Constraint-Based Mining of Formal Concepts in Transactional Data

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

[1] Institut National des Sciences Appliquées de Lyon
LIRIS CNRS FRE 2672, Bâtiment Blaise Pascal
F-69621 Villeurbanne cedex, France
{jeremy.besson,jean-francois.boulicaut}@insa-lyon.fr
[2] Institut National des Sciences Appliquées de Lyon
PRISMA, Bâtiment Blaise Pascal
F-69621 Villeurbanne cedex, France
celine.robardet@insa-lyon.fr

Abstract. We are designing new data mining techniques on boolean contexts to identify a priori interesting concepts, i.e., closed sets of objects (or transactions) and associated closed sets of attributes (or items). We propose a new algorithm D-Miner for mining concepts under constraints. We provide an experimental comparison with previous algorithms and an application to an original microarray dataset for which D-Miner is the only one that can mine all the concepts.

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

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 denotes 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}.

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. 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 the various Apriori-like algorithms that have been designed during the last decade [2]. When the data are dense and highly-correlated, these algorithms fail but the so-called condensed representations of the frequent itemsets can be computed. For instance, efficient algorithms can compute the frequent closed sets from which every frequent set

H. Dai, R. Srikant, and C. Zhang (Eds.): PAKDD 2004, LNAI 3056, pp. 615–624, 2004.

	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

Fig. 1. Example of a boolean context $\mathbf{r}_1$

and its frequency can be derived without accessing the data [10,6,11,3,14]. Other important applications concern datasets with only a few transactions, e.g., for typical gene expression data where items denote gene expression properties in biological situations. 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 [12].

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 belong to each transaction of T (1-rectangles). It is interesting to constrain further bi-set components to be closed sets (also called maximal 1-rectangles or concepts [13]). Other constraints, e.g., minimal and maximal frequency, can be used as well.
We propose an original algorithm called D-Miner that computes concepts under constraints. It works differently from other concept discovery algorithms (see, e.g., [8,4,9]) and (frequent) closed set computation algorithms. D-Miner can be used in dense boolean datasets when the previous algorithms generally fail. Thanks to an active use of the constraints, it enlarges the applicability of concept discovery for matrices whose none of the dimensions is small. Section 2 contains the needed definitions and a presentation of D-Miner. Section 3 provides an experimental validation. Finally, Section 4 is a short conclusion.

2 D-Miner

Let $\mathcal{O}$ denote a set of objects or transactions and $\mathcal{P}$ denote 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 object i.

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

Definition 1. *A bi-set (T, G) is a 1-rectangle in $\mathbf{r}$ iff $\forall t \in T$ and $\forall g \in G$ then $(t, g) \in \mathbf{r}$. 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 2. *(Concept) A bi-set* (T, G) *is a concept in* $\mathbf{r}$ *iff* (T, G) *is a 1-rectangle and* $\forall T' \subseteq \mathcal{O} \backslash T$, $(T \cup T', G)$ *is not a 1-rectangle and* $\forall G' \subseteq \mathcal{P} \backslash G$, $(T, G \cup G')$ *is not a 1-rectangle.*

Notice that, by construction, both sets of a concept are closed sets and any algorithm that computes closed sets can be used for concept discovery [12].
Given Figure 1, $(\{t_1, t_2, t_3\}, \{g_1, g_2\})$ is a 1-rectangle in $\mathbf{r}_1$ but it is not a concept. Twelve bi-sets are concepts in $\mathbf{r}_1$. Two of them are $(\{t_1, t_2, t_3, t_5\}, \{g_1, g_3\})$ and $(\{t_2, t_3\}, \{g_1, g_2, g_3, g_4, g_9, g_{10}\})$. Interesting data mining processes on transactional data can be formalized as the computation of bi-sets whose set components satisfy combinations of primitive constraints.

Definition 3. *(Monotonic and anti-monotonic constraints) Given* $\mathcal{L}$ *a collection of sets, a constraint* $\mathcal{C}$ *is said anti-monotonic w.r.t.* $\subseteq$ *iff* $\forall \alpha, \beta \in \mathcal{L}$ *such that* $\alpha \subseteq \beta$, $\mathcal{C}(\beta) \Rightarrow \mathcal{C}(\alpha)$. $\mathcal{C}$ *is said monotonic w.r.t.* $\subseteq$ *iff* $\forall \alpha$, $\beta \in \mathcal{L}$ *such that* $\alpha \subseteq \beta$, $\mathcal{C}(\alpha) \Rightarrow \mathcal{C}(\beta)$.

In A-PRIORI like algorithms, the minimal frequency constraint (on $\mathcal{L}_\mathcal{P}$) is used to prune the search space. This constraint is anti-monotonic w.r.t. $\subseteq$ on $\mathcal{L}_\mathcal{P}$. This constraint can be considered as monotonic on $\mathcal{L}_\mathcal{O}$ because when a set of items is larger, the associated set of transactions is smaller.

Definition 4. *(Specialization relation) Our specialisation 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* $G_1 \subseteq G_2$.

We generalize the frequency constraints on this partial order $\leq$.

Definition 5. *(Frequency constraints on concepts) A concept* (T, G) *satisfies a 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* $\mathcal{L}_\mathcal{O} \times \mathcal{L}_\mathcal{P}$.

For example, the set of concepts (T,G) satisfying $\mathcal{C}_g(\mathbf{r}_1, 4, G) \wedge \mathcal{C}_t(\mathbf{r}_1, 3, T)$ (a conjonction of monotonic constraints) is $\{(\{g_1, g_2, g_3, g_4\}, \{t_1, t_2, t_3\})\}$.

2.1 D-Miner Principle

D-MINER is a new algorithm for extracting concepts (T, G) under constraints. It builds the sets T and G and it uses monotonic constraints simultaneously on $\mathcal{L}_O$ and $\mathcal{L}_P$ to reduce the search space. 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 such that it is a partition of the false values (0) of the boolean matrix, i.e., $\forall g \in \mathcal{P}$ and $\forall t \in \mathcal{O}$ such that $(t, g) \notin \mathbf{r}$, it exists one and only one element (X, Y) of H such that $t \in X$ and $g \in Y$. The elements of H are called *cutters*. 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 attribute valued by 0 in this line. Time complexity for computing H is in $\mathcal{O}(n \times m)$ where n and m are the dimensions of the matrix. 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.

D-MINER starts with the couple $(\mathcal{O},\mathcal{P})$ and then splits it recursively using the elements of H until H is empty and consequently 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 $(X \backslash a, Y)$ and the right son by $(X, Y \backslash b)$. Recursive splitting leads to all the concepts, i.e., the maximal 1-rectangles (see Example 1) but also some non-maximal ones (see Example 2). We consider in Example 3 how to prune them to obtain all the concepts and only the concepts.

We now provide examples of D-MINER executions. The use of monotonic constraints on $\mathcal{L}_\mathcal{O}$ and $\mathcal{L}_\mathcal{P}$ is presented later. Notice that for clarity, sets like $\{g_1, g_2\}$ are written g_1g_2.

Example 1. Assume $\mathcal{O} = \{t_1, t_2, t_3\}$ and $\mathcal{P} = \{g_1, g_2, g_3\}$. $\mathbf{r}_2$ is defined in Table 1 (left). Figure 2 (left) illustrates D-MINER execution. We get 4 1-rectangles that are the 4 concepts for this boolean context.

Table 1. Contexts $\mathbf{r}_2$ for Example 1 (left) and $\mathbf{r}_3$ for Examples 2 and 3 (right)

	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

Example 2. Assume now $\mathbf{r}_3$ as given in Table 1 (right). Computing H provides $\{(t_1, g_1g_2),\ (t_2, g_2),\ (t_3, g_1g_2)\}$. Figure 2 (right) illustrates D-MINER execution. Some bi-sets are underlined and this will be explained in Example 3.

From Figure 2 (right), 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 solve this problem, let us 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 couple (X, Y) is split by a cutter $(a, b) \in H$, then $(X \backslash a, Y)$ (the left son) and $(X, Y \backslash b)$ (the

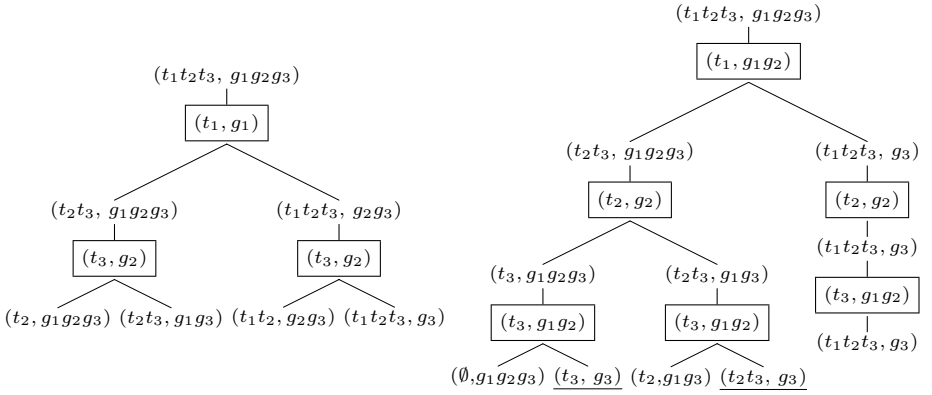

Fig. 2. Concept construction on $\mathbf{r}_2$ (left) and on $\mathbf{r}_3$ (right)

right son) are generated. By construction of H $(a, Y \backslash b)$ is a 1-rectangle which is not necessarily maximal. If a concept (C_X, C_Y) exists in $\mathbf{r}[X \backslash a, Y]$ such that $C_Y \cap b = \emptyset$ then $(C_X \cup a, C_Y)$ is a concept in $\mathbf{r}[X, Y]$. However $(C_X \cup a, C_Y)$ is a concept in $\mathbf{r}[X, Y \backslash b]$ and consequently would be a son of the right son of (X, Y) (see Figure 3). To avoid these non-maximal 1-rectangles, we have to enforce that the property $b \cap Y \neq \emptyset$ is always satisfied for all the previously used left-cutters.

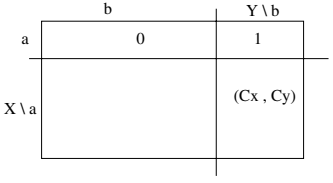

Fig. 3. Non-maximal 1-rectangle occurrence

Property 1: Let (X, Y) be a leaf of the tree and $H_L(X, Y)$ be the set of cutters associated to the left branches of the path from the root to (X, Y). Then (X, Y) is a concept iff it contains at least one item of each element of $H_L(X, Y)$. 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 is called later the left cutting constraint.
This has been formally studied in [5] that contains correctness and completeness proofs for D-Miner.

Example 3. We take the context used for Example 2 (see Table 1 on the right). 1-rectangles (t_3, g_3) and (t_2t_3, g_3) are pruned using Property 1. (t_3, g_3) comes

from the left cutting of $(t_1t_2t_3,g_1g_2g_3)$ and then the left cutting of $(t_3, g_1g_2g_3)$. The items of (t_3, g_3) must contain at least one item of $\{g_1, g_2\}$ and 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. Nodes that are underlined in Figure 2 (right) are pruned.

2.2 Algorithm

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

D-Miner is a depth-first method which generates couples 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). For instance, we can push the constraint $\mathcal{C}((T, G)) \equiv \mathcal{C}_t(T) \wedge \mathcal{C}_g(G)$ where (T, G) is a bi-set, $\mathcal{C}_t(T) \equiv |T| \geq 5$, and $\mathcal{C}_g(G) \equiv |G| \geq 4$.

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 couple (X, Y) with the first cutter $H[i]$ that satisfies the following constraints. First, (X, Y) must have a non empty intersection with $H[i]$. If it is not the case, *cutting* is called with the next cutter. Before cutting (X, Y) in $(X\backslash a, Y)$, we have to check the monotonic constraint on $X\backslash a$ (denoted $\mathcal{C}_t(X\backslash a)$) 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 second cutting of (X, Y), two constraints have to be checked. First the monotonic constraint on $Y\backslash b$ (denoted $\mathcal{C}_g(Y\backslash b)$) is checked. Therefore, D-MINER constructs first an element (X, Y) and then reduces simultaneously X and Y to have the collection of concepts derived from (X, Y). Secondly, monotonic constraints can be applied on X and Y to prune the search space: if $\alpha \leq \beta$ and $\neg\, \mathcal{C}(\beta)$ then $\neg\, \mathcal{C}(\alpha)$.

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. H must be sorted by decreasing order of size of the object components. Moreover, to reduce the size of H, the cutters which have the same items are gathered: $\forall (a_1, b_1),\ (a_2, b_2) \in H,\ if\ b_1 = b_2\ then\ H = H\backslash\{(a_1, b_1), (a_2, b_2)\} \cup (a_1 \cup a_2, b_1)$. If $|\mathcal{P}| > |\mathcal{O}|$, we transpose the data matrix to obtain a set H of minimum size. The symmetry of our extractor on $\mathcal{L}_O$ and $\mathcal{L}_P$ allows to transpose the matrix without loosing the possibility of using the constraints. Indeed, in some contexts where there are few objects and many items, we first perform a simple transposition like in [12].

Algorithm 1: D-MINER

Input : Database $\mathbf{r}$ with n lines and m columns, $\mathcal{O}$ 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 and $H_{size} = |H|$ are computed from $\mathbf{r}$;
$\mathcal{Q} \leftarrow$ cutting($(\mathcal{O}, \mathcal{P})$, H, 0, H_{size}, 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 number of iterations, H_{size} the size of H, H_L a set of precedent cutters in left cuttings, $\mathcal{C}_t$ monotonic constraint on $\mathcal{O}$, $\mathcal{C}_g$ 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 $(i \leq H_{size} - 1)$ // i-th cutter is selected
  If $((a \cap X = \emptyset)$ or $(b \cap Y = \emptyset))$
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup cutting((X, Y), H, i + 1, H_{size}, H_L)$
  Else
    If ($\mathcal{C}_t(X \backslash a)$ is satisfied)
      $H_L \leftarrow H_L \cup (a, b)$
      $\mathcal{Q} \leftarrow \mathcal{Q} \cup cutting((X \backslash a, Y), H, i + 1, H_{size}, H_L)$
      $H_L \leftarrow H_L \backslash (a, b)$
    If ($\mathcal{C}_g(Y \backslash b)$ is satisfied $\wedge$ $\forall (a', b') \in H_L$, $b' \cap Y \backslash b \neq \emptyset$)
      $\mathcal{Q} \leftarrow \mathcal{Q} \cup cutting((X, Y \backslash b), H, i + 1, H_{size}, H_L)$
Else
  $\mathcal{Q} \leftarrow (X, Y)$
Return $\mathcal{Q}$

3 Experimental Validation

We compare the execution time of D-MINER with those of CLOSET [11], AC-MINER [6] and CHARM [14] in three datasets. CLOSET, CHARM and AC-MINER compute closed sets under a minimal frequency constraint, i.e., the frequent closed sets. Due to Galois connection, in a given dataset, the number of closed sets in $\mathcal{L}_O$ is the number of closed sets in $\mathcal{L}_P$. For a fair comparison, we transpose the matrices to have a smaller number of columns for CLOSET, AC-MINER and CHARM and to have a smaller number of lines for D-MINER. In the three first experiments, we compare the effectiveness of the four algorithms when computing the collection of closed sets under a minimal frequency constraint. We used Zaki's implementation of CHARM and Bykowski's implementations of AC-MINER and CLOSET [7]. In all the following figures, the minimal frequencies are relative frequencies.

First, we have studied the performance of D-Miner for computing the frequent closed sets from benchmark datasets available on line at IBM Almaden[1] and the UCI repository. All extractions have been performed on a Pentium III (450 MHz, 128 Mb). We have used the benchmark "Mushroom". Its derived boolean context contains 8 124 lines and 120 columns. The needed execution time (in seconds) to obtain the frequent closed sets is shown on Figure 4 (left).

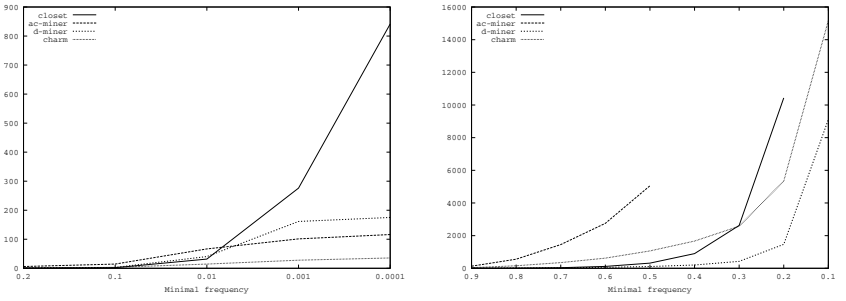

Fig. 4. Mushroom (left) and Connect4 (right)

The four algorithms can compute every closed set and thus all concepts (minimum frequency 0.0001) within a few minutes. Indeed, the lowest frequency threshold corresponds to at least 1 object. Once every closed set on one dimension is computed, the associated concept is obtained easily. 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 to obtain frequent closed sets is shown on Figure 4 (right). Only Charm and D-Miner can extract concepts with minimal frequency equals to 0.1 (10%). D-Miner is almost twice faster than Charm on this dataset.

We now provide an experimental validation on an original biological dataset that contains 104 lines and 304 columns. For the purpose of this paper, an important information is that its density is high: 17 % of the cells contain the value true. This is a gene expression dataset that can not be described further due to space limitation (see [5] for details). The execution time (in seconds) for computing frequent closed sets with Closet, Ac-Miner, Charm and D-Miner is shown on Figure 5 (left).

D-Miner is the only algorithm which succeeds in extracting all the concepts. These data are in fact very particular: there are very few concepts before the frequency 0.1 (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

[1] See www.almaden.ibm.comcsquestdemos.html.

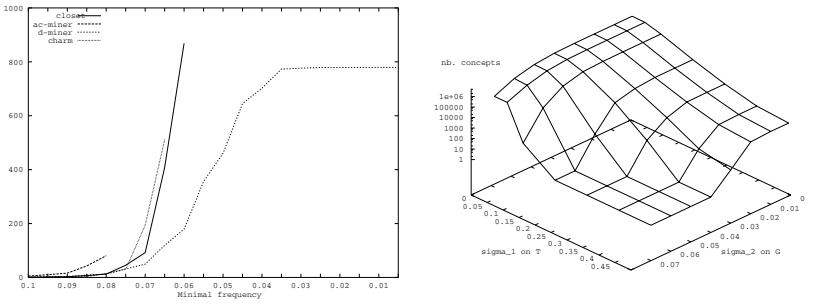

Fig. 5. A microarray dataset analysis

very low frequency threshold, otherwise almost no concept is provided. Consequently D-Miner is much better than the other algorithms 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.

Let us consider the computation of the bi-sets that satisfy two minimal frequency constraints on $\mathcal{L}_{\mathcal{O}}$ and $\mathcal{L}_{\mathcal{P}}$: one on item (gene) sets and the other one on objects (biological situations). In Figure 5 (right), we plot the number of concepts obtained using both $\mathcal{C}_t(\mathbf{r}, \sigma_1, T)$ and $\mathcal{C}_g(\mathbf{r}, \sigma_2, G)$ when σ_1 and σ_2 vary. It appears that using only one of the two constraints does not reduce significantly the number of extracted concepts (see values when $\sigma_1 = 0$ or $\sigma_2 = 0$). 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 $|G| \geq 10$ and $|T| \geq 21$ is 142 279. The number of concepts verifying $|G| \geq 10$ and $|T| \geq 0$ is 5 422 514. The number of concepts verifying $|G| \geq 0$ and $|T| \geq 21$ is 208 746. The gain when using simultaneously both constraints is significant.

4 Conclusion

Computing formal concepts has been proved useful in many application domains but remains extremely hard from dense boolean datasets 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 concept and the D-Miner algorithm can push them into the extraction process. Considering one of our applications that is described in [5], 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 [4].

Acknowledgments. We thank M. Zaki who has kindly provided his CHARM implementation. J. 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 *ACM SIGMOD Conference on Management of Data SIGMOD'93*, pages 207–216, 1993.
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. A. Berry, J.-P. Bordat, and A. Sigayret. Concepts can not afford to stammer. In *Proceedings JIM'03*, pages 25–35, Metz, France, September 2003.
5. J. Besson, C. Robardet, J.-F. Boulicaut, and S. Rome. Constraint-based bi-set mining for biologically relevant pattern discovery in microarray data. *Intelligent Data Analysis*. Accepted for publication in February 2004.
6. J.-F. Boulicaut, A. Bykowski, and C. Rigotti. Approximation of frequency queries by mean of free-sets. In *PKDD'00*, volume 1910 of *LNAI*, pages 75–85, 2000.
7. A. Bykowski. *Condensed representations of frequent sets: application to descriptive pattern discovery*. PhD thesis, Institut National des Sciences Appliquées de Lyon, LIRIS, F-69621 Villeurbanne cedex, France, Oct. 2002.
8. B. Ganter. Two basic algorithms in concept analysis. Technical report, Technisch Hochschule Darmstadt, Preprint 831, 1984.
9. L. Nourine and O. Raynaud. A fast algortihm for building lattices. *Information Processing Letters*, 71:190–204, 1999.
10. 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.
11. J. Pei, J. Han, and R. Mao. CLOSET an efficient algorithm for mining frequent closed itemsets. In *ACM SIGMOD Workshop DMKD'00*, 2000.
12. 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.
13. R. Wille. Restructuring lattice theory: an approach based on hierarchies of concepts. In I. Rival, editor, *Ordered sets*, pages 445–470. Reidel, 1982.
14. M. J. Zaki and C.-J. Hsiao. CHARM: An efficient algorithm for closed itemset mining. In *Proceedings SIAM DM'02*, Arlington, USA, April 2002.