A, B)\rangle)) \equiv \frac{\sharp(\{\alpha,\gamma\} \cap \{\alpha,\gamma\}) \times \sharp(\{A,B,C\} \cap \{A,B\})}{\sharp\{\alpha\} \times \sharp\{C\}} \geq 0.5$ is true.

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 $N_0 = (\langle \emptyset, \ldots, \emptyset \rangle, \langle D^1, \ldots, D^n \rangle)$ and an empty stack $\mathcal{S} = \emptyset$. Its major steps are presented in Figure 5. First of all, the closeness property is checked (see Section 3.4) as well as a user-defined piecewise (anti)-monotonic constraint $\mathcal{C}$ (see Section 3.5). If both are satisfied and no element remains to be enumerated, the n-set U is output. Otherwise the enumeration process splits the current node N into two new nodes N_L and N_R. To do so, an element p of V is selected (see Section 4.2) and the function $Children(N, p)$, described in Section 3.2, is called. Finally, DATA-PEELER$(N_L, \mathcal{S})$ and DATA-PEELER$(N_R, \mathcal{S} \cup \{p\})$ are recursively called. Notice 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 $\mathcal{C}_{connected}$.

4.2 Selecting the Element to be Enumerated

As explained in Section 3.2, an element $p \in V$ must be selected. Its choice determines the two nodes N_L and N_R deriving from the current one. The more elements their V n-sets contain, the greater the remaining search space. V_R

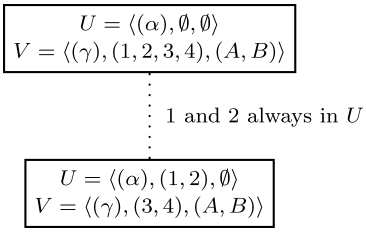


Fig. 6. Illustration of Example 4.1.

always contains $\sharp V - 1$ elements. Hence, DATA-PEELER's selection strategy for p focuses on minimizing the number of elements V_L contains, that is, it aims at maximizing the number of elements $\mathcal{C}_{connected}$ removes from the search space when p is set present.

Whenever an element is enumerated, $\mathcal{C}_{connected}$ removes some elements if: (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 $\mathcal{R}$ that are browsed when enforcing $\mathcal{C}_{connected}$ after an element from V^d is enumerated.

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

DATA-PEELER enumerates an element on the attribute domain d maximizing this formula. The chosen element in V^d is the one presenting the lowest density in $\mathcal{R}$. Indeed, the less elements are connected in $\mathcal{R}$, the more likely $\mathcal{C}_{connected}$ removes elements from V to build V_L. The experiment in Section 6.2 empirically shows that the proposed selection criterion outperforms other sensible criteria.

4.3 Optimizations

4.3.1 *Moving Elements from V to U.* Every n-set represented by a node $N = (U, V)$ is "included in" $\langle U^1 \cup V^1, \ldots, U^n \cup V^n\rangle$. As a consequence, an element of V^i which, in $\mathcal{R}$, is associated to all the elements of $\times_{j \neq i} U^j \cup V^j$ is necessarily an element of every closed n-set represented by N. It can be moved to U.

This set of elements which can be moved to U is $\{v \in V | \mathcal{C}_{connected}(\langle U^1 \cup V^1, \ldots, \{v\}, \ldots, U^n \cup V^n\rangle)\}$. It can be easily deduced from this expression that, given N, finding the elements of this set means checking the presence of at most (if all the elements of V can extend N) $\sum_{k=1}^{n} (\sharp V^k \times \prod_{l \neq k} \sharp(U^l \cup V^l))$ tuples of $\mathcal{R}$. This cost may look high. However, recall that the enumeration subtree whose root is N contains at worst (no pruning) $2^{1+\sum_{i=1}^{n} \sharp V^i} - 1$ nodes. Hence, removing elements from V as soon as possible significantly reduces the number of nodes to consider and, as a consequence, the running time of DATA-PEELER.

Example 4.1. Given the node $(U, V) = (\langle(\alpha), \emptyset, \emptyset\rangle, \langle(\gamma), (1,2,3,4), (A,B)\rangle)$ and relation $\mathcal{R}_E$, the elements 1 and 2 from D^2 are safely moved from V to U (see Figure 6). Indeed, both $\langle(\alpha, \gamma), (1), (A,B)\rangle$ and $\langle(\alpha, \gamma), (2), (A,B)\rangle$ satisfy $\mathcal{C}_{connected}$.

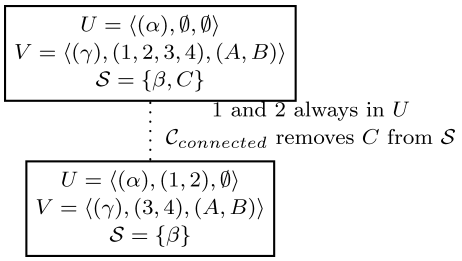


Fig. 7. Illustration of Example 4.2.

4.3.2 *Removing Elements from $\mathcal{S}$.* Every n-set represented by a node $N = (U, V)$ "contains" $\langle U^1, \ldots, U^n\rangle$. As a consequence, the elements of $\mathcal{S}$ that violate $\mathcal{C}_{connected}$ when added to $\langle U^1, \ldots, U^n\rangle$ will not enlarge any n-set represented by N. They can be removed from $\mathcal{S}$. Formally, this set of elements is $\{s \in \mathcal{S} | \neg\mathcal{C}_{connected}(\langle U^1, \ldots, \{s\}, \ldots, U^n\rangle)\}$. These elements which are safely removed from $\mathcal{S}$ are found in the following way: $\forall s \in \mathcal{S}$, whenever an element p is moved from V to U, if $\langle U^1, \ldots, \{p\}, \ldots, \{s\}, \ldots, U^n\rangle$ does not verify $\mathcal{C}_{connected}$, s is removed from $\mathcal{S}$. This process is similar to the enforcement of $\mathcal{C}_{connected}$ (see Section 3.3) but applied on $\mathcal{S}$ instead of V. This optimization speeds-up the enforcement of $\mathcal{C}_{closed}$ for all the nodes deriving from N. The gain is twofold.

—$\mathcal{S}$ containing less elements, the global cost pertaining to the enforcement of $\mathcal{C}_{closed}$ is lowered;
—when enforcing $\mathcal{C}_{closed}$, there is no need to browse $\{U^1 \times \cdots \times \{s\} \times \cdots \times U^n | s \in \mathcal{S}\}$: All these tuples are present otherwise some $s \in \mathcal{S}$ would have been removed by this optimization.

Example 4.2. Let us return to Example 4.1 and assume that $\mathcal{S} = \{\beta, C\}$. When 2 is moved from V to U, the elements of $\mathcal{S}$ are browsed. β is kept ($\langle(\beta), (2), \emptyset\rangle$ *does* verify $\mathcal{C}_{connected}$), whereas C is removed ($\langle(\alpha), (2), (C)\rangle$ *does not* verify $\mathcal{C}_{connected}$) (see Figure 7).

4.4 Example of Computation

Figure 8 depicts a part of the computation of DATA-PEELER on $\mathcal{R}_E$ where every closed 3-set satisfying $\mathcal{C}_{5-volume}$ is to be extracted. The dashed leaf is a closed 3-set satisfying $\mathcal{C}_{5-volume}$. The dotted ones are pruned. The enumeration strategy follows the rule enunciated in Section 4.2.

5. SPACE COMPLEXITY

In this section, 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 Dataset

Unlike for binary relation mining algorithms, it is not possible to store the projection (usually called "tidset") of the input dataset $\mathcal{R}$ on each element e of $D^1 \cup \cdots \cup D^n$. The use of sophisticated data structures like FP-trees [Han et al. 2000] remains an open problem because of the multiple attributes to consider

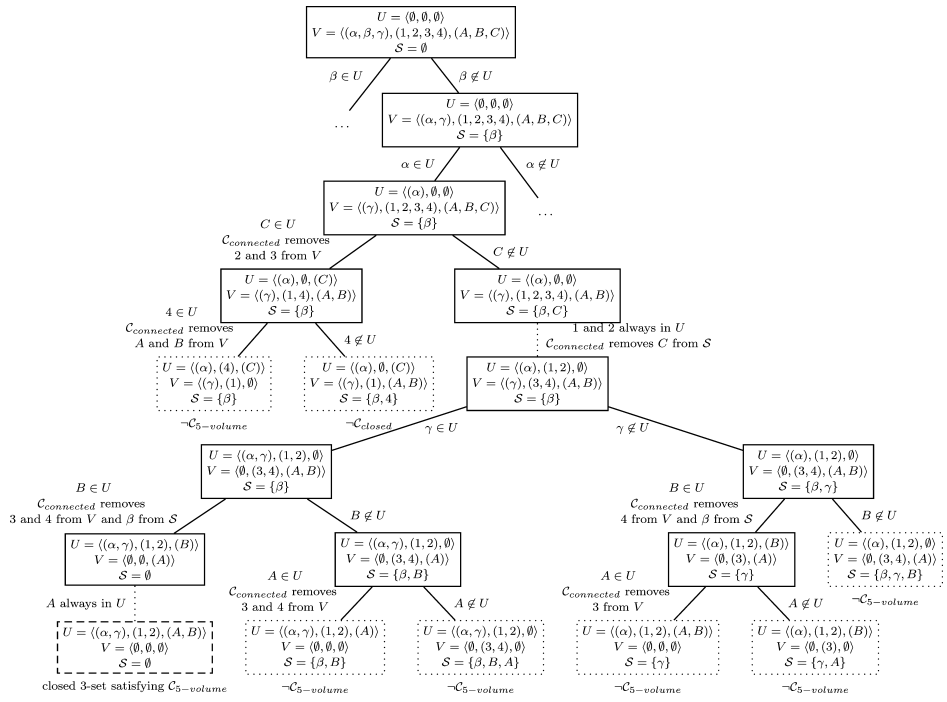


Fig. 8. Part of the computation of DATA-PEELER on $\mathcal{R}_E$.

and the required ability to enumerate any of them all along the enumeration. As a consequence, the whole dataset must be stored in main memory so that $\mathcal{C}_{connected}$ and $\mathcal{C}_{closed}$ can be enforced.

Two classes of data structures were investigated, namely a bitset-based structure and a list-based structure. In both cases, the dataset is stored in a complete prefix tree of height $n-1$ corresponding to the $n-1$ first attributes. The nodes at depth $i = 0 \ldots n-2$ always have $\sharp D^{i+1}$ children, one for every element of D^{i+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 relies 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 dataset 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 to be the smallest attribute domain minimizes the access time. If $d = \frac{\sharp \mathcal{R}}{\prod_{i=1}^{n} \sharp D^i}$ denotes the density of the dataset, 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 dataset 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 requirements as well. Therefore, this structure was chosen 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, on average, a node at depth i has less than $\frac{b}{a+b} \sharp D^{i+1}$ children. Unless the dataset is very sparse and/or nonhomogeneous, 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

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

5.3 Space Complexity

Combining the results from Section 5.1 and Section 5.2, the space complexity of DATA-PEELER is linear in the size of the input relation ($\prod_{i=1}^{n} \sharp D^i$). More precisely it is:

—$\mathcal{O}((\sharp D^1 + \sharp D^2)^2)$ if $n = 2$ (the space requirement for the V set predominates); or

—$\mathcal{O}(\prod_{i=1}^{n} \sharp D^i)$ if $n > 2$ (the space requirement for the dataset predominates).

6. EXPERIMENTAL RESULTS

Every experiment described here has been performed on a GNU/Linux system equipped with an AMD 2600 + processor and 512 Mo of RAM. DATA-PEELER is implemented in C++ and compiled with GCC 4.1.2.

6.1 Presentation of the Datasets

6.1.1 *Synthetic Datasets.* To study the behavior of DATA-PEELER and to compare it to competitors in different situations, we have used the IBM Quest data generator [Agrawal and Srikant 1994]. Various synthetic basket datasets with predefined attributes and densities have been generated. Three attributes are considered: the customers, the bought items, and the time periods (in months).

To test the scalability of DATA-PEELER with respect to the arity of the relation (the size of the input data remaining constant), three kinds of uniformly random datasets are generated.

(1) 16 attributes with 2-valued domains (Boolean attributes);
(2) 8 attributes with 4-valued domains; and
(3) 4 attributes with 16-valued domains.

In such a relation, every tuple has a given probability to be in $\mathcal{R}$. When generating a large dataset, its density $d = \frac{\sharp\mathcal{R}}{\prod_{i=1}^{n} \sharp D^i}$ is close to this probability. Datasets built in this way usually do not contain any large closed n-sets: The extraction problem is known to be hard.

6.1.2 *Real Datasets.* To assess the added-value of DATA-PEELER both in terms of the relevancy of the extracted closed n-sets and its performance with respect to competitors, we have been working on logs from the `DistroWatch.com` Web site. This popular Web site gathers comprehensive information about GNU/Linux, BSD, and Solaris distributions. Every distribution being described on a separate page, a visitor loading such a page is considered "interested" in the distribution. Every IP address contacting the server is analyzed so that the country the connection comes from is logged as well. Finally, timestamps enable to study the evolution of the interest granted to the different distributions along time. The whole dataset gathers 36 months, 243 countries, and 538 distributions. Two different datasets have been derived from it. In both cases, data has 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 have kept the elements of $\mathcal{R}$ containing this distribution and such that its normalized interest exceeds a threshold equal 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.

(1) For each node (U, V), the enumerated attribute j is chosen such that it has the smallest nonempty $\sharp V^j$. Among all the elements of V^j, the element with the smallest density in $\mathcal{R}$ is selected.

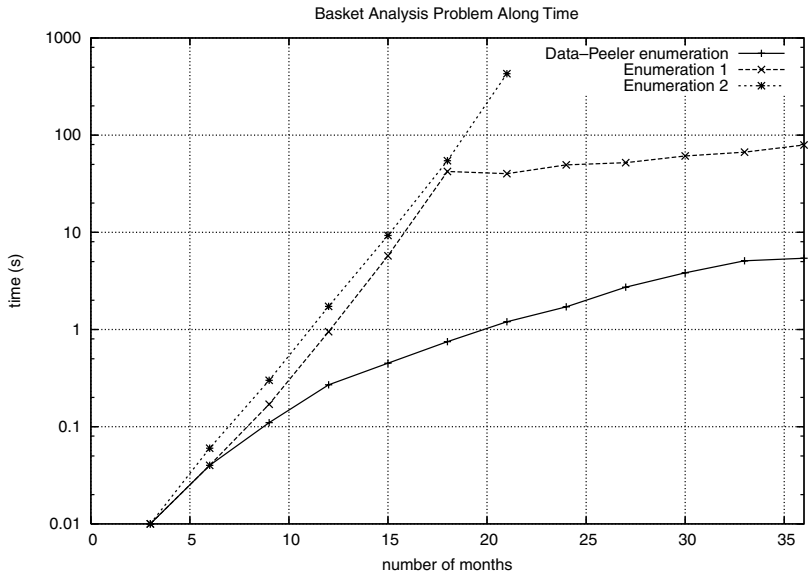


Fig. 9. Comparing DATA-PEELER enumeration with two other sensible strategies.

(2) For each node (U, V), 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. Indeed, $n-1$ attributes need to be set ($U^i \neq \emptyset$) for $\mathcal{C}_{connected}$ to, hopefully, remove elements from the last attribute.

The second strategy globally sorts the elements of the attributes all together. 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 datasets 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 on average 6 items out of 18 (density of about 33%) per month. The number of months varies from 6 to 36 and we enforce the constraint that every closed 3-set must contain at least 3 customers, 2 items, and 3 months.

Results are represented in Figure 9. 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 that is constant). 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.

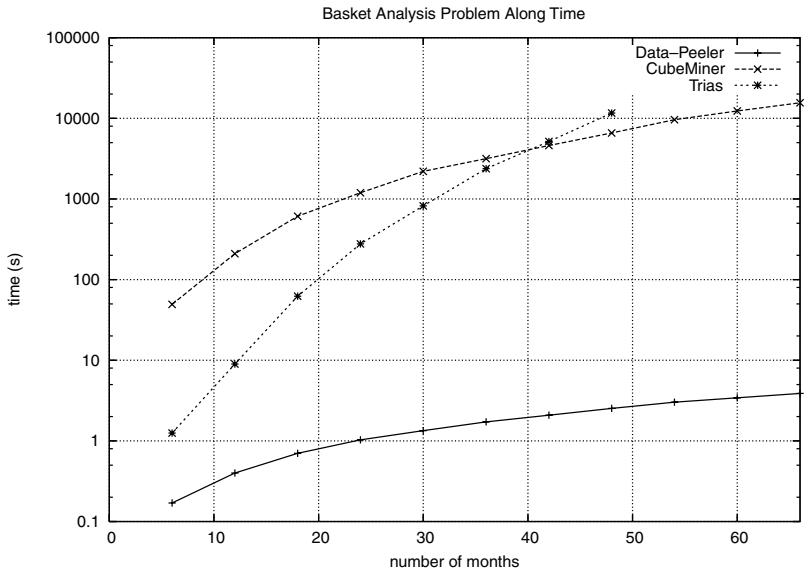


Fig. 10. Comparison with respect to CUBEMINER and TRIAS.

6.3 Comparisons with Competitors

DATA-PEELER is compared to both CUBEMINER [Ji et al. 2006] and TRIAS [Jaschke et al. 2006] on 3-ary relations. We have been using the implementations provided by their respective authors.

6.3.1 *Mining Synthetic Datasets.* Comparisons with CUBEMINER and TRIAS are achieved on synthetic QUEST-generated datasets. 144 customers buying on 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 10. 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, namely, 1 minute and 21 seconds for 431892 closed n-sets.

6.3.2 *Empirical Evaluation on a Real Dataset.* 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 datasets considered earlier, this one is relatively large. It is also much sparser since its density is 0.55%. We constrain every closed 3-set

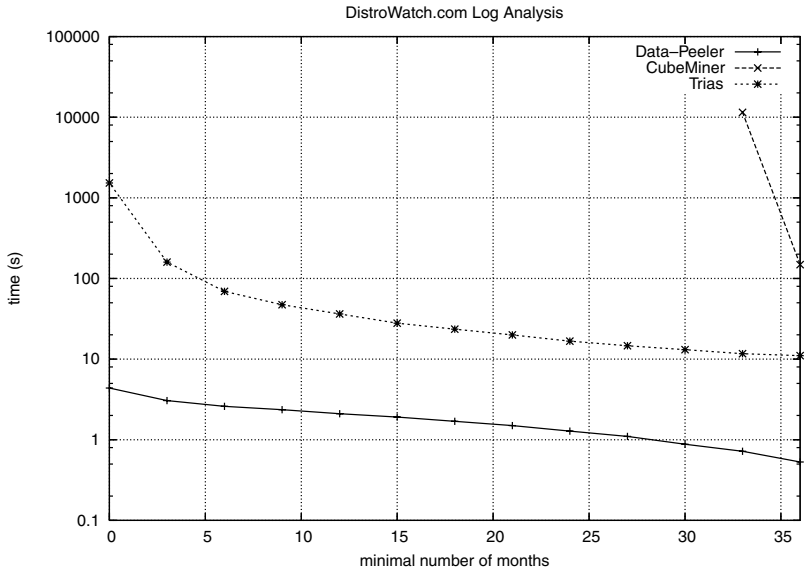


Fig. 11. Comparison with respect to CubeMiner and Trias (real data).

to involve at least 2 countries and 2 distributions and the minimal number of months a closed 3-set must contain varies from 0 to 36.

Results are represented in Figure 11. Data-Peeler outperforms its competitors by several orders of magnitude. Thanks to the small number of months in the dataset, 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 dataset. 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 cannot perform the task.

6.4 Scalability with Respect to the Arity

Here we study whether the performances of Data-Peeler are affected by the arity of the relation (versus only the size of the input data). The three kinds of uniformly random datasets (presented in Section 6.1.1) were generated with densities varying between 0 and 0.5 (given a density, the size of the input data is the same for all three datasets). The extracted closed *n*-sets are constrained to contain at least 4 tuples (this constraint depends neither on the arity of the relation nor on the size of the attribute domains). The results are plotted in Figure 12. When the datasets are sparse (e.g., a 0.05 density), a high arity has a negative impact on the performance of Data-Peeler. With greater densities the extraction times on the three datasets, are of the same order of magnitude.

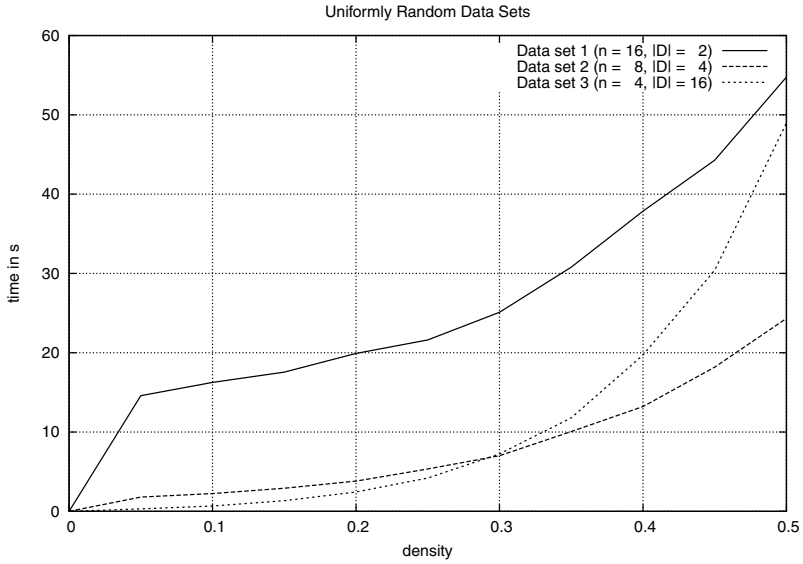


Fig. 12. Effect of the arity on the extraction times.

6.5 A Qualitative Feedback

6.5.1 *Data and Problem Setting.* An interesting 4-ary relation has been derived from the logs of `DistroWatch.com`. It takes in consideration visitors (identified by 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 that 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 antimonotonic. 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 closed n-sets will satisfy this constraint.

6.5.2 *Minimal Size and Volume Constraints.* DATA-PEELER extracts every closed 4-set from the dataset 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 a systematic interpretation of each of them.

Therefore, we enforce a volume constraint to keep only the largest patterns. Consider the new constraint $\mathcal{C}_{v\text{-}weighted\ volume} \equiv \sum_{d=1}^{n} (\sharp X^d)^{\alpha(d)} \geq v$ 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 with respect to its cardinality. The function α is such that elements from an attribute domain that is twice as small will "count" twice more in the weighted volume. $\mathcal{C}_{v\text{-}Weighted\ Volume}$ is 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 is possible to manually inspect them and assess their relevancy.

—94 closed 4-sets have on the distribution domain 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 the United Kingdom is showcased by being present in all these closed 4-sets but one.

—64 closed 4-sets have on the distribution domain 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 domain 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 domain a subset of {dyne:bolic, ArtistX, AGNULA, MoviX, GeeXboX} ∪ {Ubuntu, Damn Small, MEPIS, KNOPPIX, PCLinuxOS, Xandros}. This 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 that 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.

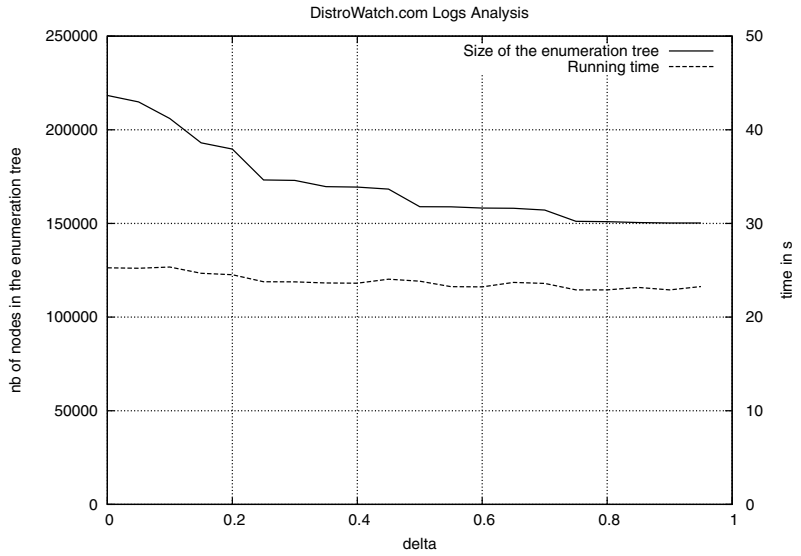


Fig. 13. Effect of $\mathcal{C}_{\delta-isolated}$ on the size of the enumeration tree and the extraction times.

6.5.3 *δ-Isolated Constraint.* Instead of searching for large closed patterns, we now focus on extracting closed 4-sets that 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 (3 distributions, 2 countries, and 2 semesters), 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 dataset). 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.

It could be expected that the $\mathcal{C}_{\delta-isolated}$ constraint dramatically reduces the extraction times, since the explored search space (i.e., the traversed enumeration tree) is smaller. However, to verify this constraint, DATA-PEELER needs, at every node, to browse the parts of the relations related to every element outside $U \cup V$. More precisely, it checks whether the proportion of "0" values in the projection of $U \cup V$ on every element outside $U \cup V$ exceeds δ. This additional cost almost compensates for the gain granted by the reduction of the search space. Figure 13 depicts both the size of the enumeration tree and the extraction times when δ (on the country attribute) varies.

7. ROBUSTNESS WITH RESPECT TO BINARIZATION

Many relations are built from binarized numerical datasets. This is the case of our ternary relation based on DistroWatch.com's logs (Section 6.1.2): Every tuple binding a month, a country, and a distribution is numerically valued by how many times the Web page related to the distribution was downloaded by visitors from the country during the month. To decide which tuples are eventually elements of $\mathcal{R}$, several binarization methods exist and/or, for a given method, various parameter settings are possible.

Gene expression data analysis is one of the promising application domains that has motivated the design of closed pattern mining algorithms (see, e.g., Besson et al. [2005], Pan et al. [2003], Jiang et al. [2004], and Zhao and Zaki [2005]). In this typical setting, the gene expression value is a number that has to be transformed into truth values for Boolean properties (e.g., a given gene is inhibited or not in a given experiment). Even though various methods have been proposed to capture relevant Boolean gene expression [Pensa and Boulicaut 2005], we do not know what is the best method or, using a method, which parameter setting is correct for capturing relevant patterns. Since various settings give rise to different datasets and thus different collections of patterns, it makes sense to study binarization assessment by means of the expansion of the relation with one more attribute that encodes the different values for different binarization procedures.

7.1 Examples of Binarization Methods

Many different binarization methods exist. Taking for example the binarization of DistroWatch.com's logs, a few of them are presented here.

7.1.1 *Per-Distribution Binarization.* In Section 6.1.2, the DistroWatch.com's logs are first normalized: For any pair of month and country, a coefficient is applied such that the sum of the values along the distributions is constant. Then, for each distribution, a tuple is kept if its normalized value reaches at least $a_{distribution} = 1/4$ of the greatest normalized value *for this distribution*. Hence, whatever the distribution, some tuples involving it are kept in the relation. They correspond to the months and countries (simultaneously) where this distribution has been granted the most attention. With this binarization method, the popularity of the distribution does not play any role.

7.1.2 *Global Binarization.* Another binarization method starts with the same normalization, but then consists in keeping the tuples having a normalized value greater than $b_{distribution} = 1/16$ of the maximal one *in the whole dataset*. In this way, the most popular distributions will be well represented in $\mathcal{R}$, whereas obscure ones may be totally absent. With this binarization method, the popularity of a distribution plays an essential role.

7.1.3 *Focusing on Another Attribute.* In these two binarization methods, the attributes "months" and "countries" play the same role, whereas the attribute "distributions" is particular. If the purpose of the mining task focuses on the adoption of Free operating systems along time (respectively space), the

Table I. Binarization Parameters

Attributes	$a_{attribute}$ (per-value coeff.)	$b_{attribute}$ (global coeff.)
"Months"	3/4	1/2
"Countries"	1/4	1/16
"Distributions"	1/4	1/16

same two binarization methods can be used with months (respectively, countries) being the particular attribute. In this way, we have six sensible binarization methods providing six different relations to mine. Obviously, many others exist.

7.2 Binarization as an Additional Attribute

If the pattern extraction aims at picking the most relevant information in a dataset where no specific question is to be answered, the binarization can be considered as a bias. Indeed, as illustrated in the previous paragraphs, different binarization methods provide different perspectives on the data. Focusing on one perspective may hide relevant patterns which would be present with many others. In the opposite, direction a particular binarization may bring out some patterns which would not appear in any other. The most interesting patterns are (at least partially) present in several relations obtained with various binarization methods.

Given a numerical n-ary relation, gathering different binarizations as the values of a new attribute provides an $(n + 1)$-ary relation: Every tuple is tagged with the binarization methods selecting it. Extracting closed $(n+1)$-sets, whose number of values in the binarization attribute exceeds a given size, provides patterns showing a robustness to the binarization method. Beyond "real" n-ary relations, this original approach emphasizes how useful multidimensional patterns are.

7.3 Experimental Results

Experiments were made on the `DistroWatch.com`'s logs binarized with the six different methods mentioned in Section 7.1. The coefficients $a_{attribute}$ and $b_{attribute}$ must be wisely chosen. Indeed, if a binarization is included in another one, it is a simple restriction rather than a real different perspective on the data. As a consequence, given an attribute D^i, we must have $a_{D^i} > b_{D^i}$. Furthermore, when an attribute is rather homogeneous with respect to the others, the applied coefficients must be larger (in our case, the number of connections along time does not vary as much as along any other attribute). The chosen coefficients are listed in Table I. The obtained 4-ary relation contains 132528 tuples out of 25681968 possible ($d = 0.5\%$).

We made DATA-PEELER extract the closed 4-sets present in it. They are only constrained to run along a minimal number of binarizations varying from 3 (presence in at least half of the binarizations) to 6 (presence in every binarization). Both the extraction times and the number of extracted 4-sets are listed in Table II.

Table II. Extracting Robust Patterns in `Distrowatch.com`'s Logs

Minimal nb of binarizations	Nb of closed 4-sets	Time (in s)
3	3580	18.93
4	1297	15.82
5	229	11.62
6	2	8.58

8. TILING AN *n*-ARY RELATION $\mathcal{R}$

8.1 Problem Setting

Usually, a relation is defined as a set of tuples. For instance, in Section 2, $\mathcal{R}_E$ is expressed in this way. Storing and retrieving a relation by listing all of its tuples one by one is both time and space consuming. Hence, minimizing the expression of an arbitrary n-ary relation $\mathcal{R}$ is an interesting problem. Since a collection of tuples is not a satisfactory solution, we may look for relevant collections of patterns that, in this context, are generically called *tiles*. The *tiling task* consists in finding a collection of (possibly overlapping) tiles that is as compact as possible but still entirely expresses $\mathcal{R}$(i.e., the union of the tuples in all tiles equals $\mathcal{R}$).

Choosing the tiles to be closed n-sets looks like a clever idea. Indeed, a closed n-set can be seen as a syntactical summary of a part of $\mathcal{R}$, since it is shorter to write a closed n-set than listing all the tuples it covers. For example, without any loss of information, we can write that $\mathcal{R}_E$ contains $\langle(\beta, \gamma), (1, 2, 4), (A)\rangle$ instead of listing all the tuples this closed 3-set covers.

$$(\beta, 1, A), (\beta, 2, A), (\beta, 4, A), (\gamma, 1, A), (\gamma, 2, A), (\gamma, 4, A)$$

A collection of a few, *well-chosen*, closed 3-sets shortly expresses the whole relation $\mathcal{R}_E$.

$$\begin{gathered}\langle(\alpha, \gamma), (1, 2), (A, B)\rangle\\ \langle(\beta, \gamma), (3, 4), (C)\rangle\\ \langle(\alpha, \beta), (1), (A, B, C)\rangle\\ \langle(\gamma), (1, 2, 4), (A, B)\rangle\\ \langle(\beta), (1, 2, 4), (A, C)\rangle\\ \langle(\alpha, \beta, \gamma), (4), (C)\rangle\\ \langle(\alpha), (1, 2, 3), (B)\rangle\end{gathered}$$

By definition, closed n-sets satisfy both $\mathcal{C}_{connected}$ and $\mathcal{C}_{closed}$. However, for the sake of minimizing the expression of $\mathcal{R}$, constraining the tiles to be closed does not make sense. Indeed, in some situations, when two (or more) closed n-sets are overlapping, one of them can be "cropped" so that the relation is expressed in a shorter way. For example, in the preceding collection, $\langle(\beta), (1, 2, 4), (A, C)\rangle$ can be "cropped" in $\langle(\beta), (2, 4), (A, C)\rangle$ if $\langle(\alpha, \beta), (1), (A, B, C)\rangle$ is kept (unaltered) in the collection. Indeed, this closed 3-set already covers $\{\beta\} \times \{1\} \times \{A, C\}$.

Hence, the tiles are advantageously chosen among the n-sets rather than the closed n-sets. The n-set domain is a superset of the closed n-sets domain, where $\mathcal{C}_{connected}$ still needs to be checked but where $\mathcal{C}_{closed}$ does not necessarily hold.

With tiles from this larger pattern domain, the previous tiling of $\mathcal{R}_E$ can be improved into the following one.

$$\begin{gathered}\langle(\alpha,\gamma),(1,2),(A,B)\rangle\\ \langle(\beta,\gamma),(3,4),(C)\rangle\\ \langle(\alpha,\beta),(1),(A,B,C)\rangle\\ \langle(\gamma),(1,2,4),(A,B)\rangle\\ \langle(\beta),(2,4),(A)\rangle\\ \langle(\alpha),(4),(C)\rangle\\ \langle(\alpha),(3),(B)\rangle\end{gathered}$$

Interestingly, the attributes $(A^i)_{i=1\ldots n}$ can be seen as variables of a multivalued logic function whose truth table is given by $\mathcal{R}$. Notice that Boolean functions are a specialization of this framework ($\forall i = 1\ldots n, \sharp D^i = 2$). In this perspective, tiling $\mathcal{R}$ is equivalent to simplifying the related multivalued logic function.

For example, the tiling of $\mathcal{R}_E$ written before provides this simplified expression of the related multivalued logic function.

$$\begin{aligned}&(A^1=\alpha\vee A^1=\gamma)\wedge(A^2=1\vee A^2=2)\wedge(A^3=A\vee A^3=B)\\ \vee\ &(A^1=\beta\vee A^1=\gamma)\wedge(A^2=3\vee A^2=4)\wedge(A^3=C)\\ \vee\ &(A^1=\alpha\vee A^1=\beta)\wedge(A^2=1)\wedge(A^3=A\vee A^3=B\vee A^3=C)\\ \vee\ &(A^1=\gamma)\wedge(A^2=1\vee A^2=2\vee A^2=4)\wedge(A^3=A\vee A^3=B)\\ \vee\ &(A^1=\beta)\wedge(A^2=2\vee A^2=4)\wedge(A^3=A)\\ \vee\ &(A^1=\alpha)\wedge(A^2=4)\wedge(A^3=C)\\ \vee\ &(A^1=\alpha)\wedge(A^2=3)\wedge(A^3=B)\end{aligned}$$

Given a tiling, its quality can be measured with the number of logic operators ($\vee$ and $\wedge$) in the simplified expression of the related multivalued logic function, the smaller the better. For example, 18 $\vee$ and 14 $\wedge$ are present in our simplified expression of the multivalued logic function related to $\mathcal{R}_E$. The quality of this tiling is $18 + 14 = 32$.

8.2 Postprocessing DATA-PEELER's Output

The set of all closed n-sets returned by DATA-PEELER (without enforcing any constraint) is a tiling of the relation, since it is integrally covered. Its quality is very poor, though: Most of the time, it is far worse than listing every tuple one by one. Nevertheless, we consider the closed n-sets as a starting point. Postprocessing DATA-PEELER will take care of removing useless information from the computed closed n-sets to obtain a tiling of $\mathcal{R}$ with a good quality.

8.2.1 *Removing the Complete Sets.* Given a tile $\langle X^1, X^2, \ldots, X^n\rangle$, when one of the X^i equals D^i, listing its elements one by one is useless. For example, in $\mathcal{R}_E$, $\langle(\alpha,\beta),(1),(A,B,C)\rangle$ can be shortened into $\langle(\alpha,\beta),(1),-\rangle$ meaning that, whatever the value v of the last attribute, the tuples $\langle(\alpha),(1),(v)\rangle$ and $\langle(\beta),(1),(v)\rangle$ are in $\mathcal{R}_E$. In this way, when a tile has its set $X^i = D^i$, the number of required logic operators to express it is lowered (i.e., the quality is improved) by $\sharp D^i$. Let us analyze the previous example using the multivalued logic form.

$(A^1 = \alpha \vee A^1 = \beta) \wedge (A^2 = 1) \wedge (A^3 = A \vee A^3 = B \vee A^3 = C)$ requires 5 logic operators to be written. Once turned into $(A^1 = \alpha \vee A^1 = \beta) \wedge (A^2 = 1)$, only $5 - \sharp D^3 = 2$ logic operators are needed.

8.2.2 *Tightening the Tiles.* Verifying whether all the tuples related to one element of one set of a tile were already covered by the previously output tiles is straightforward. Here is how it is achieved: Whenever a tile is output, the tuples it covers are removed from the relation. In this way, a tile $\langle(\beta, \gamma), (1, 2, 4), (A)\rangle$ can be tightened into $\langle(\beta), (2, 4), (A)\rangle$ if $\langle(\alpha, \beta), (1), -\rangle$ and $\langle(\gamma), (1, 2, 4), (A, B)\rangle$ were previously output. Indeed, they already cover both $\langle(\beta, \gamma), (1), (A)\rangle$ (slice related to element 1) and $\langle(\gamma), (1, 2, 4), (A)\rangle$ (slice related to element γ). The quantity of information safely removed in this way greatly depends on the order in which the tiles are processed.

8.2.3 *Ordering the Tiles.* Relying on the order in which *n*-sets are discovered by DATA-PEELER does not provide a good tiling of $\mathcal{R}$. Advantageously, we replace it by the following heuristic.

Heuristic 8.1. Output first the tile presenting the best ratio between the number of newly covered (i.e., not covered by previously output tiles) tuples and the number of logic operators ($\vee$ and $\wedge$) needed to express it.

To do so, the closed *n*-sets are stored in main memory instead of being directly output. Whenever a tile is output, the part of $\mathcal{R}$ it covers is removed. Thus, a large tile may be moved towards the end of the sequence (and even never be output) if there are larger tiles covering many of its tuples. The algorithm terminates when $\mathcal{R}$ is completely covered.

Notice that the sequence of remaining tiles is not maintained ordered at any time. Instead, only the first tile is considered. If the quantity of tuples it covers (initialized at extraction time) has decreased, it is moved down the sequence. Otherwise, it is output.

8.2.4 *Don't-Care Set. Don't-care set* is the name given to a set $\mathcal{R}_{\text{DCS}}$ of tuples that can either be considered as elements of $\mathcal{R}$ or not. Typically they are impossible combinations of values for the *n* attributes. The don't-care set plays an interesting role in the tiling problem: Its elements can enable bigger tiles even though they are not required to be part of a tile. Tiling with DATA-PEELER easily takes advantage of a don't-care set. The closed *n*-sets are extracted on $\mathcal{R} \cup \mathcal{R}_{\text{DCS}}$. Then the tuples of $\mathcal{R}_{\text{DCS}}$ are removed (i.e., they are considered covered) and the postprocess, detailed in this section, is performed unchanged.

8.3 Experimental Results

Here again, the experiments have been performed on a GNU/Linux system equipped with a AMD 2600+ processor and 512Mo of RAM. The performance in tiling a multivalued logic function is evaluated with respect to:

—the time it takes to tile (extraction of the closed *n*-sets included),
—the number of logic operators ($\vee$ and $\wedge$) in the tiling.

Table III. Tiling of Random Multivalued Logic Functions

(a) 6.25% density

	Time performances in s			Quality in number of $\vee$ and $\wedge$		
Data set	E-MV	D-P	Var. rate	E-MV	D-P	Var. rate
1	2.21	10.62	+380.54%	46068	46518	+0.97%
2	146.88	1.95	−98.67%	23662	23166	−2.09%
3	103.62	0.52	−99.49%	10734	10035	−6.51%

(b) 25% density

	Time performances in s			Quality in number of $\vee$ and $\wedge$		
Data set	E-MV	D-P	Var. rate	E-MV	D-P	Var. rate
1	36.09	19.75	−45.27%	107869	109722	+1.71%
2	957.75	6.34	−99.33%	69460	57014	−17.91%
3	552.11	4.39	−99.20%	28118	23082	−17.91%

(c) 50% density

	Time performances in s			Quality in number of $\vee$ and $\wedge$		
Data set	E-MV	D-P	Var. rate	E-MV	D-P	Var. rate
1	122.21	48.81	−60.06%	120170	122452	+1.89%
2	1781.62	25.31	−98.57%	99763	66512	−33.32%
3	1064.33	50.81	−95.22%	45060	27886	−38.11%

8.3.1 *Comparision with* ESPRESSO-MV. An implementation of the ESPRESSO-MV algorithm [Rudell and Sangiovanni-Vincentelli 1985] (shipped with the MVSIS 3.0 package [Gao et al. 2000] for logic synthesis and verification) was used as a reference in our tests. ESPRESSO-MV is a generalization to multivalued functions of the ESPRESSO-II algorithm [Brayton et al. 1984], which only simplifies Boolean functions. It is widely used in programmable logic devices (electronic components building reconfigurable digital circuits).

DATA-PEELER and ESPRESSO-MV are compared on the uniformly random datasets presented in Section 6.1.1. We recall here the dimensions of these datasets:

(1) 16 attributes with 2-valued domains (Boolean attributes);
(2) 8 attributes with 4-valued domains; and
(3) 4 attributes with 16-valued domains.

Typical results (no significant variation from one random generation to another) are listed in Table III for three different densities ($d = 6.25\%$, $d = 25\%$, and $d = 50\%$).

8.3.2 *Discussion. Boolean attributes.* Although its simplification is slightly (between 0 and 3%) worse than ESPRESSO-MV's, DATA-PEELER shows good performances, especially on dense datasets. Above a 15% density, it performs faster than ESPRESSO-MV.

Multivalued attributes. When the attributes take more than 2 values, DATA-PEELER significantly outperforms ESPRESSO-MV both in quality and running time. The gain in quality grows with the number of values per attribute and the density of the dataset. With a 50% density, the tilings obtained from DATA-PEELER are more than one-third smaller than what ESPRESSO-MV achieves. The

gain in time is impressive: DATA-PEELER performs the task in about 1% of the time required by ESPRESSO-MV.

8.4 Using Constraints

The ability to enforce constraints allows to focus on the extraction of the best closed n-sets for tiling purposes. If the constraints are piecewise (anti)-monotonic, they can be handled along the extraction phase, tiling is more quickly achieved, and the memory requirements are lowered. Since the constrained closed n-sets may not totally cover $\mathcal{R}$, the collection of tiles is completed by a linear procedure browsing the uncovered dataset and outputting aggregates of tuples along the largest attribute domain. Of course, the quality of the tiling gradually decreases with the stringiness of the constraints until no closed n-set satisfies them. In this extreme case, the tiling is the collection of aggregates of tuples along the largest attribute domain.

The more natural constraint to enforce when tiling $\mathcal{R}$ is the one related to the order in which the tiles are stored: The ratio between the area of a closed n-set and the number of logic operators ($\vee$ and $\wedge$) needed to express it must exceed a given threshold k. Expressed formally, the constraint is

$$\mathcal{C}_{k-summary} \equiv \prod_{i=1}^{n} \sharp X^i \geq k \sum_{i=1}^{n} f(X^i) \text{ where } f(X^i) = \begin{cases} 0 \text{ if } X^i = D^i \\ \sharp X^i \text{ otherwise} \end{cases}.$$

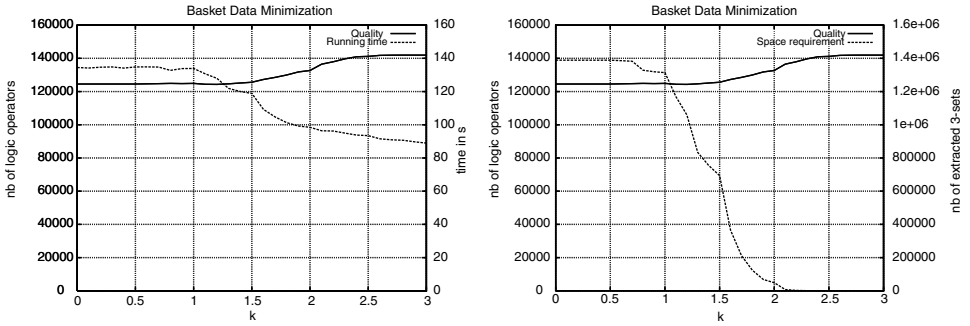
This constraint is piecewise (anti)-monotonic. In $\mathcal{R}_E$, the six closed 3-sets satisfying $\mathcal{C}_{1.5-summary} \wedge \mathcal{C}_{1-area}$ cover 17 tuples out of 23. They are completed with aggregates of the remaining tuples along D^2 (the largest domain). Once the steps detailed in Section 8.2 are applied, the tiling is

$$\begin{gathered} \langle(\alpha, \beta), (1), -\rangle \\ \langle -, (1, 2), (A)\rangle \\ \langle(\gamma), -, (A)\rangle \\ \langle -, (4), (C)\rangle \\ \langle(\gamma), (1, 2, 4), (B)\rangle \\ \langle(\alpha), (2, 3), (B)\rangle \\ \langle(\gamma), (3), (C)\rangle \\ \langle(\beta), (3), (C)\rangle \\ \langle(\beta), (4), (A)\rangle. \end{gathered}$$

The four first tiles come from the extracted closed 3-sets. The five last tiles are aggregates of tuples along D^2. In this example, two closed 3-sets do not generate any tile (see Section 8.2.3).

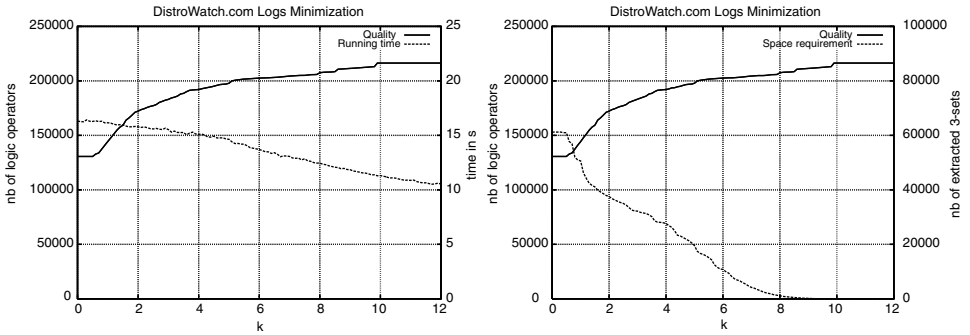
On particular datasets, other constraints can be useful. For example, to tile the graph dataset derived from `DistroWatch.com` logs, we can restrict ourselves to tiles having identical sets in the two "distributions" attributes. In this way, the tiling is performed very quickly. In the shorter expression of the dataset, one of the two distributions attribute can be omitted (since identical to the other one).

A synthetic QUEST-generated dataset (144 customers buying in average 6 items out of 72 during 144 months) and the ternary relation derived from the



(a) Time (b) Space

Fig. 14. Tiling under $\mathcal{C}_{k-summary}$ a QUEST-generated dataset.



(a) Time (b) Space

Fig. 15. Tiling under $\mathcal{C}_{k-summary}$ `DistroWatch.com`'s logs.

logs of `DistroWatch.com` are now tiled under $\mathcal{C}_{k-summary} \wedge \mathcal{C}_{1-volume}$ constraints. The running time (extraction included) and the space requirements (estimated by the number of extracted closed 3-sets) are plotted in Figures 14 and 15.

The QUEST-generated dataset is not very prone to being tiled. In other terms, the 3-sets it contains do not make good summaries; Indeed, none of them satisfy $\mathcal{C}_{2.8-summary} \wedge \mathcal{C}_{1-volume}$. As a consequence the quality of the tiling of the QUEST-generated dataset is not much altered when k increases. Indeed, when no 3-set is extracted, the tiling (the conjunction of every aggregate of tuples along the largest attribute domain) is less than 14% bigger. Thus, enforcing $\mathcal{C}_{1.5-summary}$ divides by two the space requirements against a minor alteration of the quality of the tiling (+0.84% logic operators).

Although the ternary relation derived from the `DistroWatch.com`'s logs is very sparse ($d = 0.55\%$), it contains many 3-sets that are well suited for tiling purposes. For example, 32572 3-sets satisfy $\mathcal{C}_{2.8-summary} \wedge \mathcal{C}_{1-volume}$. Indeed, when no 3-set is extracted, the tiling has about 66% more logic operators. The quality of the tiling significantly decreases as soon as k begins moving away from 0.

9. RELATED WORK

We do not consider that pattern discovery in binary relations (e.g., mining closed 2-sets) has to be discussed here. Many surveys are available (see, e.g., Goethals and Zaki [2004]) and so are excellent algorithms (see, e.g., Stumme et al. [2002], Zaki and Hsiao [2002], Pei et al. [2000], Wang et al. [2003], Uno et al. [2005], Grahne and Zhu [2005], and Besson et al. [2005]). Therefore, we discuss related work on n-ary relation data mining where $n \geq 3$. A couple of techniques for n-dimensional real-valued matrix analysis are considered as well.

9.1 Preliminary Version

First of all, it must be noticed that DATA-PEELER has been introduced in Cerf et al. [2008]. In this extended article, the way DATA-PEELER enumerates the elements is clearer (more illustrations) and the new optimizations, presented in Section 4.3, significantly improve the time performance.

In the applicative domain, DATA-PEELER provides original solutions to interesting problems which are not necessarily in the direct vicinity of closed n-set mining. The extended article focuses on two of them, namely the robustness with respect to binarization and the tiling, whereas the preliminary version settles for direct applications only.

9.2 Real-Valued Matrices

Considering kinetic microarray data, Jiang et al. [2004] propose an ad hoc multistep algorithm to compute maximal sets of genes that 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. Correlations above a user-defined threshold are captured. 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 gene. Finally, for each sample set, the corresponding maximal gene set is computed; that is, 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, Jiang et al. consider the time attribute 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 application, Zhao and Zaki [2005] propose to mine different types of patterns in the ternary relation on "genes" $\times$ "samples" $\times$ "timestamps." First of all, a range multigraph is computed for each time slice. Vertices stand for biological samples and an edge binds two gene sets presenting a similar expression ratio. On this graph, maximal cliques are computed and postprocessed. In this way, biclusters are obtained. They associate sets of samples (vertices) with sets of genes that are obtained by intersection of the sets bound to the edges. Finally, a new multigraph is computed where timestamps are vertices and pairs of highly overlapping biclusters (gene set, samples) form edges between these vertices. The so-called, *triclusters* are obtained by extracting the maximal cliques in this graph. By constructing a ternary relation on

"genes" × "samples" × "timestamps," DATA PEELER computes similar patterns in a single step.

Sun et al. [2006] propose to extend Principal Component Analysis (PCA) to sequences of tensors (data cubes with $M \geq 3$ attribute domains). Each tensor gathers the measurements for one timestep. 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 may partly cancel each other.

9.3 Mining Multirelational Data

Any n-ary relation can be turned into n binary relations. Indeed, every n-tuple in the relation can be given an id that each of the elements in the tuple is related to. This leads to a multirelational data mining problem which is more general, therefore more difficult, than the particular case we tackle with DATA-PEELER.

In this very general framework, new problems arise as how to preserve the (anti)-monotonic properties of the constraints and how to define and check pattern closeness. Afrati et al. [2005] propose different algorithms and conditions under which the a priori framework can be used while preserving the monotonicity properties. Garriga et al. [2007] propose, within an inductive logic programming framework, a clever way to unify several pattern mining problems, such as, itemset and graph mining. Intuitively, patterns are sets of elements connected by sets of relations and closed with respect to these connections. Using two popular subsumptions and two interpretations, the authors propose different algorithms to deal with multirelational datasets.

9.4 Logic Minimization

DATA-PEELER extracts patterns from an n-ary relation between finite sets. Considering these sets as the domains of multivalued variables, the relation can be seen as the truth table of a multivalued logic function with {0, 1} as a range. 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* [Karnaugh 1953] is a tool to simplify 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 becomes impractical 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) tiling 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 simplification of the original Boolean function. Later, the *QuineMcCluskey algorithm* [McCluskey 1956] 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 EPRESSO algorithm [Brayton et al. 1984] 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 programmable logic devices. ESPRESSO was adapted to deal with multivalued logic functions as well. Called ESPRESSO-MV [Rudell and Sangiovanni-Vincentelli 1985], this algorithm addresses the same task as DATA-PEELER postprocessed for tiling.

Hence, DATA-PEELER can be seen as a prime implicant extractor for multivalued logic functions. Section 8 of this article details how postprocessing these patterns can tile/minimize multivalued logic functions. Apart from gains in both quality and running time (compared to ESPRESSO-MV), the ability to push constraints during the extraction makes DATA-PEELER differ from the classical logic minimization algorithms presented here.

9.5 Closed 3-Set Mining

Considering ternary relations, DATA-PEELER faces two direct competitors: CUBEMINER [Ji et al. 2006] and TRIAS [Jaschke et al. 2006]. As detailed in Section 6.3, our algorithm outperforms these state-of-the-art algorithms by orders of magnitude.

Ji et al. [2006] propose two algorithms to extract closed 3-sets from ternary relations. The first one, called *representative slice mining*, consists in enumerating all subsets of the smallest attribute domain. For each of them, the related binary relation is computed by bitwise AND operations between elements of this subset. Then, any closed 2-set extractor can be used on each of these relations. A postprocessing phase removes the 3-sets that are not closed.

The second one, called CUBEMINER, directly operates on the ternary relation. It consists in using the cubes $X \times Y \times Z$, called *cutters*, presenting the following particularity: None of their tuples is in relation. Thus, the authors generalize the notion of *cutter* introduced in Besson et al. [2005] for closed 2-set mining. CUBEMINER first considers the whole ternary relation as a candidate. Along a depth-first enumeration, the cutters are recursively applied to generate 3 candidate children containing less tuples absent from the relation than the parent: a first one without the elements of X, a second one without the elements of Y, and a third one without the elements of Z. For each candidate, several checks are required to ensure its closeness and unicity. For a child candidate to be unique, its newly removed elements must not be included in a cutter previously applied on this branch. To verify this, every formerly applied cutter is intersected with the current one. For a child candidate to be closed, the elements of these formerly applied cutters should not extend it. Hence, every candidate is twice compared to the formerly applied cutters. By contrast, the enumeration process of DATA-PEELER ensures, by itself, the unicity of all nodes. We believe that the additional checks ensuring the unicity (whose cost per node grows linearly with the height of the enumeration tree), the absence of a clever enumeration

strategy (like that of Section 4.2) generating smaller enumeration trees, and the high cost required to check the closeness (at every node, intersections with all cutters are performed) mainly explain the differences in extraction times between CubeMiner and Data-Peeler.

Trias [Jaschke et al. 2006] also extracts closed 3-sets from ternary relations. It basically relies on closed 2-set extractions from two binary relations. From a relation on $D^1 \times D^2 \times D^3$, Trias first constructs a new binary relation on $D^1 \times (D^2 \times D^3)$ whose columns correspond to couples of elements of D^2 and D^3. Every closed 2-set $\langle A, B \rangle$, extracted from this relation, is such that B contains couples of $D^2 \times D^3$ 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 extracts every closed 2-set from the relation generated from B and checks its closeness with respect to D^1. This verification is easily carried out: Its closure must be A. Nevertheless, the larger D^1 is, the more elements are, on average, in $D^1 \setminus A$, and the more unclosed 3-sets are generated before being discarded. We believe that this is why Trias suffers a lot from large smallest attribute domains. In contrast, Data-Peeler prunes early the search spaces containing unclosed 3-sets.

10. CONCLUSION

We have proposed a new correct and complete algorithm called Data-Peeler and have defined the class of constraints it can handle at extraction time. From an 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, often known as *formal concept mining*. Indeed, numerous proposals have tackled the extraction 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 shows 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 the considered mining task. In this application, we have interpreted the relevancy of the closed n-sets Data-Peeler extracts under various piecewise (anti)-monotonic constraints. In particular, the introduced $\mathcal{C}_{\delta-isolated}$ constraint has been proved useful. Furthermore, we have shown how expanding a relation with an additional attribute can help in enforcing rather abstract (though very useful) constraints such as the robustness with respect to binarization.

Postprocessing the closed n-sets returned by Data-Peeler can also bear fruit. In this article, we have detailed how tiling can be achieved in this way. According to our experiments, the quality of Data-Peeler's tilings outperforms Espresso-MV's when dealing with "real" multivalued functions (as opposed to Boolean functions). Furthermore, our algorithm performs much faster.

We look forward to experimenting with Data-Peeler in different applicative domains. So far, only basket data analysis and Web usage mining scenarios have been considered. We want to apply Data-Peeler to kinetic gene expression data

and dynamic graphs. On the theoretical side, mining fault-tolerant patterns in *n*-ary relations appears as an interesting challenge.

ACKNOWLEDGMENTS

We would like to thank the authors of CubeMiner and Trias for providing the implementations of their algorithms and L. Bodnar for sharing with us the `DistroWatch.com`'s logs.

REFERENCES

Afrati, F., Das, G., Gionis, A., Mannila, H., Mielikainen, T., and Tsaparas, P. 2005. Mining chains of relations. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM'05)*. IEEE Computer Society, 553–556.

Agrawal, R., Imielinski, T., and Swami, A. N. 1993. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'93)*. ACM Press, 207–216.

Agrawal, R. and Srikant, R. 1994. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94)*. Morgan Kaufmann, 487–499. Introduction to the Quest data generator.

Besson, J., Robardet, C., Boulicaut, J.-F., and Rome, S. 2005. Constraint-based formal concept mining and its application to microarray data analysis. *Intell. Data Anal. 9*, 1, 59–82.

Boulicaut, J.-F. and Jeudy, B. 2005. Constraint-Based data mining. In *The Data Mining and Knowledge Discovery Handbook*, O. Maimon and L. Rokach, Eds. Springer, 399–416.

Brayton, R. K., Sangiovanni-Vincentelli, A. L., McMullen, C. T., and Hachtel, G. D. 1984. *Logic Minimization Algorithms for VLSI Synthesis*. Kluwer Academic, Norwell, MA.

Cerf, L., Besson, J., Robardet, C., and Boulicaut, J.-F. 2008. Data-Peeler: Constraint-Based closed pattern mining in *n*-ary relations. In *Proceedings of the 8th SIAM International Conference on Data Mining (SDM'08)*. SIAM.

McCluskey, J. 1956. Minimization of Boolean functions. *Bell Syst. Tech. J. 35*, 5, 1417–1444.

Gao, M., Jiang, J.-H., Jiang, Y., Li, Y., Sinha, S., and Brayton, R. 2000. MVSIS. In *Notes of the IEEE International Workshop on Logic Synthesis*. IEEE Computer Society.

Garriga, G. C., Khardon, R., and Raedt, L. D. 2007. On mining closed sets in multi-relational data. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*. AAAI Press, 804–809.

Gély, A. 2005. A generic algorithm for generating closed sets of a binary relation. In *Proceedings of the 3rd International Conference on Formal Concept Analysis (ICFCA'05)*. Lecture Notes in Computer Science, Vol, 3403, Springer, 223–234.

Goethals, B. and Zaki, M. J. 2004. Advances in frequent itemset mining implementations: Report on FIMI'03. *ACM SIGKDD Explor. Newslett. 6*, 1, 109–117.

Grahne, G. and Zhu, J. 2005. Fast algorithms for frequent itemset mining using FP-trees. *IEEE Trans. Knowl. Data Eng. 17*, 10, 1347–1362.

Han, J., Pei, J., and Yin, Y. 2000. Mining frequent patterns without candidate generation. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'00)*. ACM Press, 1–12.

Jaschke, R., Hotho, A., Schmitz, C., Ganter, B., and Stumme, G. 2006. Trias–An algorithm for mining iceberg tri-lattices. In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM'06)*. IEEE Computer Society, 907–911.

Ji, L., Tan, K.-L., and Tung, A. K. H. 2006. Mining frequent closed cubes in 3D data sets. In *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB'06)*. VLDB Endowment, 811–822.

Jiang, D., Pei, J., Ramanathan, M., Tang, C., and Zhang, A. 2004. Mining coherent gene clusters from gene-sample-time microarray data. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*. ACM Press, 430–439.

KARNAUGH, M. 1953. The map method for synthesis of combinational logic circuits. *Trans. Amer. Institute Electric. Eng. Part I 72*, 9, 593–599.

NG, R. T., LAKSHMANAN, L. V. S., HAN, J., AND PANG, A. 1998. Exploratory mining and pruning optimizations of constrained associations rules. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'98)*. ACM Press, 13–24.

PAN, F., CONG, G., TUNG, A. K., YANG, J., AND ZAKI, M. J. 2003. CARPENTER: Finding closed patterns in long biological datasets. In *Proceedings of the ACM SIGKDD'03*. ACM Press, 637–642.

PASQUIER, N., BASTIDE, Y., TAOUIL, R., AND LAKHAL, L. 1999. Efficient mining of association rules using closed itemset lattices. *Inf. Syst. 24*, 1 (Jan.), 25–46.

PEI, J., HAN, J., AND LAKSHMANAN, L. V. S. 2001. Mining frequent item sets with convertible constraints. In *Proceedings of the 17th International Conference on Data Engineering (ICDE'01)*. ACM Press, 433–442.

PEI, J., HAN, J., AND MAO, R. 2000. CLOSET: An efficient algorithm for mining frequent closed itemsets. In *Workshop on Research Issues in Data Mining and Knowledge Discovery (SIGMOD'00)*. ACM Press, 21–30.

PENSA, R. G. AND BOULICAUT, J.-F. 2005. Boolean property encoding for local set pattern discovery: An application to gene expression data analysis. In *Local Pattern Detection*. Vol. 3539. Springer, 115–134.

RUDELL, R. AND SANGIOVANNI-VINCENTELLI, A. 1985. Espresso-MV: Algorithms for multiple valued logic minimization. In *Proceedings of the IEEE Custom International Circuit Conference*. IEEE Computer Society, 230–234.

STUMME, G., TAOUIL, R., BASTIDE, Y., PASQUIER, N., AND LAKHAL, L. 2002. Computing iceberg concept lattices with titanic. *Data Knowl. Eng. 42*, 189–222.

SUN, J., TAO, D., AND FALOUTSOS, C. 2006. Beyond streams and graphs: Dynamic tensor analysis. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'06)*. ACM Press, 374–383.

UNO, T., KIYOMI, M., AND ARIMURA, H. 2005. LCM ver.3: Collaboration of array, bitmap and prefix tree for frequent itemset mining. In *Proceedings of the 1st ACM International Workshop on Open Source Data Mining (OSDM'05)*. ACM Press, 77–86.

WANG, J., HAN, J., AND PEI, J. 2003. CLOSET+: Searching for the best strategies for mining frequent closed itemsets. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'03)*. ACM Press, 236–245.

WILLE, R. 1982. Restructuring lattice theory: An approach based on hierarchies of concepts. In *Ordered Sets*, I. Rival, Ed. Reidel, 445–470.

ZAKI, M. J. AND HSIAO, C. J. 2002. CHARM: An efficient algorithm for closed itemset mining. In *Proceedings of the 2nd SIAM International Conference on Data Mining (SDM'02)*. SIAM.

ZHAO, L. AND ZAKI, M. J. 2005. TRICLUSTER: An effective algorithm for mining coherent clusters in 3D microarray data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'05)*. ACM Press, 694–705.

Received April 2008; revised October 2008; accepted November 2008