Intelligent Data Analysis 9(1):59-82, 2005. IOS Press.

Constraint-based concept mining and its application to microarray data analysis

Jérémy Besson[1,2], Céline Robardet[3], Jean-François Boulicaut[1], and Sophie Rome[2]

1: INSA Lyon, LIRIS CNRS FRE 2672

F-69621 Villeurbanne cedex, France

2: UMR INRA/INSERM 1235

F-69372 Lyon cedex 08, France

3: INSA Lyon, PRISMA

F-69621 Villeurbanne cedex, France

Contact email: Jeremy.Besson@liris.cnrs.fr

Abstract

We are designing new data mining techniques on boolean contexts to identify a priori interesting bi-sets, i.e., sets of objects (or transactions) and associated sets of attributes (or items). It improves the state of the art in many application domains where transactional/boolean data are to be mined (e.g., basket analysis, WWW usage mining, gene expression data analysis). The so-called (formal) concepts are important special cases of a priori interesting bi-sets that associate closed sets on both dimensions thanks to the Galois operators. Concept mining in boolean data is tractable provided that at least one of the dimensions (number of objects or attributes) is small enough and the data is not too dense. The task is extremely hard otherwise. Furthermore, it is important to enable user-defined constraints on the desired bi-sets and use them during the extraction to increase both the efficiency and the a priori interestingness of the extracted patterns. It leads us to the design of a new algorithm, called D-Miner, for mining concepts under constraints. We provide an experimental validation on benchmark data sets. Moreover, we introduce an original data mining technique for microarray data analysis. Not only boolean expression properties of genes are recorded but also we add biological information about transcription factors. In such a context, D-Miner can be used for concept mining under constraints and outperforms the other studied algorithms. We show also that data enrichment is useful for evaluating the biological relevancy of the extracted concepts.

Keywords. Pattern discovery, constraint-based data mining, closed sets, formal concepts, microarray data analysis.

1 Introduction

One of the most popular data mining techniques concerns transactional data analysis by means of set patterns. Indeed, following the seminal paper [1], hundreds of research papers have considered the efficient computation of a priori interesting association rules from the so-called frequent itemsets. Transactional data can be represented as boolean matrices (see Figure 1). Lines denote transactions and columns are boolean attributes that enable to record item occurrences. For instance, in Figure 1, transaction t_4 contains the items g_5, g_6, g_7, g_8, g_9, and g_{10}.

	Items									
	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}
t_1	1	1	1	1	0	1	1	0	0	0
t_2	1	1	1	1	0	0	0	0	1	1
t_3	1	1	1	1	0	0	0	0	1	1
t_4	0	0	0	0	1	1	1	1	1	1
t_5	1	0	1	0	1	1	1	1	0	0

Figure 1: Example of a boolean context $\mathbf{r}_1$

The frequent set mining problem concerns the computation of sets of attributes that are true together in enough transactions, i.e., given a frequency threshold. For instance, the set $\{g_1, g_2, g_3\}$ is considered frequent with a relative threshold of 30% since its frequency is 60%. It makes sense to associate to such a set, the set of transactions $\{t_1, t_2, t_3\}$, i.e., the set of transactions in which all the items from $\{g_1, g_2, g_3\}$ occur. It provides a so-called bi-set.

The typical case of basket analysis (huge - eventually millions - number of transactions, hundreds of attributes, but sparse and lowly-correlated data) can be handled by many algorithms, including APRIORI [2] and the various APRIORI-like algorithms that have been designed during the last decade. When the data are dense and highly-correlated, these algorithms fail but the so-called condensed representations of the frequent itemsets can be computed [7]. For instance, efficient algorithms can compute the frequent closed sets from which every frequent set and its frequency can be derived without accessing the data [18, 8, 19, 3, 26].

Interestingly, other important applications concern data sets with only a few transactions. This is the case for typical gene expression data analysis where items denote gene expression properties in biological situations. Here, the frequent sets denote sets of genes that are frequently co-regulated and thus can be suspected to participate to a common function within the cells. In common microarray data, the number of lines is a few tens and the number of columns is a few thousands. It is however possible to use the properties of Galois connection to compute the closed sets on the smaller dimension and derive the closed sets on the other dimension [20, 21]. Notice that once every closed set is known, all the frequent sets and their frequencies are known as well.

In this paper, we consider bi-set mining in difficult cases, i.e., when the data is dense and when none of the dimensions is quite small. Bi-sets are composed of a set of lines T and a set of columns G. T and G can be associated by various relationships, e.g., the fact that all the items of G are in relation with each transaction of T. Such a bi-set is called a 1-rectangle. It is interesting to constrain further 1-rectangle set components to be closed sets: it leads to maximal 1-rectangles that are also called formal concepts or concepts [25]. Minimal and maximal frequency but also syntactic constraints (see, e.g., [16]) on set components can be used as well.

The contribution of this paper is twofold.

First, we propose an original algorithm called D-Miner that computes concepts under constraints[1]. It works differently from other concept discovery algorithms (see, e.g., [13, 5, 17]) and frequent closed set computation algorithms [18, 8, 19, 3, 26]. Starting from the bi-set with all items and all transactions, D-Miner performs a depth-first search of concepts by recursively splitting into bi-sets that do not contain "0" values. D-Miner can be used in dense boolean data sets when the previous algorithms generally fail. Furthermore, thanks to an active use of user-defined monotonic constraints, it enlarges the applicability of concept discovery for matrices whose none of the dimensions is small.

A second major contribution concerns microarray data analysis. We introduce a new gene expression data mining technique based on constraint-based concept mining. Indeed, we consider a rather generic scenario where gene expression is available for two sets of biological situations, e.g., one data set about individuals that have a given pathology and another data set about individuals without this pathology. Boolean data is derived from raw microarray data and enriched with information about transcription factors. It leads to a richer boolean context from which a priori interesting concepts can be extracted. Working on an original data set, we point out one biologically relevant extracted concept that demonstrates the added-value of data enrichment with transcription factors.

Section 2 contains the needed definitions when considering constraint-based extraction of bi-sets from transactional/boolean data. Section 3 introduces the D-Miner algorithm. Proofs of D-Miner properties are given in technical Annex A. We also provide an experimental validation on artificial and benchmark data sets. For that purpose, D-Miner is compared with several efficient algorithms that compute frequent closed sets. Section 4 presents our current work on gene expression data analysis. It contains a formalization of a new interesting data mining technique for microarray data and several experimentations on real data. Finally, Section 5 concludes.

[1]A preliminary version of this paper has introduced D-Miner in [6].

2 Problem setting

Let $\mathcal{O}$ denotes a set of objects or transactions and $\mathcal{P}$ denotes a set of items or properties. In Figure 1, $\mathcal{O} = \{t_1, \ldots, t_5\}$ and $\mathcal{P} = \{g_1, g_2, \ldots, g_{10}\}$. The transactional data is represented by the matrix $\mathbf{r}$ of relation $R \subseteq \mathcal{O} \times \mathcal{P}$. We write $(t_i, g_j) \in \mathbf{r}$ to denote that item j belongs to transaction i or that property j holds for objects i. Such a data set can be represented by means of a boolean matrix (e.g., $\mathbf{r}_1$ in Figure 1).

For a typical basket analysis problem, $\mathcal{O}$ is the set of transactions and $\mathcal{P}$ corresponds to the products. The matrix then records which are the products that belong to the transactions, i.e., the products that have been purchased together. For instance, in Figure 1, we see that products identified by g_9 and g_{10} are bought together in transactions t_2, t_3, and t_4.

For a typical boolean gene expression data set (see, e.g., [4]), $\mathcal{O}$ is the set of biological situations and $\mathcal{P}$ corresponds to the genes. The relation then records which are the genes that are, e.g., over-expressed in the biological situations. Other expression properties can be considered (e.g., strong variation, under-expression). For instance, given Figure 1, we might say that genes g_5 and g_6 are over-expressed in situations t_4 and t_5.

The language of bi-sets is the collection of couples from $\mathcal{L} = \mathcal{L}_\mathcal{O} \times \mathcal{L}_\mathcal{P}$ where $\mathcal{L}_\mathcal{O} = 2^\mathcal{O}$ (sets of objects) and $\mathcal{L}_\mathcal{P} = 2^\mathcal{P}$ (sets of items).

Let us now consider evaluation functions for such patterns. This is not an exhaustive list of useful primitives. We mainly consider Galois operators (denoted as ϕ and ψ) that have been proved extremely useful.

Definition 1 *(Galois connection [25]) If $T \subseteq \mathcal{O}$ and $G \subseteq \mathcal{P}$, assume $\phi(T, \mathbf{r}) = \{g \in \mathcal{P} \mid \forall t \in T,\ (t, g) \in \mathbf{r}\}$ and $\psi(G, \mathbf{r}) = \{t \in \mathcal{O} \mid \forall g \in G,\ (t, g) \in \mathbf{r}\}$. ϕ provides the set of items that are common to a set of objects and ψ provides the set of objects that share a set of items. (ϕ, ψ) is the so-called Galois connection between $\mathcal{O}$ and $\mathcal{P}$. We use the classical notations $h = \phi \circ \psi$ and $h' = \psi \circ \phi$ to denote the Galois closure operators.*

Definition 2 *(Frequency) The frequency of a set of items $G \subseteq \mathcal{P}$ in $\mathbf{r}$ is $|\psi(G, \mathbf{r})|$. The frequency of a set of objects $T \subseteq \mathcal{O}$ in $\mathbf{r}$ is $|\phi(T, \mathbf{r})|$.*

Example 1 *Given Figure 1, let us consider the bi-set (T, G) where $G = \{g_1, g_2\}$ and $T = \{t_1, t_2, t_3\}$. The frequency of G is $|\psi(G, \mathbf{r}_1)| = |\{t_1, t_2, t_3\}| = 3$, the frequency of T is $|\phi(T, \mathbf{r}_1)| = |\{g_1, g_2, g_3, g_4\}| = 4$. Notice that, for this bi-set, $T = \psi(G, \mathbf{r}_1)$ but $G \neq \phi(T, \mathbf{r}_1)$.*

Definition 3 *(Constraints on frequencies) Given a set of items $G \subseteq \mathcal{P}$ and a frequency threshold γ, the minimal frequency constraint is denoted $\mathcal{C}_{\mathrm{minfreq}}(\mathbf{r}, \gamma, G) \equiv |\psi(G, \mathbf{r})| \geq \gamma$. The maximal frequency w.r.t. a threshold γ' is $\mathcal{C}_{\mathrm{maxfreq}}(\mathbf{r}, \gamma', G) \equiv |\psi(G, \mathbf{r})| \leq \gamma'$. These constraints can be defined easily on sets of objects as well: $\mathcal{C}_{\mathrm{minfreq}}(\mathbf{r}, \gamma, T) \equiv |\phi(T, \mathbf{r})| \geq \gamma$ and $\mathcal{C}_{\mathrm{maxfreq}}(\mathbf{r}, \gamma', T) \equiv |\phi(T, \mathbf{r})| \leq \gamma'$.*

Definition 4 *(Closed set and $\mathcal{C}_{Close}$ constraint) A set of items $G \subseteq \mathcal{P}$ is closed when it satisfies constraint $\mathcal{C}_{Close}$ in $\mathbf{r}$ and $\mathcal{C}_{Close}(G, \mathbf{r}) \equiv h(G, \mathbf{r}) = G$. Dually, for sets of objects $T \subseteq \mathcal{O}$, we have $\mathcal{C}_{Close}(T, \mathbf{r}) \equiv h'(T, \mathbf{r}) = T$.*

Example 2 *Let us consider again the bi-set ($\{t_1, t_2, t_3\}, \{g_1, g_2\}$) and the data from Figure 1. $\{g_1, g_2\}$ satisfies $\mathcal{C}_{\text{minfreq}}(\mathbf{r}_1, 3, G)$. The set $\{g_1, g_2\}$ is not closed: $h(\{g_1, g_2\}, \mathbf{r}_1) = \phi(\psi(\{g_1, g_2\}, \mathbf{r}_1), \mathbf{r}_1) = \{g_1, g_2, g_3, g_4\}$. An example of a closed set on items is $\{g_1, g_2, g_3, g_4\}$: $h(\{g_1, g_2, g_3, g_4\}, \mathbf{r}_1) = \{g_1, g_2, g_3, g_4\}$. Also, the set of objects $\{t_1, t_2, t_3\}$ is closed: $h'(\{t_1, t_2, t_3\}, \mathbf{r}_1) = \{t_1, t_2, t_3\}$.*

The closure of a set of items G, $h(G, \mathbf{r})$, is the maximal (w.r.t. set inclusion) superset of G which has the same frequency than G in $\mathbf{r}$. A closed set of items is thus a maximal set of items which belong to a given set of transactions. For instance, the closed set $\{g_1, g_3\}$ in $\mathbf{r}_1$ (see Figure 1) is the largest set of items that are shared by transactions t_1, t_2, t_3 and t_5.

Definition 5 *(1-rectangles and 0-rectangles) A bi-set (T, G) is a 1-rectangle iff $\forall g \in G$ and $\forall t \in T$, $(t, g) \in \mathbf{r}$. A bi-set (T, G) is a 0-rectangle in $\mathbf{r}$ iff $\forall t \in T$ and $\forall g \in G$ then $(t, g) \notin \mathbf{r}$.*

Definition 6 *(Concept) (T, G) is called a formal concept or concept in $\mathbf{r}$ when $T = \psi(G, \mathbf{r})$ and $G = \phi(T, \mathbf{r})$ [25]. A concept is thus a maximal 1-rectangle.*

Given the Galois connection, concepts are built on closed sets and each closed set of items (resp. objects) is linked to a closed set of objects using ψ (resp. items using ϕ) [25]. In other terms, for a concept (T, G) in $\mathbf{r}$, constraint $\mathcal{C}_{Close}(T, \mathbf{r}) \wedge \mathcal{C}_{Close}(G, \mathbf{r})$ is satisfied. Notice also that the collection of concepts is obviously included in the collection of 1-rectangles.

Example 3 *Given Figure 1, $(\{t_1, t_2, t_3\}, \{g_1, g_2\})$ is a 1-rectangle but it is not a concept. Twelve bi-sets are concepts in $\mathbf{r}_1$. $(\{t_2, t_3\}, \{g_1, g_2, g_3, g_4, g_9, g_{10}\})$ and $(\{t_1, t_2, t_3, t_5\}, \{g_1, g_3\})$ are examples of concepts in $\mathbf{r}_1$.*

Other interesting primitive constraints can be defined, e.g., syntactic constraints on each set component of a bi-set. A typical example is to enforce that a given item belongs or does not belong to the item set component.

Many data mining processes on transactional data can be formalized as the computation of bi-sets whose set components satisfy combinations of primitive constraints.

Mining the frequent sets of items is specified as the computation of $\{G \in \mathcal{L}_{\mathcal{P}} \mid \mathcal{C}_{\text{minfreq}}(\mathbf{r}, \gamma, G) \text{ satisfied}\}$. We can then provide the bi-sets of the form $\{(T, G) \in \mathcal{L}_{\mathcal{O}} \times \mathcal{L}_{\mathcal{P}} \mid \mathcal{C}_{\text{minfreq}}(\mathbf{r}, \gamma, G) \wedge T = \psi(G, \mathbf{r})\}$. These bi-sets are frequent sets of attributes associated to the objects in which they occur. Another typical post-processing of frequent sets is to compute frequent association rules from the frequent sets and keep the ones with enough confidence [2].

An important data mining task concerns frequent closed set mining, i.e., the computation of $\Theta_1 = \{G \in \mathcal{L_P} \mid \mathcal{C}_{\text{minfreq}}(\mathbf{r}, \gamma, G) \wedge \mathcal{C}_{Close}(G, \mathbf{r}) \text{ satisfied}\}$. This task has been studied a lot as the computation of a condensed representation for the frequent sets [18, 7]. We illustrate in Section 3.3 the computation of such collections using Charm, Ac-Miner, and Closet. Frequent (closed) sets of objects can be desired as well.

The collection $\Theta_2 = \{(T, G) \in \mathcal{L_O} \times \mathcal{L_P} \mid \mathcal{C}_{Close}(G, \mathbf{r}) \wedge T = \psi(G, \mathbf{r})\}$ is the collection of concepts. We have also $\Theta_2 = \{(T, G) \in \mathcal{L_O} \times \mathcal{L_P} \mid \mathcal{C}_{Close}(T, \mathbf{r}) \wedge G = \phi(T, \mathbf{r})\}$. Let us point out that this provides a strategy for computing concepts by first choosing to compute the closed sets on the smaller dimension.

When we use our algorithm D-Miner in Section 3.3, we compute $\Theta_3 = \{(T, G) \in \mathcal{L_O} \times \mathcal{L_P} \mid \mathcal{C}_{\text{minfreq}}(\mathbf{r}, \gamma, G) \wedge \mathcal{C}_{Close}(G, \mathbf{r}) \wedge T = \psi(G, \mathbf{r})\}$, i.e., the sets from Θ_1 to which we associate sets of transactions that support them (using ψ).

Definition 7 *(Transposition) If $\mathbf{r} \subseteq \mathcal{O} \times \mathcal{P}$, its transposition is ${}^t\mathbf{r} \subseteq \mathcal{P} \times \mathcal{O}$ where $(g, t) \in {}^t\mathbf{r} \equiv (t, g) \in \mathbf{r}$.*

Property 1 *If $G \subseteq \mathcal{P}$ is a set of items and $T \subseteq \mathcal{O}$ is a set of objects, we have $\psi(T, {}^t\mathbf{r}) = \phi(T, \mathbf{r})$ and $\phi(G, {}^t\mathbf{r}) = \psi(G, \mathbf{r})$. As a result, we have also $h(T, {}^t\mathbf{r}) = h'(T, \mathbf{r})$ and $h'(G, {}^t\mathbf{r}) = h(G, \mathbf{r})$. The proof of these properties is straightforward.*

These observations explain why it is possible to compute the whole collection of concepts as $\Theta = \{(T, G) \in \mathcal{L_O} \times \mathcal{L_P} \mid h(G, \mathbf{r}) = G \wedge T = \psi(G, \mathbf{r})\}$ when $|\mathcal{P}| < |\mathcal{O}|$ or as $\Theta = \{(T, G) \in \mathcal{L_O} \times \mathcal{L_P} \mid h'(T, \mathbf{r}) = T \wedge G = \phi(T, \mathbf{r})\}$ otherwise. In this second case, the same algorithm can be used after a simple transposition since $h'(T, \mathbf{r}) = h(T, {}^t\mathbf{r})$ and $\phi(T, \mathbf{r}) = \psi(T, {}^t\mathbf{r})$.

We are currently applying concept discovery techniques on typical microarray data sets (see Section 5).

In such applications, we can get dense boolean contexts that are hard to process if further user-defined constraints are not only specified but also pushed deeply into the extraction algorithms. Using user-defined constraints enables to produce putative interesting concepts before any post-processing phase. Indeed, concept discovery techniques can provide huge collection of patterns (e.g., we report in Section 5 an output of more than 5 millions of concepts) and supporting post-processing on such collections is hard or even impossible. It motivates the a priori use of constraints on both $\mathcal{L_O}$ and $\mathcal{L_P}$. Typical examples concern constraints on the size of T and G. It leads to the extraction of frequent concepts. Furthermore, the key issue for pushing user-defined constraints deeply into the mining algorithms is a clever use of specialization relations on the pattern language. For instance, it is well-known that pushing monotonic or anti-monotonic (w.r.t. a specialization relation) constraints can drastically reduce the amount of needed resources during an extraction (see, e.g., [16, 14]). Let us now introduce our specialization relation on bi-sets.

Definition 8 *(Specialization relation and monotonicity) Our specialization relation on bi-sets from $\mathcal{L} = \mathcal{L}_{\mathcal{O}} \times \mathcal{L}_{\mathcal{P}}$ is defined by $(T_1, G_1) \leq (T_2, G_2)$ iff $T_1 \subseteq T_2$ and $G_1 \subseteq G_2$. A constraint $\mathcal{C}$ is said anti-monotonic w.r.t. $\leq$ iff $\forall \alpha, \beta \in \mathcal{L}$ such that $\alpha \leq \beta$, $\mathcal{C}(\beta) \Rightarrow \mathcal{C}(\alpha)$. $\mathcal{C}$ is said monotonic w.r.t. $\leq$ iff $\forall \alpha$, $\beta \in \mathcal{L}$ such that $\alpha \leq \beta$, $\mathcal{C}(\alpha) \Rightarrow \mathcal{C}(\beta)$.*

Definition 9 *(Frequent concept) A concept (T, G) is called frequent when constraint $\mathcal{C}_t(\mathbf{r}, \sigma_1, T)$ (resp. $\mathcal{C}_g(\mathbf{r}, \sigma_2, G)$) if $|T| \geq \sigma_1$ (resp. $|G| \geq \sigma_2$). These constraints are both monotonic w.r.t. $\leq$ on $\mathcal{L}_{\mathcal{O}} \times \mathcal{L}_{\mathcal{P}}$.*

Notice that the constraint $\mathcal{C}_t(\mathbf{r}, \sigma, T) \equiv |T| \geq \sigma \equiv \mathcal{C}_{\text{minfreq}}(\mathbf{r}, \sigma, \phi(T, \mathbf{r}))$. When considering frequent concept mining, we use monotonic constraints on the size of sets but any other monotonic constraint could be used as well.

We choose to enforce the symmetry of our extractor on $\mathcal{L}_{\mathcal{O}}$ and $\mathcal{L}_{\mathcal{P}}$ to be able to transpose the matrix without loosing the possibility of using the constraints. Indeed, as explain earlier, where there are few objects and many items, we first perform a simple transposition to reduce the extraction complexity.

3 D-Miner

D-Miner is a new algorithm for extracting concepts (T, G) under constraints. It builds simultaneously the sets T and G and it uses monotonic constraints w.r.t. our specialization relation, simultaneously on $\mathcal{L}_{\mathcal{O}}$ and $\mathcal{L}_{\mathcal{P}}$, to reduce the search space. Its originality is to construct simultaneously closed set on $\mathcal{L}_{\mathcal{O}}$ and $\mathcal{L}_{\mathcal{P}}$ such that it enables to push monotonic constraints on the two set components.

3.1 Principle

A concept (T, G) is such that all its items and objects are in relation by R. Thus, the absence of relation between an item g and an object t generates two concepts, one with g and without t, and another one with t and without g. D-Miner is based on this observation.

Let us denote by H a set of 0-rectangles (Definition 5) such that it is a cover of the false values (0) of the boolean matrix $\mathbf{r}$, i.e., $\forall g \in \mathcal{P}$ and $\forall t \in \mathcal{O}$ such that $(t, g) \notin \mathbf{r}$, it exists at least one element (X, Y) of H such that $t \in X$ and $g \in Y$. The elements of H are called *cutters*. The principle of D-Miner consists in starting with the couple $(\mathcal{O}$,$\mathcal{P})$ and then splitting it recursively using the elements of H until H is empty and thus each couple is a 1-rectangle. An element (a, b) of H is used to cut a couple (X, Y) if $a \cap X \neq \emptyset$ and $b \cap Y \neq \emptyset$.

By convention, one defines the left son of (X, Y) by $(X \setminus a, Y)$ and the right son by $(X, Y \setminus b)$.

This principle enables to obtain all the concepts, i.e., the maximal 1-rectangles (see Example 4) but also some non maximal ones (see Example 5). Therefore, we will explain how to prune them to obtain all the concepts and only the concepts.

The construction of H is a key phase since its elements are used to recursively split couples of sets: H must be as small as possible to reduce the depth of recursion and thus execution time. On another hand, one should not waste too much time to compute H. H contains as many elements as lines in the matrix. Each element is composed of the attributes valued by '0' in this line.

Time complexity for computing H is in $O(|\mathcal{O}| \times |\mathcal{P}|)$. It is thus easy to compute it and computing time is negligible w.r.t. the one of the cutting procedure. Furthermore, using this definition makes easier the pruning of 1-rectangles that are not concepts.

We now provide examples of D-Miner executions considering only the constraints that enforce a bi-set to be a concept. The use of monotonic constraints on $\mathcal{L}_\mathcal{O}$ and $\mathcal{L}_\mathcal{P}$ is presented later.

Example 4 *Assume $\mathcal{O} = \{t_1, t_2, t_3\}$ and $\mathcal{P} = \{g_1, g_2, g_3\}$. Relation $\mathbf{r}_2$ is defined in Table 1 (left). Here, H is $\{(t_1, g_1), (t_3, g_2)\}$.* D-Miner *algorithm starts with $(\mathcal{O}, \mathcal{P})$. Figure 2 illustrates this process. We get four 1-rectangles that are the four concepts for this boolean context. Notice that for the sake of simplicity, we do not use here the standard notation for set components within bi-sets: a bi-set like $(t_1t_2t_3, g_1g_2g_3)$ denotes $(\{t_1, t_2, t_3\}, \{g_1, g_2, g_3\})$.*

	g_1	g_2	g_3
t_1	0	1	1
t_2	1	1	1
t_3	1	0	1

	g_1	g_2	g_3
t_1	0	0	1
t_2	1	0	1
t_3	0	0	1

Table 1: Contexts $\mathbf{r}_2$ for Example 4 (left) and $\mathbf{r}_3$ for Examples 5 and 6 (right)

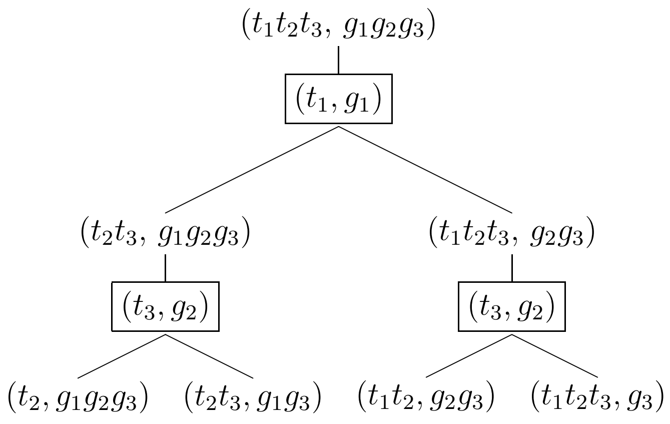

Figure 2: Concept construction on $\mathbf{r}_2$ (Example 4)

Example 5 *Assume now the relation* $\mathbf{r}_3$ *as given in Table 1 (right). Computing* H *provides* $\{(t_1, g_1g_2),\ (t_2, g_2),\ (t_3, g_1g_2)\}$. *Figure 3 illustrates* D-MINER *execution. Some bi-sets are underlined and this will be explained in Example 6.*

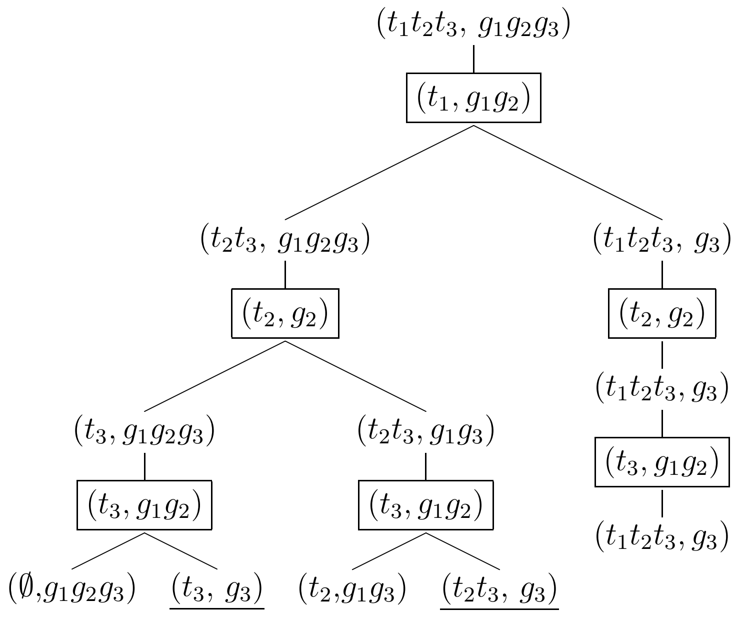

Figure 3: Concept construction on $\mathbf{r}_3$ (Example 5)

From Figure 3, we can see that (t_2, g_2) and (t_3, g_1g_2) from H are not used to cut $(t_1t_2t_3, g_3)$ because $\{g_2\} \cap \{g_3\} = \emptyset$ and $\{g_1g_2\} \cap \{g_3\} = \emptyset$. The computed collection of bi-sets is:

$$\{(t_1t_2t_3, g_3),\ (t_2, g_1g_3),\ (\emptyset, g_1g_2g_3),\ \underline{(t_3, g_3),\ (t_2t_3, g_3)}\}$$

We see that $(t_3, g_3) \leq (t_1t_2t_3, g_3)$ and $(t_2t_3, g_3) \leq (t_1t_2t_3, g_3)$ and thus these 1-rectangles are not concepts.

To explain why the method computes the concepts and some subsets of them, we introduce a new notation. Let $\mathbf{r}[T, G]$ denote the reduction of $\mathbf{r}$ on objects from T and on items from G. When a bi-set (X, Y) is split by a cutter $(a, b) \in H$, then $(X \setminus a, Y)$ (the left son) and $(X, Y \setminus b)$ (the right son) are generated. If a concept (C_X, C_Y) exists in $\mathbf{r}[X \setminus a, Y]$ such that $C_Y \cap b = \emptyset$ then $(C_X \cup a, C_Y)$ is a concept in $\mathbf{r}[X, Y]$. Indeed, given our technique for computing H, $(a, Y \setminus b)$ is a 1-rectangle (see Figure 4).

Property 2 *Let* (X, Y) *be a leaf of the tree and* $H_L(X, Y)$ *be the set of cutters associated of the left branches of the path from the root to* (X, Y). *Then* (X, Y) *is a concept iff* Y *contains at least one item of each itemset of* $H_L(X, T)$.

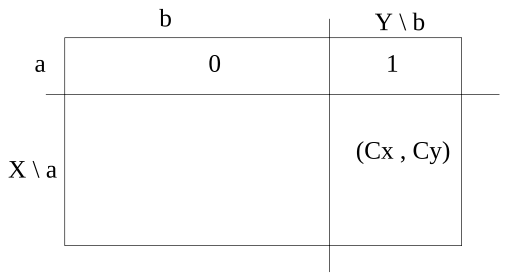

Figure 4: Building a non maximal 1-rectangle

Proofs of correctness and completeness of D-Miner are given in Annex A.

It means that when trying to build a right son (X, Y) (i.e., to remove some elements from Y), we must check that $\forall (a, b) \in H_L(X, Y), b \cap Y \neq \emptyset$. This constraint is called the left cutting constraint.

Example 6 *We take the context used for Example 5 (see Table 1 on the right). 1-rectangles (t_3, g_3) and (t_2t_3, g_3) are pruned using Property 2. (t_3, g_3) comes from the left cutting of ($t_1t_2t_3$,$g_1g_2g_3$) and then the left cutting of ($t_2t_3, g_1g_2g_3$). The items of (t_3, g_3) must contain at least one item of $\{g_1, g_2\}$ and one item of $\{g_2\}$, i.e., the precedent left cutter set of items. It is not the case and thus (t_3, g_3) is pruned. (t_2t_3, g_3) comes from just one left cutter: (t_1,g_1g_2). It contains neither g_1 nor g_2. Consequently, nodes that are underlined in Figure 3 are pruned.*

3.2 Algorithm

Before cutting a bi-set (X, Y) by a cutter (a, b) in two bi-sets $(X \backslash a, Y)$ and $(X, Y \backslash b)$, two types of constraints must be checked, first the monotonic constraints and then the left cutting constraint. Closeness property (or maximality) is implied by the cutting procedure.

D-Miner is a depth-first method which generates bi-sets ordered by relation $\leq$. Monotonic constraints w.r.t. either $\mathcal{O}$ or $\mathcal{P}$ are used to prune the search space: if (X, Y) does not satisfy a monotonic constraint $\mathcal{C}$ then none of its sons satisfies $\mathcal{C}$ and it is unnecessary to cut (X, Y). Constraint $\mathcal{C}$ is a conjunction of a monotonic constraint $\mathcal{C}_t$ on $\mathcal{O}$ and a monotonic constraint $\mathcal{C}_g$ on $\mathcal{P}$. Algorithms 1 and 2 contain the pseudo-code of D-Miner. First, the set H of cutters is computed. Then the recursive function *cutting()* is called.

Function *cutting* cuts out a bi-set (X, Y) with the first cutter $H[i]$ that satisfies the following constraints. First, (X, Y) must have an non empty intersection with $H[i]$. If it is not the case, *cutting* is called with the next cutter. Before cutting (X, Y) into $(X \backslash a, Y)$, we have to check the monotonic constraint on $X \backslash a$ (i.e., $\mathcal{C}_t(X \backslash a)$) to try to prune the search space. (a, b) is inserted into H_L, the set of cutters in the left cutting. Then *cutting* is called on $(X \backslash a, Y)$ and (a, b) is removed from H_L. For the

Algorithm 1: D-Miner

```
Input : Relation $\mathbf{r}$ with $n$ objects and $m$ items, $\mathcal{O}$ the set of objects, $\mathcal{P}$
the set of items, $\mathcal{C}_t$ and $\mathcal{C}_g$ are monotonic constraints on $\mathcal{O}$ and $\mathcal{P}$.
Output : $\mathcal{Q}$ the set of concepts that satisfy $\mathcal{C}_t$ and $\mathcal{C}_g$
$H_L \leftarrow$ empty()
$H$ is computed from $\mathbf{r}$
$\mathcal{Q} \leftarrow$ cutting(($\mathcal{O}$, $\mathcal{P}$), $H$, 0, $H_L$)
```

Algorithm 2: cutting

```
Input: $(X, Y)$ a couple of $2^{\mathcal{O}} \times 2^{\mathcal{P}}$, $H$ the list of cutters, $i$ the depth
of the iteration, $H_L$ a set of precedent cutters in left cuttings,
$\mathcal{C}_t$ a monotonic constraint on $\mathcal{O}$, $\mathcal{C}_g$ a monotonic constraint on $\mathcal{P}$.
Output: $\mathcal{Q}$ the set of concepts that satisfy $\mathcal{C}_t$ and $\mathcal{C}_g$
$(a, b) \leftarrow$ H[i]
If $(i \leq |H| - 1)$ // i-th cutter is selected
  If $((a \cap X = \emptyset)$ or $(b \cap Y = \emptyset))$
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup cutting((X, Y), H, i + 1, H_L)$
  Else
    If $\mathcal{C}_t(\mathbf{r}, \sigma_1, X \setminus a)$ is satisfied
      $H_L \leftarrow H_L \cup (a, b)$
      $\mathcal{Q} \leftarrow \mathcal{Q} \cup$ $cutting((X \backslash a, Y), H, i + 1, H_L)$
      $H_L \leftarrow H_L \setminus (a, b)$
    If $\mathcal{C}_g(\mathbf{r}, \sigma_2, Y \backslash b)$ is satisfied $\wedge$ $\forall (a', b') \in H_L$, $b' \cap Y \backslash b \neq \emptyset$
      $\mathcal{Q} \leftarrow \mathcal{Q} \cup cutting((X, Y \backslash b), H, i + 1, H_L)$
Else
  $\mathcal{Q} \leftarrow (X, Y)$
Return $\mathcal{Q}$
```

second cutting of (X, Y), two constraint checking are needed. First the monotonic constraints on $Y \backslash b$ (i.e., $\mathcal{C}_g(Y \setminus b)$) is checked and then we prune redundant concepts using left cutters.

It is possible to optimize this algorithm. First, the order of the elements of H is important. The aim is to cut as soon as possible the branches which generate non-maximal 1-rectangles. Then, H is sorted by decreasing order of size of the itemset component. Moreover, to reduce the size of H, the cutters which have the same set of items are gathered: $\forall (a_1, b_1)$, $(a_2, b_2) \in H$, if $b_1 = b_2$ then $H = H \setminus \{(a_1, b_1), (a_2, b_2)\} \cup (a_1 \cup a_2, b_1)$. Finally, if $|\mathcal{P}| > |\mathcal{O}|$, we transpose the matrix to obtain a set H of minimum size.

3.3 Experimental validation

We compare the execution time of D-Miner with those of Closet [19], Ac-Miner [8, 9] and Charm [26]. Closet, Ac-Miner and Charm are efficient algorithms that compute frequent closed sets. These algorithms are based on different techniques. For instance, Closet and Charm perform a depth-first search for frequent closed sets. Ac-Miner computes level-wise the frequent free sets and then output their closures, i.e., the frequent closed sets. An algorithm like Pascal [3] does the same even though the free sets are here called key patterns. On another hand, D-Miner can be used to compute frequent concepts and thus frequent closed sets. It motivates this experimental validation where we extract frequent closed sets for various frequency thresholds on artificial data sets and two benchmark data sets from the UCI repository. As we explained earlier, it is easy to provide concepts from closed sets on $\mathcal{O}$ or $\mathcal{P}$ and, in these experiments, we just compare the execution time of closed set mining algorithms like Closet or Charm without the small extra-time for concept generation.

We use Zaki's implementation of Charm [26] and Bykowski's implementations of Ac-Miner [8, 9] and Closet [19]. For a fair comparison, in every experiment, we transpose the matrices to have the smaller number of columns for Closet, Ac-Miner and Charm and to have the smaller number of lines for D-Miner. Indeed, the computation complexity depend on the number of columns for Closet, Ac-Miner and Charm and on the number of lines for D-Miner. The minimal frequency threshold is given as a relative frequency w.r.t. the total number of objects/items. All extractions have been performed on a Pentium III (450 MHz, 128 Mb).

In Annex B, we present the execution time of the four algorithms on several artificial data sets generated with the IBM[2] generator. We generated 60 dense data sets by varying the number of columns (300, 700 and 900 columns), the number of lines (100 and 300 lines) and the density of "1" values (15 % and 35 %). For each combination of parameters, we generated 5 data sets. Table 2 of Annex B provides the mean and the standard deviation of execution time (in seconds) used by each algorithm on data sets with 15% density. We observe that d-miner succeeds to extract closed sets in contexts where other algorithms do not succeed. Table 3 of Annex B provides the mean and the standard deviation of execution time (in seconds) used by each algorithms on data sets with 35% density. For this type of data, the superiority of d-miner is even more obvious.

In Figure 5, we show the needed execution time for frequent closed pattern extraction on the benchmark "Mushroom". Its derived boolean context contains 8 124 lines and 120 columns.

[2]http://miles.cnuce.cnr.it/ palmeri/datam/DCI/datasets.php

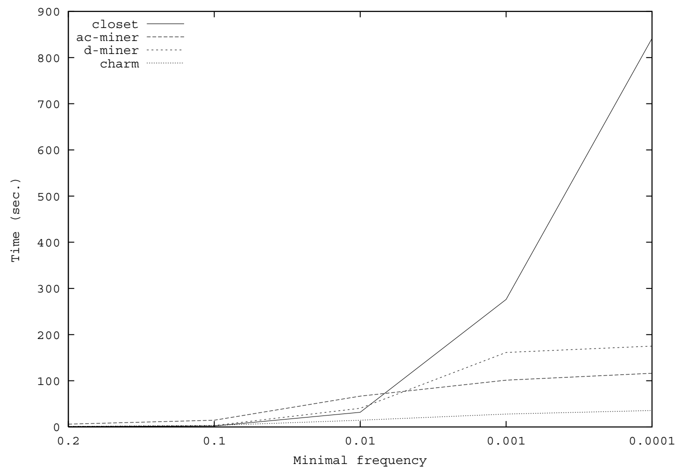

Figure 5: Mushroom

On “Mushroom”, the four algorithms succeed in finding all the concepts (minimal frequency 0.0001) within a few minutes. The lowest frequency threshold corresponds to at least 1 object. The execution time of Closet increases very fast compared to the three others.

Next, we considered the benchmark “Connect4”. The derived boolean context contains 67 557 lines and 149 columns. The execution time is shown on Figure 6. Only Charm and D-Miner can extract closed sets with minimal frequency equal to 0.1 and the associated frequent concepts. D-Miner is almost twice faster than Charm on this data set. Notice, however, that the extraction of every concept, including the infrequent ones, remains intractable on “Connect4”.

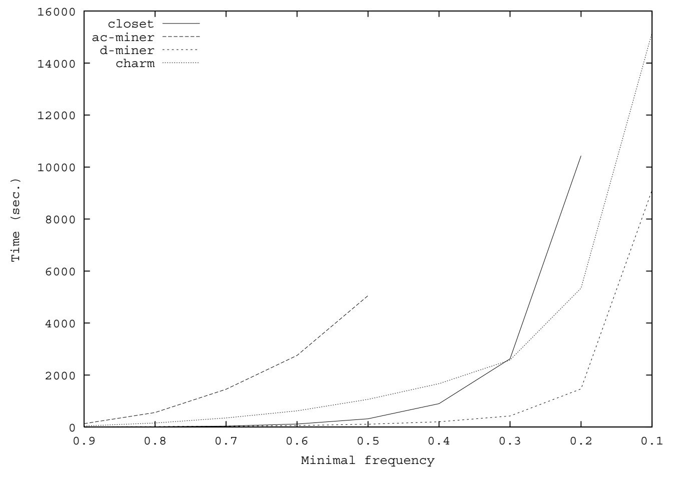

Figure 6: Connect4

4 An application to microarray data analysis

4.1 An original data set and mining task

One of the well accepted biological hypothesis is to consider that most of the specific properties of a cell result from the interaction between its genes and its cellular environment. At a molecular level, functional adaptation involves quantitative and qualitative changes in gene expressions. Analysis of all the RNA transcripts present in a cell or in a tissue (named the transcriptome) offers unprecedented opportunities to map the transitions between an healthy state and a pathological one. It has motivated a huge effort for technologies that could monitor simultaneously the expression level of a large number of genes [10, 24]. The cDNA microarray technology [10] is such a powerful tool for the global analysis of gene expression and thus for identifying genes related to disease states and targets for clinical intervention. The data generated by these experiments can be seen as expression matrices in which the expression level (real value) of genes are recorded in various situations. Life scientists are facing an unprecedent data-analysis challenge since the large data sets generated from microarray analysis (involving thousands of genes) are not easily analyzed by classical one-by-one genetics. They need new techniques that can support the discovery of biological knowledge from these data sets.
Indeed, the logical step which follows the identification of groups of co-regulated genes is the discovery of the common mechanism which coordinated the regulation of these genes.

Typical techniques support the search for genes and situations with similar expression profiles. This can be achieved with hierarchical clustering (e.g., [12]), biclustering (e.g., [22, 15]), CPA and statistical analysis (e.g., [11]). Frequent set mining has been studied as a complementary approach for the identification of a priori interesting set of genes and situations (see, e.g., [4]). In that case, boolean matrices are built that record expression properties of the genes (e.g., over-expression, strong variation).

Our concrete application concerns the analysis of DNA microarray data. Each DNA microarray contains the RNA expression level of about 20 000 genes before and after a perfusion of insulin in their skeletal muscle [23]. For each gene and each subject we take the logarithm of the ratio between the two measures. In our experimentation 10 DNA microarrays have been made, 5 for healthy persons and 5 for type 2 diabetic patients.

Statistical preprocessing has lead to the selection of 8 171 genes. Among these genes, many of them (more than 10%) are differently expressed between the healthy and the diabetic persons. A subset of such data is given in Figure 7 to illustrate the discretization procedure and the enrichment of the data. Assume that here, we have 5 genes (g_1, g_2, g_3, g_4, g_5) and two sets of biological situations ($\mathcal{H}$ composed of the two healthy subjects and $\mathcal{D}$ composed of three diabetic patients). The first two situations correspond to gene expression level for healthy people, $\mathcal{H} = \{h_1, h_2\}$, and the three following ones correspond to gene expression level for diabetic patients, $\mathcal{D} = \{d_1, d_2, d_3\}$. A positive value (resp. negative value) means that the gene is up-regulated (resp. down-regulated) in the situation.

		Genes				
		g_1	g_2	g_3	g_4	g_5
$\mathcal{H}$	h_1	0.36	0.42	0.56	0.124	0.35
	h_2	-0.24	0.01	0.28	0.02	-0.32
$\mathcal{D}$	d_1	0.25	0.35	0.55	0.012	-0.21
	d_2	0.27	0.89	-1.02	0.71	0.52
	d_3	0.53	0.24	0.64	-0.6	-0.01

Figure 7: Microarray raw data

We are interested in genes with strong variation under the insulin clamp. Figure 8 provides a boolean context for the expression data from Figure 7. A value "1" for a biological situation and a gene means that the gene is up (greater than $|t|$) or down (lower than $-|t|$) regulated in this situation. On this toy example, we choose a threshold $t = 0.3$. The discretization of the gene expression data into boolean expression properties is however a complex task (see, [4] for a discussion). It is out of the scope of this paper to discuss further this issue.

		Genes				
		g_1	g_2	g_3	g_4	g_5
$\mathcal{H}$	h_1	1	1	1	0	1
	h_2	0	0	0	0	1
$\mathcal{D}$	d_1	0	1	1	0	0
	d_2	0	1	1	1	1
	d_3	1	0	1	1	0

Figure 8: Example of boolean microarray data $\mathbf{r}_4$

Let us now propose an original enrichment for this kind of boolean context. Additional information on, e.g., molecular functions, biological processes and the presence of transcription factors, are useful to interpret the extracted bi-sets or concepts. Transcription factors (TF) are proteins which specifically recognize and bind DNA regions of the genes. Following the binding, the expression of the gene is increased or decreased. A gene is usually regulated by many TF, and a TF can regulate different genes at the same time. In our current application, the challenge for the biologist is the identification of a combination of TF associated to a group of genes which are regulated in the group of healthy subjects but not in the diabetic group. To the best of our knowledge, no data mining technique supports such an analysis. We have been looking for the transcription factors that can bind to the regulatory region of each gene of the microarray data. For that purpose, we have used EZ-Retrieve, a free software for batch retrieval of coordinated-specified human DNA sequences [27]. In Figure 9, we continue with our toy example by considering two transcription factors tf_1 and tf_2. A value "1" for a gene and a transcription factor means that this transcription factor regulates the gene. We have a value "0" otherwise.

		Genes				
		g_1	g_2	g_3	g_4	g_5
$\mathcal{F}$	tf_1	1	1	0	1	0
	tf_2	1	0	0	1	1

Figure 9: Transcription factors for $\mathbf{r}_4$

It is now possible to merge the boolean data about biological situations and transcription factors into a unique boolean context (see Figure 10). Such a context encodes information about different gene properties that are biologically relevant: its expression level for healthy people, its expression level for diabetic patient and the regulation by transcription factors. The set $\mathcal{O}$ is thus partitioned into the set $\mathcal{H}$, the set $\mathcal{D}$ and a set of transcription factors $\mathcal{F}$.

		Genes				
		g_1	g_2	g_3	g_4	g_5
$\mathcal{H}$	h_1	1	1	1	0	1
	h_2	0	0	0	0	1
$\mathcal{D}$	d_1	0	1	1	0	0
	d_2	0	1	1	1	1
	d_3	1	0	1	1	0
$\mathcal{F}$	tf_1	1	1	0	1	0
	tf_2	1	0	0	1	1

Figure 10: An enriched microarray boolean context

For technical reasons, 600 genes among the 8 171 ones for which expression value is available have been processed. Among these 600 genes, 496 genes have unknown functions and thus the list of the transcription factors that might regulate them is not available. We finally got a list of 94 transcription factors that can regulate 304 genes. In other terms, the boolean context we have has 104 objects (94 transcription factors and 10 biological situations, 5 for healthy individuals and 5 for diabetic patients) and 304 genes. Concept discovery from such a boolean context is already hard. Furthermore, after discretization, we got a dense boolean context with 17% of the cells containing the true value (i.e., "1"). In such a situation, available algorithms for mining closed sets can turn to be intractable. Furthermore, in a short term perspective, we will get much more transcription factors and much more genes (up to several hundreds or thousands).

The challenge that has been proposed by the biologists is as follows. The potentially interesting bi-sets are concepts that involve a minimum number of genes (more than γ) to ensure that the extracted concepts are not due to noise. They must also be made of enough biological situations from $\mathcal{D}$ and few biological situations of $\mathcal{H}$ or vice versa. In other terms, the potentially interesting bi-sets (T, G) are concepts that verify the following constraints as well:

$$(|T_H| \geq 4 \ \wedge \ |T_D| \leq 2 \ \wedge \ |G| \geq \gamma) \tag{1}$$
$$\vee \, (|T_D| \geq 4 \ \wedge \ |T_H| \leq 2 \ \wedge \ |G| \geq \gamma) \tag{2}$$

where T_H and T_D are the subsets of T that concern respectively the biological situations from $\mathcal{H}$ (healthy individuals) and the ones from $\mathcal{D}$ (for diabetic patients).

Notice that this constraint which is the disjunction of Equation (1) and Equation (2) is a disjunction of a conjunction of monotonic and anti-monotonic constraints on $\mathcal{L}_\mathcal{O}$ and $\mathcal{L}_\mathcal{P}$. Using D-Miner, we push the monotonic ones.

4.2 Comparison with other algorithms

The execution time for computing frequent closed sets on this biological data with Closet, Ac-Miner, Charm and D-Miner is shown on Figure 11.

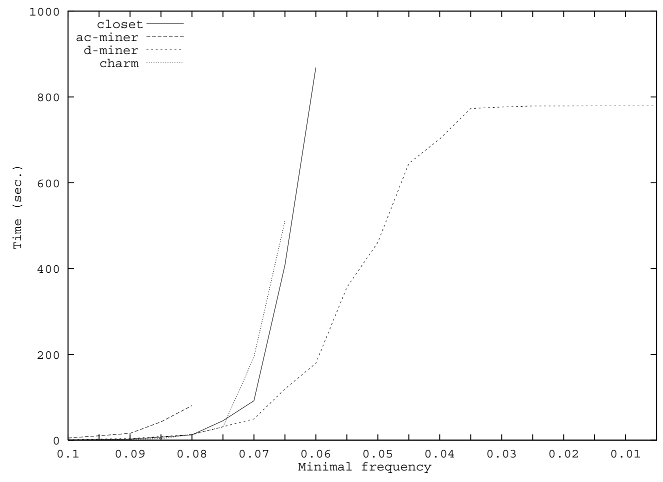

Figure 11: Extended microarray data analysis

D-Miner is the only algorithm which succeeds to extract all the concepts. Notice that the data are particular: there are very few frequent concepts before the relative frequency threshold 0.1 on genes (5 534 concepts) and then the number of concepts increases very fast (at the lowest frequency threshold, there are more than 5 millions of concepts). In this context, extracting putative interesting concepts needs for a very low frequency threshold, otherwise almost no concepts are provided. Consequently D-MINER is much better than the other algorithms for this application because it succeeds to extract concepts when the frequency threshold is lower than 0.06 whereas it is impossible with the others.

End users, e.g., biologists, are generally interested in a rather small subset of the extracted collections. These subsets can be specified by means of user-defined constraints that can be checked afterwards (post-processing) on the whole collection of concepts. It is clear however that this approach leads to tedious or even impossible post-processing phases when, e.g., more than 5 millions of concepts are computed (almost 500M bytes of patterns). It clearly motivates the need for constraint-based mining of concepts.

4.3 Extraction of concepts under constraints

Let us first extract the concepts which contain at least 4 among the 5 situations associated to the diabetic patients and at least γ genes. The first constraint is monotonic on $\mathcal{L}_{\mathcal{O}}$ whereas the second one is monotonic on $\mathcal{L}_{\mathcal{P}}$. Figure 12 presents the number of concepts verifying the constraints when γ vary.

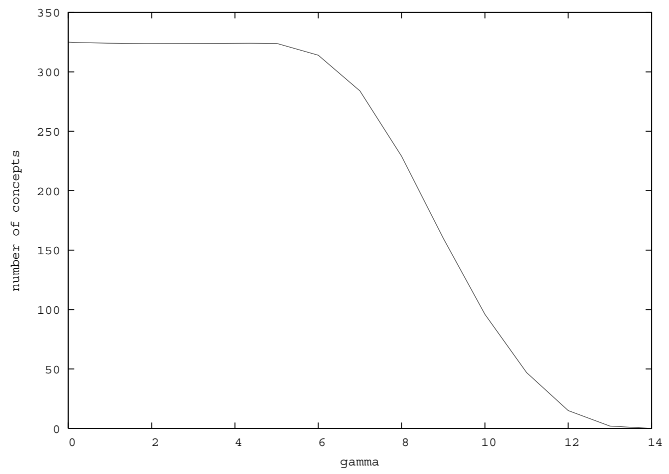

Figure 12: Number of concepts w.r.t. the minimum number of genes γ for $|T_D| \geq 4$

Using these constraints, we can reduce drastically the number of extracted concepts while selecting relevant ones according to end users. However, the small number of biological situations associated to the diabetic patients in our data set strongly constrain the size of the extracted collection: only 323 concepts satisfy $\mathcal{C}_t(\mathbf{r}, 4, T_D)$. Thus, in this data set, this single constraint appears to be quite selective and the extraction can be performed with any frequent closed set extraction algorithm using the constraint $\mathcal{C}_t(\mathbf{r}, 4, T_D)$. 234 concepts are obtained and can be post-processed in order to check constraint $\mathcal{C}_g(\mathbf{r}, \gamma, G)$ afterwards.

Notice however that, in other data sets, using simultaneously these two constraints can make the extraction possible whereas the extraction using only one of the two constraints and a post-processing strategy can be intractable. To illustrate this point, let us consider two constraints $\mathcal{C}_t(\mathbf{r}, \gamma_1, T)$ and $\mathcal{C}_g(\mathbf{r}, \gamma_2, G)$ on $\mathcal{L}_\mathcal{O}$ and $\mathcal{L}_\mathcal{P}$. In Figure 13, we plot the number of concepts obtained when γ_1 and γ_2 vary.

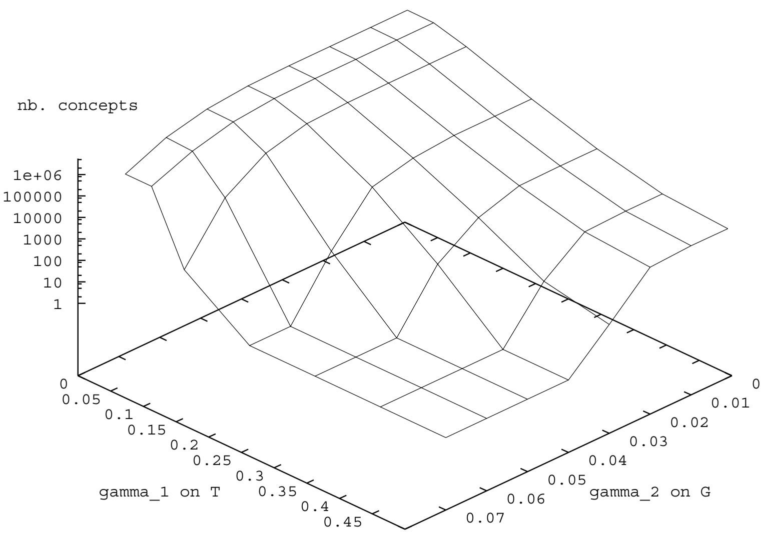

Figure 13: Extended microarray data analysis (2)

It appears that using only one of the two constraints does not reduce significantly the number of extracted concepts (see values when $\gamma_1 = 0$ or $\gamma_2 = 0$). However, when we use simultaneously the two constraints, the size of the concept collection decreases strongly (the surface of the values forms a basin). For example, the number of concepts verifying the constraints $\mathcal{C}_g(\mathbf{r}, 10, G)$ and $\mathcal{C}_t(\mathbf{r}, 21, T)$ is 142 279. The number of concepts verifying the constraints $\mathcal{C}_g(\mathbf{r}, 10, G)$ and $\mathcal{C}_t(\mathbf{r}, 0, T)$ is 5 422 514. The number of concepts verifying the constraints $\mathcal{C}_g(\mathbf{r}, 0, G)$ and $\mathcal{C}_t(\mathbf{r}, 21, T)$ is 208 746. The gain when using simultaneously both constraints is very significant.

The aim of our work with biologists is to extract concepts in larger microarray data sets. In this context the use of conjunction of constraints on genes, biological situations and transcription factors appears to be crucial.

4.4 Biological interpretation of an extracted concept

To validate the biological relevancy of the bi-sets that can be extracted on our biological data, let us consider one of the concepts D-Miner has computed. It contains 17 genes Hs.2022, Hs.184, Hs.1435, Hs.21189, Hs.25001, Hs.181195, Hs.295112, Hs.283740, Hs.12492, Hs.2110, Hs.139648, Hs.119, Hs.20993, Hs.89868, Hs.73957, Hs.57836 and Hs.302649 which are all potentially regulated by the same 8 transcription factors (c-Ets-, MZF1, Ik-2, CdxA, HSF2, USF, v-Myb, AML-1a). This association holds for 4 healthy subjects. This concept is very interesting as it contains genes which are either up-regulated or down-regulated after insulin stimulation, based on the homology of their promotor DNA sequences. In other terms, common transcription factors can regulate these genes either positively or negatively. Moreover this association does not occur in the muscle of the type 2 diabetic patients. Using traditional tools (e.g., the well known Eisen clustering tool [12]), the biologist would separate these genes into two groups: one containing the over-expressed gene, and the other the down-regulated ones. However, it is well-accepted that following an external cell stimulus, transcription factors regulate positively a set of genes and at the same time, down-regulated other sets of genes. To build the regulatory networks biologists need tools like D-Miner to extract concepts containing genes regulated by the same set of transcription factors. Among the 12 genes, one has already been suspected to be linked with insulin resistance, a pathological state associated with diabetes (Hs.73957, the RAS-associated protein RAB5A [23]). Finally, some genes among the 12 have unknown functions because the human genome is not fully annoted. Based on the results we obtained from D-Miner we can now group these genes with other genes having common regulatory behaviors during the type 2 diabetes.

5 Conclusion

Computing formal concepts has been proved useful in many application domains but remains extremely hard from dense boolean data sets like the one we have to process nowadays for gene expression data analysis. We have described an original algorithm that computes concepts under monotonic constraints. First, it can be used for closed set computation and thus concept discovery. Next, for difficult contexts, i.e., dense boolean matrices where none of the dimension is small, the analyst can provide monotonic constraints on both set components of desired concepts and the D-Miner algorithm can push them into the extraction process. We are now working on the biological validation of the extracted concepts. We have also to compare D-Miner with new concept lattice construction algorithms like [5].

Acknowledgment The authors thank Mohammed Zaki who has kindly provided his CHARM implementation. Jérémy Besson is funded by INRA. This work has been partially funded by the EU contract cInQ IST-2000-26469 (FET arm of the IST programme).

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In Proceedings ACM SIGMOD'93, pages 207–216,

Washington, D.C., USA, May 1993. ACM Press.

[2] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In Advances in Knowledge Discovery and Data Mining, pages 307–328. AAAI Press, 1996.

[3] Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, and L. Lakhal. Mining frequent patterns with counting inference. SIGKDD Explorations, 2(2):66 – 75, Dec. 2000.

[4] C. Becquet, S. Blachon, B. Jeudy, J.-F. Boulicaut, and O. Gandrillon. Strong association rule mining for large gene expression data analysis: a case study on human SAGE data. Genome Biology, 12, 2002. See http://genomebiology.com/2002/3/12/research/0067.

[5] A. Berry, J.-P. Bordat, and A. Sigayret. Concepts can not affor to stammer. In Proceedings JIM'03, pages 25–35, Metz, France, September 2003.

[6] J. Besson, C. Robardet, and J.-F. Boulicaut. Constraint-based mining of formal concepts in transactional data. In Proceedings PaKDD'04, Sydney, Australia, May 2004. Springer-Verlag. To appear as a LNAI volume.

[7] J.-F. Boulicaut and A. Bykowski. Frequent closures as a concise representation for binary data mining. In Proceedings PaKDD'00, volume 1805 of LNAI, pages 62–73, Kyoto, JP, Apr. 2000. Springer-Verlag.

[8] J.-F. Boulicaut, A. Bykowski, and C. Rigotti. Approximation of frequency queries by mean of free-sets. In Proceedings PKDD'00, volume 1910 of LNAI, pages 75–85, Lyon, F, Sept. 2000. Springer-Verlag.

[9] J.-F. Boulicaut, A. Bykowski, and C. Rigotti. Free-sets: a condensed representation of boolean data for the approximation of frequency queries. Data Mining and Knowledge Discovery journal, 7(1):5–22, 2003.

[10] J. L. DeRisi, V. R. Iyer, and P. O. Brown. Exploring the metabolic and genetic control of gene expression on a genomic scale. Science, 278, 1997.

[11] S. Dudoit and Y. H. Yang. Bioconductor R packages for exploratory analysis and normalization of cDNA microarray data. In The Analysis of Gene Expression Data: Methods and Software, pages 25–35, Springer, New York, 2003.

[12] M. Eisen, P. Spellman, P. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. Proceedings National Academy of Science USA, 95:14863–14868, 1998.

[13] B. Ganter. Two basic algorithms in concept analysis. Technical report, Technisch Hochschule Darmstadt, Preprint 831, 1984.

[14] B. Jeudy and J.-F. Boulicaut. Optimization of association rule mining queries. Intelligent Data Analysis journal, 6:341–357, 2002.

[15] D. Jiang and A. Zhang. Cluster analysis for gene expression data: A survey. Available at citeseer.nj.nec.com/575820.html.

[16] R. Ng, L. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained associations rules. In Proceedings ACM SIGMOD'98, pages 13–24, Seattle, USA, May 1998. ACM Press.

[17] L. Nourine and O. Raynaud. A fast algortihm for building lattices. Information Processing Letters, 71:190–204, 1999.

[18] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Efficient mining of association rules using closed itemset lattices. Information Systems, 24(1):25–46, Jan. 1999.

[19] J. Pei, J. Han, and R. Mao. CLOSET an efficient algorithm for mining frequent closed itemsets. In Proceedings ACM SIGMOD Workshop DMKD'00, Dallas, USA, May 2000.

[20] F. Rioult, J.-F. Boulicaut, B. Crémilleux, and J. Besson. Using transposition for pattern discovery from microarray data. In Proceedings ACM SIGMOD Workshop DMKD'03, pages 73–79, San Diego, USA, June 2003.

[21] F. Rioult, C. Robardet, S. Blachon, B. Crémilleux, O. Gandrillon, and J.-F. Boulicaut. Mining concepts from large SAGE gene expression matrices. In Proceedings KDID'03 co-located with ECML-PKDD'03, pages 107–118, Cavtat-Dubrovnik, Croatia, September 22 2003.

[22] C. Robardet and F. Feschet. Comparison of three objective functions for conceptual clustering. In Proceedings PKDD 2001, number 2168 in LNAI, pages 399–410. Springer-Verlag, September 2001.

[23] S. Rome, K. Clément, R. Rabasa-Lhoret, E. Loizon, C. Poitou, G. S. Barsh, J.-P. Riou, M. Laville, and H. Vidal. Microarray profiling of human skeletal muscle reveals that insulin regulates 800 genes during an hyperinsulinemic clamp. Journal of Biological Chemistry, May 2003. 278(20):18063-8.

[24] V. Velculescu, L. Zhang, B. Vogelstein, and K. Kinzler. Serial analysis of gene expression. Science, 270:484–487, 1995.

[25] R. Wille. Restructuring lattice theory: an approach based on hierarchies of concepts. In I. Rival, editor, Ordered sets, pages 445–470. Reidel, 1982.

[26] M. J. Zaki and C.-J. Hsiao. CHARM: An efficient algorithm for closed itemset mining. In Proceedings SIAM DM'02, Arlington, USA, April 2002.

[27] H. Zhang, Y. Ramanathan, P. Soteropoulos, M. Recce, and P. Tolias. Ez-retrieve: a web-server for batch retrieval of coordinate-specified human dna sequences and underscoring putative transcription factor-binding sites. Nucleic Acids Res., 30:–121, 2002.

A Proofs

We first prove that all tree leaves are different from each other in Section A.2. Thus the leaves can be considered as a set denoted by L. In a second step we prove that L is equal to the collection of concepts denoted by C:

- We prove in Section A.3 that $C \subseteq L$
- We prove in Section A.4 that $L \subseteq C$

Section A.1 is a preliminary step to facilitate these proofs.

A.1 All tree leaves are 1-rectangles

Let (L_X, L_Y) be a D-Miner tree leaf. Let us demonstrate that $\forall i \in L_X$, $\forall j \in L_Y$, $(i,j) \in \mathbf{r}$. We suppose first the contrary, that is, it exists $i \in L_X$ and $j \in L_Y$ such that $(i,j) \notin \mathbf{r}$. Thus, following definition of H, it exists $(A,B) \in H$ such that $i \in A$ and $j \in B$. By definition of the algorithm, each element of H is used on the path from the root to each tree leaf. Let (A_X, A_Y) be the ancestor of (L_X, L_Y) cut by (A,B). The order relation implies that $L_X \subseteq A_X$ and $L_Y \subseteq A_Y$ and thus $i \in A_X$ and $j \in A_Y$. Consequently, it is cut by (A,B). The sons of (A_X, A_Y), if they exist, are $(A_X \setminus A, A_Y)$ and $(A_X, A_Y \setminus B)$. Thus $L_X \subseteq A_X \setminus A \subseteq A_X \setminus \{i\}$ and $L_Y \subseteq A_Y \setminus B \subseteq A_Y \setminus \{j\}$. Given that it contradicts the hypothesis,

$$\forall i \in L_X,\ \forall j \in L_Y,\ (i,j) \in \mathbf{r}$$

A.2 All tree leaves are different

Let (L_{X_1}, L_{Y_1}) and (L_{X_2}, L_{Y_2}) be two D-Miner tree leaves. Let (A_X, A_Y) be the smallest common ancestor of these two leaves and (A,B) be the H element used to cut it. We suppose, without loss of generality, that (L_{X_1}, L_{Y_1}) is the left son of (A_X, A_Y) and that (L_{X_2}, L_{Y_2}) is its right son. By definition of the algorithm, each of the left successors of (A_X, A_Y) does not contain elements of A thus

$$L_{X_1} = A_X \setminus A$$

Each of the right successors of (A_X, A_Y) contains all elements of A and thus

$$A \subset L_{X_2}$$

Consequently, $L_{X_1} \neq L_{X_2}$ and $(L_{X_1}, L_{Y_1}) \neq (L_{X_2}, L_{Y_2})$.

A.3 $C \subseteq L$

Let us demonstrate that for each concept (C_X, C_Y), there exists a path from the root node $(\mathcal{O}, \mathcal{P})$ to this concept:

$$(\mathcal{O}, \mathcal{P}),\ (X_1, Y_1),\ \cdots,\ (X_{n-1}, Y_{n-1}),\ (C_X, C_Y)$$

- In a first step we recursively demonstrate that the path is such that:

  $$\begin{array}{c}(X_0, Y_0),\ (X_1, Y_1),\ \cdots,\ (X_{n-1}, Y_{n-1}),\ (L_X, L_Y) \\ \text{with } (L_X, L_Y) \supseteq (C_X, C_Y) \text{ and } (\mathcal{O}, \mathcal{P}) = (X_0, Y_0)\end{array}$$

  We denote by $H_{L_k}(X_k, Y_k)$ the set of elements of H which leads to a left branch on the path from (X_0, Y_0) to (X_k, Y_k). The recurrence hypotheses are the following:

  (1) $(C_X, C_Y) \subseteq (X_k, Y_k)$

  (2) $\forall (A, B) \in H_{L_k},\ C_Y \cap B \neq \emptyset$

  Proof:

  Base By definition of formal concepts, each concept (C_X, C_Y) is included in the root node, i.e., $C_X \subseteq X_0$ and $C_Y \subseteq Y_0$. Hypothesis (1) is thus verified for rank 0. Furthermore, $H_{L_0} = \emptyset$ and thus hypothesis (2) is verified for rank 0 as well.

  Induction Let us suppose that hypotheses (1) and (2) are verified for rank $k-1$ and let us prove that they are still true at rank k when the node (X_{k-1}, Y_{k-1}) is cut using $(A, B) \in H$.

  ∗ If $X_{k-1} \cap A = \emptyset$ or $Y_{k-1} \cap B = \emptyset$ then $(X_k, Y_k) = (X_{k-1}, Y_{k-1})$ and $H_{L_k} = H_{L_{k-1}}$. It means that (X_k, Y_k) verifies hypotheses (1) and (2).

  ∗ Otherwise, given the definition of formal concepts, only one of the two following conditions is verified:

    · If $C_X \cap A \neq \emptyset$ and $C_Y \cap B = \emptyset$ then the right son $(X_{k-1}, Y_{k-1} \setminus B)$, a subset of (X_{k-1}, Y_{k-1}), contains the concept (C_X, C_Y) because $C_Y \subseteq Y_{k-1} \setminus B$. This son is generated by the algorithm because hypothesis (2) is verified for the rank $k-1$. $H_{L_k} = H_{L_{k-1}}$ ((A, B) does not belong to H_{L_k} because (X_k, Y_k) is the right son of (X_{k-1}, Y_{k-1})), and thus hypothesis (1) is verified.

· If $C_X \cap A = \emptyset$ and $C_Y \cap B \neq \emptyset$ then $(X_{k-1} \setminus A, Y_{k-1})$, the left son, contains (C_X, C_Y). This left son is always generated by the algorithm. Hypothesis (1) is thus verified for rank k. Furthermore, as $C_Y \cap B \neq \emptyset$ and $H_{L_k} = H_{L_{k-1}} \cup (A, B)$, hypothesis (2) is also verified.

- We have proved that $(C_X, C_Y) \subseteq (L_X, L_Y)$. Let us now prove that $(L_X, L_Y) = (C_X, C_Y)$. In Section A.1, we have proved that each leaf is a 1-rectangle. Thus a leaf is at the same time a superset of (C_X, C_Y) and a 1-rectangle and thus $(C_X, C_Y) = (L_X, L_Y)$.

A.4 $L \subseteq C$

We have proved in Section A.1 that each leaf is a 1-rectangle. Then we have proved that each concept is associated to a leaf. We now prove that each leaf is a concept.

A leaf (L_X, L_Y) is a concept if and only if

$$\begin{cases} (L_X, L_Y) \text{ is a 1-rectangle} \\ \nexists x \in \mathcal{O} \setminus L_X \textit{ such that } \forall y \in L_Y,\ (x, y) \in \mathbf{r} \ (1) \\ \nexists y \in \mathcal{P} \setminus L_Y \textit{ such that } \forall x \in L_X,\ (x, y) \in \mathbf{r} \ (2) \end{cases}$$

Let us now prove by contradiction Property (1) and then Property (2).

- Assume that it exists $x \in \mathcal{O} \setminus L_X$ such that $\forall y \in L_Y,\ (x, y) \in \mathbf{r}$. Let (x, B) be an element of H. This 0-rectangle exists because

  - either x is in relation with all the elements of $\mathcal{P}$ and, in that case, it leads to a contradiction because if $x \notin H$ then all the tree nodes contains x while we assume $x \notin L_X$.
  - or, it exists $B \subseteq \mathcal{P}$ such that $\forall y \in B,\ (x, y) \notin \mathbf{r}$. (x, B) is, in this case, unique given the computation of H.

  Let (A_X, A_Y) be the ancestor of (L_X, L_Y) before being cut by (x, B). (L_X, L_Y) has to be a descendant of $(A_X \setminus x, A_Y)$ which is the left son of (A_X, A_Y). This is due to the fact that $x \notin L_X$ (see Figure 14). By the construction of the tree, the left son of (A_X, A_Y) contains B but, by hypothesis ($\forall y \in L_Y,\ (x, y) \in \mathbf{r}$), L_Y does not contain B. To get that one of the descendants of $(A_X \setminus x, A_Y)$ does not contain any element of B, it is needed that one of these descendants is a right son of another one. Let us denote by (A', B') the 0-rectangle which leads to a right son without elements from B. To enable this cutting, it is needed that $B \cap B' \neq \emptyset$. Consequently, each descendant contains at least an element of B and it contradicts the fact that $L_Y \cap B = \emptyset$.

- Let us assume that it exists $y \in \mathcal{P} \setminus L_Y$ such that $\forall z \in L_X,\ (z, y) \in \mathbf{r}$. Let (A_{X_1}, A_{Y_1}) and (A_{X_2}, A_{Y_2}) be two ancestors of (L_X, L_Y) such that (A_{X_1}, A_{Y_1})

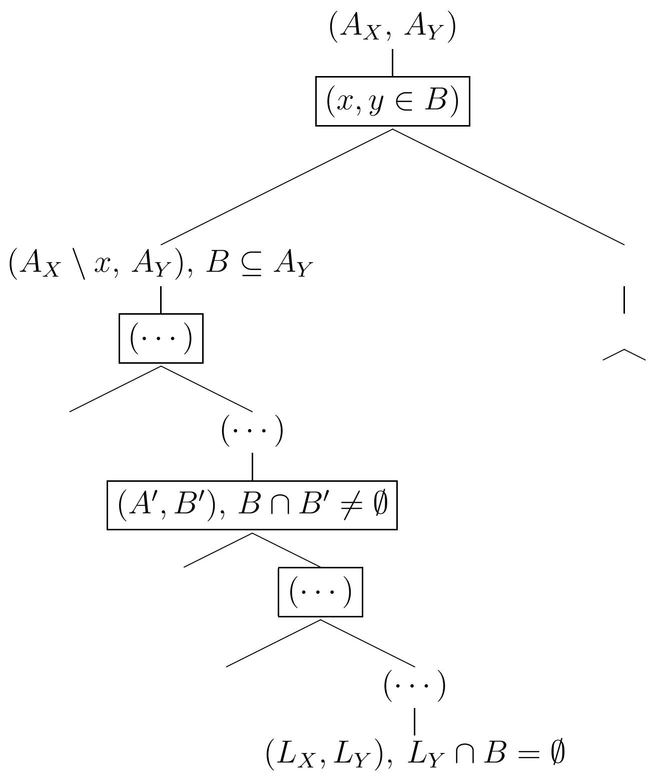
$(A_X,\, A_Y)$
$\boxed{(x, y \in B)}$
$(A_X \setminus x,\, A_Y),\, B \subseteq A_Y$
$\boxed{(\cdots)}$
$(\cdots)$
$\boxed{(A', B'),\, B \cap B' \neq \emptyset}$
$\boxed{(\cdots)}$
$(\cdots)$
$(L_X, L_Y),\, L_Y \cap B = \emptyset$

Figure 14: First illustration: $L_Y \subseteq A_Y \wedge B \subseteq A_Y \wedge B \cap L_Y \neq \emptyset \Rightarrow \neg\,(B \cap L_Y = \emptyset)$

is the father of (A_{X_2}, A_{Y_2}), $y \in A_{Y_1}$ and $y \notin A_{Y_2}$. We denote by (x, B) the 0-rectangle which is used to cut (A_{X_1}, A_{Y_1}) (see Figure 15). Given the algorithm principle, these two ancestors exist and (A_{X_2}, A_{Y_2}) is the right son of (A_{X_1}, A_{Y_1}). Because $y \notin L_Y$,

$$(L_X, L_Y) \text{ is a descendant of } (A_{X_2}, A_{Y_2}) = (A_{X_1}, A_{Y_1} \setminus B)$$

By construction of H, x belongs only to one element of H. As (A_{X_1}, A_{Y_1}) has been cut by (x, B), $x \in A_{X_1}$. Furthermore (A_{X_2}, Y_{Y_2}) and all its descendants also contain x. Nevertheless, we have proved that (L_X, L_Y) is a descendant of (A_{X_2}, A_{Y_2}). Consequently,

$$x \in L_X$$

The hypothesis $\forall z \in L_X,\ (z, y) \in \mathbf{r}$ is contradicted by the fact that $(x, y) \notin \mathbf{r}$ (x and y belong to the same 0-rectangle).

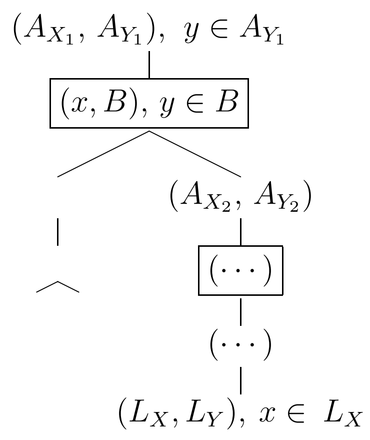
$(A_{X_1},\, A_{Y_1}),\ y \in A_{Y_1}$
$(x, B),\, y \in B$
$(A_{X_2},\, A_{Y_2})$
$(\cdots)$
$(\cdots)$
$(L_X, L_Y),\ x \in\ L_X$

Figure 15: Second illustration: $x \in L_X \wedge (x, y) \notin \mathbf{r} \Rightarrow \neg\, (\forall x \in L_X, (x, y) \in \mathbf{r})$

B Experimentations on synthetic data sets

We have discussed in Section 3.3 our complementary experimental validation of D-Miner on several artificial data sets. We generated 60 dense data sets by varying the number of columns (300, 700 and 900 columns), the number of lines (100 and 300 lines) and the density of "1" values (15 % and 35 %). For each combination of parameters, we generated 5 data sets. Table 2 (resp. Table 3 provides the mean (column MEAN) and the standard deviation (column SD) of execution time (in seconds) used by each algorithm on the 5 data sets with 15% density (resp. 35% density). The parameter σ denotes the used frequency threshold. For each data set and each value of σ, we also report the number of frequent concepts (line FrC).

σ	♯ columns	300				700				900			
	♯ lines	100		300		100		300		100		300	
		MEAN	SD	MEAN	SD	MEAN	SD	MEAN	SD	MEAN	SD	MEAN	SD
0,1	CLOSET	0,02	0	0,24	0,1	0,04	0,01	0,5	0,03	0,04	0,01	0,7	0,02
	AC-MINER	0,02	0	0,44	0,03	0,03	0,01	0,56	0,03	0,03	0	0,6	0,04
	D-MINER	0,01	0,01	0,11	0,01	0,01	0,01	0,12	0,01	0,02	0,01	0,15	0,01
	CHARM	0,01	0,01	0,03	0,01	0	0,1	0,06	0,01	0,01	0	0,06	0,01
	FRC	205	21	2631	229	199	18	2320	78	197	17	2254	53
0,01	CLOSET	1,36	0,14	451,2	64,77	1,18	0,14	475,76	32,85	1,16	0,14	384,75	25,63
	AC-MINER	1,32	0,12	–	–	1,82	0,19	–	–	1,96	0,18	–	–
	D-MINER	0,89	0,05	112,6	11,6	0,73	0,06	149,7	9,8	0,86	0,08	154,5	9,7
	CHARM	0,6	0,06	–	–	0,35	0,06	–	–	0,31	0,03	–	–
	FRC	$4{,}3\ 10^4$	$3{,}7\ 10^3$	$2{,}8\ 10^6$	$2{,}8\ 10^5$	$4\ 10^4$	$4{,}6\ 10^3$	$4{,}7\ 10^6$	$2{,}5\ 10^5$	$3{,}7\ 10^4$	$4{,}4\ 10^3$	$3{,}9\ 10^6$	$1{,}9\ 10^5$
0,001	CLOSET	3,69	0,31	–	–	20,71	2,07	–	–	34,51	3,74	–	–
	AC-MINER	4,79	0,43	–	–	19,86	1,98	–	–	30,51	2,94	–	–
	D-MINER	1,35	0,06	128,31	12,42	6,19	0,4	–	–	11,89	0,81	–	–
	CHARM	1,84	0,2	–	–	11,75	1,44	–	–	19,92	2,75	–	–
	FRC	$5{,}3\ 10^4$	$3{,}8\ 10^3$	$2{,}9\ 10^6$	$2{,}8\ 10^5$	$2{,}4\ 10^5$	$1{,}9\ 10^4$	–	–	$3{,}6\ 10^5$	$3{,}1\ 10^4$	–	–

Table 2: Execution time on data sets with 15% of density (– iff time $\geq$ 10 min)

σ	$\sharp$ columns	300				700				900			
	$\sharp$ lines	100		300		100		300		100		300	
		MEAN	SD	MEAN	SD	MEAN	SD	MEAN	SD	MEAN	SD	MEAN	SD
0,1	CLOSET	14,57	4.74	–	–	14,02	3,45	–	–	14,08	3,06	–	–
	AC-MINER	16,69	4,28	–	–	22,01	4,26	–	–	24,34	4,51	–	–
	D-MINER	2,53	0,57	–	–	2,36	0,41	–	–	2,79	0,45	–	–
	CHARM	5,86	3,65	–	–	3,53	0,74	–	–	3,34	0,78	–	–
	FRC	$2{,}7\ 10^5$	$7{,}3\ 10^4$	–	–	$2{,}5\ 10^5$	$5\ 10^4$	–	–	$2{,}4\ 10^5$	$4{,}7\ 10^4$	–	–
0,01	CLOSET	–	–	–	–	–	–	–	–	–	–	–	–
	AC-MINER	–	–	–	–	–	–	–	–	–	–	–	–
	D-MINER	544,41	35,29	–	–	–	–	–	–	–	–	–	–
	CHARM	–	–	–	–	–	–	–	–	–	–	–	–
	FRC	$3{,}7\ 10^7$	$3\ 10^6$	–	–	–	–	–	–	–	–	–	–
0,001	CLOSET	–	–	–	–	–	–	–	–	–	–	–	–
	AC-MINER	–	–	–	–	–	–	–	–	–	–	–	–
	D-MINER	550,6	34,98	–	–	–	–	–	–	–	–	–	–
	CHARM	–	–	–	–	–	–	–	–	–	–	–	–
	FRC	$3{,}7\ 10^7$	$3\ 10^6$	–	–	–	–	–	–	–	–	–	–

Table 3: Execution time on data sets with 35% of density (– iff time $\geq$ 20 min)