Mining Frequent δ-Free Patterns in Large Databases

Céline Hébert and Bruno Crémilleux

GREYC, CNRS - UMR 6072, Université de Caen,
Campus Côte de Nacre
F-14032 Caen Cédex France
{Forename.Surname}@info.unicaen.fr

Abstract. Mining patterns under constraints in large data (also called fat data) is an important task to benefit from the multiple uses of the patterns embedded in these data sets. It is a difficult task due to the exponential growth of the search space according to the number of attributes. From such contexts, closed patterns can be extracted by using the properties of the Galois connections. But, from the best of our knowledge, there is no approach to extract interesting patterns like δ-free patterns which are on the core of a lot of relevant rules. In this paper, we propose a new method based on an efficient way to compute the extension of a pattern and a pruning criterion to mine frequent δ-free patterns in large databases. We give an algorithm (FTminer) for the practical use of this method. We show the efficiency of this approach by means of experiments on benchmarks and on gene expression data.

Keywords: Large databases, δ-free patterns, extensions, rules, condensed representations.

1 Introduction

Large data are data sets characterized by a large number of columns (i.e., attributes) and few rows (i.e., transactions). Data mining algorithms extracting patterns have difficulty in running on this kind of data because the search space grows exponentially according to the number of rows and it becomes huge. Known algorithms such as Apriori [1] or the recent algorithms that compute the so-called condensed representations can fail in mining frequent or constrained patterns in large data [17]. This is an important challenge because these geometrical dimensions are often encountered in a lot of domains (e.g., bioinformatics, quality control, text mining). For instance, in gene expression data, the matrices to explore gather the expression of tens of thousands of genes in few biological situations (we will see in Section 5 an example of such a matrix with 27,679 gene expressions and 90 biological situations). In quality control, the number of steps and parameters during the mass production is very numerous.

Extracting the complete collection of patterns under various kind of constraints in such data is a promising direction research . The completeness means that every pattern which satisfies the defined constraints has to be returned

A. Hoffmann, H. Motoda, and T. Scheffer (Eds.): DS 2005, LNAI 3735, pp. 124–136, 2005.

(e.g., every frequent pattern, every closed pattern). This is important to capture all the information embedded in the data. For instance, in biological data, frequent patterns are on the basis of synexpression groups (i.e., co-regulated sets of genes assumed to take part in a common function within the cell). Thanks to the properties of Galois connections and the transposition of data, a technique has been proposed in the particular case of closed patterns [17]. Unfortunately, we will see Section 2.2 that this approach of transposition is impracticable with the δ-free patterns.

In this paper, we focus on the search of free (or key) patterns [4, 14] and δ-free patterns [6]. The latter are a generalization of free patterns. Let us recall that free patterns are made of attributes without relations among them. They reveal the sound relationships between the data. With regard to the constraint of frequency, they are the minimal patterns of the classes of equivalence. As the property of freeness (and δ-freeness) is anti-monotonous, free and δ-free patterns can be efficiently extracted even in correlated data [6]. These patterns make an efficient condensed representation of all frequent patterns and their uses are highly interesting. They enable to build rules with a bounded number of exceptions [5], non redundant rules [18], their capacity to indicate the minimal part of attributes highlighting a phenomenon is precious in classes characterization and classification [3, 7]. δ-free patterns combine the exhaustiveness of the relations within the database and the simplicity which is required to build rules (and especially classification rules) without over-fitting. There is a need of classes characterization and classification techniques in large data, for instance, to predict a cancer diagnosis according to individual gene expression profiles or, in the area of the quality control, to detect an equipment which is badly tuned in a silicon plate production chain.

We propose in this paper a method to mine frequent and δ-free patterns from large data without transposing the data set. The key idea is to use the extension of a pattern to check these constraints, because the extension has few objects in large databases. We show a new property to compute the extension of a pattern and a new pruning criterion. Their simultaneous use is on the core of the FTMINER algorithm that we propose to extract the frequent and δ-free patterns from data. Then we show the efficiency of FTMINER by means of experiments on several benchmarks and a gene expression database.

The organization of this paper is as follows. In Section 2, we recall useful definitions and we discuss related work on δ-free patterns mining. Section 3 presents our approach and new properties on extensions and pruning. The algorithm FTMINER is given in Section 4. We end this paper by some experimental results in Section 5.

2 Context and Definitions

2.1 Notations and Definitions

Basic notations. Let us recall some definitions and notations useful for the rest of this paper. We define a database r as a relation $\mathcal{R}$ between the set $\mathcal{A}$ of *attributes*

(or *items*) and the set $\mathcal{O}$ of *objects* (or *transactions*): for $a \in \mathcal{A}, o \in \mathcal{O}$, $a\mathcal{R}o$ if and only if the object o contains the attribute a. r can also be viewed as a boolean matrix. In this case, we say that $a\mathcal{R}o$ if and only if $(a, o) = 1$ in the matrix. Notice that $o \in \mathcal{O}$ is also a set of attributes. An *attribute pattern* or *itemset* is a subset of $\mathcal{A}$. Similarly, an *object pattern* is a subset of $\mathcal{O}$. We say that an attribute pattern X is supported by an object o if o contains X. A *specialization* relation is defined on the attributes patterns (resp. objects patterns): a pattern X_1 is more specific than X_2 if X_2 is a subset of X_1. Thanks to this relation, the attribute patterns can be represented in a *lattice*. We give an example of *transactional database* in Table 1.

Table 1. An example of transactional database

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
o_1	1	0	1	0	1	0	1	0
o_2	0	1	1	0	1	0	1	0
o_3	1	0	1	0	1	0	0	1
o_4	1	0	0	1	0	1	0	1
o_5	0	1	1	0	0	1	0	1

γ-frequency and δ-freeness. An attribute pattern X is *γ-frequent* if it is supported by at least γ objects in r, γ being a given threshold. In Table 1, the frequency (noted $\mathcal{F}$) of the attribute pattern a_3a_5 is 3 (i.e., $\mathcal{F}(a_3a_5) = 3$) and a_3a_5 is said 3-frequent. X is a *δ-free* pattern if there is no association rule between two of its proper subsets with less than δ exceptions (i.e., there is no rule $X_1 \Rightarrow X_2$ with $\mathcal{F}(X_1 \cup X_2) + \delta \geq \mathcal{F}(X_1)$ and $X_1 \cup X_2 = X$ and $X_1 \cap X_2 = \emptyset$). To facilitate the understanding of the next sections, we will use the following equivalent definition of the δ-freeness [6]: X is a *δ-free* pattern if for each $X_1 \subset X$, $\mathcal{F}(X) + \delta < \mathcal{F}(X_1)$. In Table 1, a_5a_8 is 1-free since $\mathcal{F}(a_5a_8) = 1$ and one have $\mathcal{F}(a_5) = \mathcal{F}(a_8) = 3 > \mathcal{F}(a_5a_8) + \delta$. When $\delta = 0$, X is called a 0-free set or a *free* set.

Extension and Intension. We recall the definition of the extension of an attribute pattern. Let X be an attribute pattern, O an object pattern. The *extension* $g(X)$ is the maximal set of the objects containing X. The *intension* $f(O)$ is the maximal set of attributes appearing in every object of O. $h = f \circ g$ and $h' = g \circ f$ are the closure operators of the Galois connection. An attribute (resp. object) pattern X (resp. O) is *closed* if $h(X) = X$ (resp. $h'(O) = O$).

2.2 Related Work

The minimal frequency constraint is the most usual constraint in data mining. It is on the core of well-known algorithms like Apriori [2] which extracts all the γ-frequent patterns by scanning the database at each level. This levelwise algorithm is generalized to anti-monotonous constraints according to the specialization of

attributes [12]. If this technique is efficient in sparse data, it fails in correlated data [5]. By computing the frequency of only a few patterns (the minimal and the maximal patterns of the classes of equivalence), condensed representations based on free (or key) patterns [4, 14] and on closed patterns [5, 13, 15, 19] improve this approach on correlated data. These condensed representations are called *exact* because the exact frequency of each pattern can be inferred. If a bounded number of errors on the frequency of patterns is accepted, the condensed representation of δ-free patterns is more concise and can be mined more efficiently. Let us note that the δ-freeness is an anti-monotonous constraint and the higher δ, the more the efficiency of the pruning increases. It is important to be able to extract these patterns because they enable multiple uses on data mining techniques [3, 11, 18] (e.g., rules with minimal body, characterization of classes, classification).

Unfortunately, if the number of attributes is very large (i.e., large data), even the algorithms on condensed representations based on closed and δ-free patterns fail (except if the frequency threshold is very high, which is not sensible in real applications). In the specific case of the closed patterns, a technique relying on the properties of Galois connections and the transposition of data has been proposed [17]. Unfortunately, there is no straightforward generalization of this approach for a lot of constraints. By using this technique, the extracted patterns are object patterns and no longer attribute patterns and it is necessary to define the transposed form of the constraint. It is easy for closed patterns (thanks to the Galois connections), but not for a lot of constraints and especially for the δ-freeness [10]. In this case, each equivalence class contains at least one constrained pattern and one has to consider each attribute pattern of the lattice [10].

Notice that another method to extract free patterns is presented in [16]. It uses generalized properties on antimatroid spaces. An antimatroid space corresponds to the particular case of a lattice where each equivalence class of frequency contains one unique minimal generator. It is unlikely that happens in real data sets but this method has been extended to spaces that are not antimatroids in [9]. In this last case, the free patterns can be extracted from the closed ones by using minimal transversals of hypergraphs, but the complexity of the technique remains an open issue [8] and this approach cannot be used in practice.

3 Computing Frequency and δ-freeness in Large Data

This section presents our approach to mine frequent δ-free patterns in large data. We start by giving the main ideas, then we specify the technical key points: the link between extensions and the minimal frequency and δ-freeness constraints, our technique to compute the extensions and a new criterion based on the conjunction of the minimal frequency and δ-freeness constraints.

3.1 Main Ideas of Our Approach

The computation of the closures of patterns is often a bottleneck for algorithms mining frequent and δ-free patterns. Unfortunately, in the case of large

databases, the closures contain a lot of attributes and their storage requires a large amount of memory. That is why this approach often fails. But, with large data, there are only few objects which satisfy a set of attributes. Our idea is to check the δ-freeness constraint by using the corresponding patterns of objects: the extensions gather small object patterns easier to store.

Let us note that γ-frequency and δ-freeness are anti-monotonous constraints. We benefit from the pruning properties coming from such constraints. Moreover, we define and exploit a new pruning criterion stemmed from the conjunction of the γ-frequency and δ-freeness. This criterion is checked by using the extensions of patterns. Finally, we think that the success of this approach lies on the combination of these two points: mining γ-frequent and δ-free patterns by using the extensions and the use of this new pruning criterion.

3.2 Extension as a Frequency

Property 1 indicates the relation between the extension and the frequency of an attribute pattern.

Property 1. *The frequency of an attribute pattern X is equal to the cardinal of its extension $|g(X)|$.*

It is clear that Property 1 is well known but its use is interesting because it enables to rewrite the definitions of the minimal frequency and δ-freeness constraints with extension:

Definition 1. *An attribute pattern X is γ-frequent if $|g(X)| \geq \gamma$.*

Definition 2. *An attribute pattern X is δ-free if for all $X_1 \subset X$, $|g(X)| + \delta < |g(X_1)|$.*

In the example in Table 1, the extension of the attribute pattern a_1a_3 is equal to o_1o_3 and its frequency is 2 as indicated by Property 1. a_1a_3 is 2-frequent. To illustrate Definition 2, let us have a look at the patterns a_1a_3 and a_1a_4. a_1a_3 is 0-free because $|g(a_1)| = 3 > |g(a_1a_3)| + \delta = 2$ and $|g(a_3)| = 4 > |g(a_1a_3)| + \delta$. Nevertheless, a_1a_4 is not 0-free since $|g(a_4)| = 1 = |g(a_1a_4)| + \delta$.

An immediate and important consequence of Definitions 1 and 2 is that we are now able to establish the frequency and the δ-freeness of any pattern only with its extension. The next section explains how to compute efficiently the extensions.

3.3 A New Property to Compute Extension

The Property 2 allows to compute the extension of a pattern X from the extension of two of its subsets provided that their union is equal to X. So, from the extensions of the patterns at the level k, we are able to determine the extensions of the patterns at the level $k + 1$.

Property 2. *Let X_1 and X_2 be two patterns, the extension of $X_1 \cup X_2$ is equal to $g(X_1) \cap g(X_2)$.*

Proof. $\subseteq$ We have $X_1 \subseteq X_1 \cup X_2$ and $X_2 \subseteq X_1 \cup X_2$. As g is a decreasing function, we obtain that $g(X_1 \cup X_2) \subseteq g(X_1)$ and $g(X_1 \cup X_2) \subseteq g(X_2)$ so we have immediately $g(X_1 \cup X_2) \subseteq g(X_1) \cap g(X_2)$.
$\supseteq$ Let us consider o an object of $g(X_1) \cap g(X_2)$. By definition, o contains the patterns of attributes X_1 and X_2. As a consequence, we deduce that o contains $X_1 \cup X_2$. So o belongs to $g(X_1 \cup X_2)$.

For instance, in the example in Table 1: $g(a_1a_8) = o_3o_4$, $g(a_3a_5) = o_1o_2o_3$ and $g(a_1a_3a_5a_8) = o_3 = g(a_1a_8) \cap g(a_3a_5)$.

Several advantages stem from this property for mining patterns in large data. Firstly, as already said, the extensions are short patterns, easy to store and their intersections are computed in a short time. Secondly, to get the extension of a pattern X, we only have to compute the intersection of the extensions of two subsets of X (and not of all its subsets). Thirdly, the database is only scanned *once* (for patterns of length 1, i.e., items). On the contrary of the running of an usual levelwise algorithm, this avoids storing for each level of the search space all the candidate patterns.

We will see in Section 4 that it is sufficient to use Property 2 on patterns having the same length and a common prefix to mine frequent δ-free patterns in large data.

3.4 A New Pruning Criterion

This section presents a new pruning criterion for mining frequent δ-free patterns. First, let us note that, as both the frequency and the δ-freeness are anti-monotonous constraints, we naturally use the efficient pruning properties linked to such constraints [12]. Nevertheless, we go further and we highlight a new pruning criterion (Criterion 1) which comes from Theorem 1. This new pruning criterion is based on the common use of the minimal frequency and the δ-freeness properties.

Theorem 1. *Let X be a pattern. If X is a γ-frequent and δ-free pattern then for all $X_1 \subset X$, $|g(X_1)|$ is greater than $\gamma + \delta$.*

Proof. Theorem 1 is an immediate consequence of Definitions 1 and 2. X is a γ-frequent and δ-free pattern. Definitions 1 and 2 imply that for all $X_1 \subset X$, $\gamma + \delta \leq |g(X)| + \delta < |g(X_1)|$.

Pruning Criterion 1. *Let X be a pattern such that $|g(X)| \leq \gamma + \delta$, there is no superset of X being a γ-frequent δ-free pattern. So a levelwise algorithm can prune the search space from X.*

Criterion 1 is obtained by the contrapositive of Theorem 1. Let us examine the pattern a_5a_7 in the example in Table 1. a_5 and a_7 are 1-frequent and 1-free. They cannot be pruned by using classical pruning properties of anti-monotonous constraints and a_5a_7 is generated. Nevertheless, by using Criterion 1, a_5a_7 is not a candidate because $|g(a_7)| = 2 = \gamma + \delta$. The explanation of this pruning is

the following. To be 1-frequent, $|g(a_5a_7)|$ should be greater than or equal to 1. But, to be 1-free, $|g(a_5a_7)|$ should be smaller than $|g(a_7)| - 1 = 1$. So, the minimal frequency is in contradiction with the δ-freeness and a_5a_7 cannot satisfy simultanously these constraints.

4 FTMINER

This section presents FTMINER (FT for Free faT[1] databases Miner), an algorithm based on our approach given in Section 3. FTMINER extracts all the γ-frequent δ-free patterns from a database r. It follows the outline of levelwise algorithms. Let us recall that its originality is that there is no generation phase for all candidates which is very useful for large data. The database is only scanned once (for items) and, thanks to the use of extension, generation and verification are simultaneous. The process is also speeded up by the pruning Criterion 1.

FTMINER (database r, threshold γ, number of exceptions δ)

1. $\mathcal{F}ree_1 := \{a \in \mathcal{A} \mid |\mathcal{O}| - \delta > |g(a)| \geq \gamma\}$
2. $\mathcal{G}en_1 := \{a \in \mathcal{F}ree_1 \mid |g(a)| > \gamma + \delta)\}$
3. $k := 1$
4. **while** $\mathcal{G}en_k \neq \emptyset$ **do**
5. **for each** $(Y \cup \{A\}, Y \cup \{B\}) \in \mathcal{G}en_k \times \mathcal{G}en_k$ **do**
   // generation of one candidate of length $k+1$
6. $X := Y \cup \{A\} \cup \{B\}$
7. $g(X) := g(Y \cup \{A\}) \cap g(Y \cup \{B\})$
   // γ-frequency
8. **if** $|g(X)| \geq \gamma$ **then**
   // δ-freeness
9. $i := 1$
10. **while** $i \leq k+1$ **and** $X \backslash \{x_i\} \in \mathcal{G}en_k$ **and** $|g(X)| + \delta < |g(X \backslash \{x_i\})|$ **do**
    $i := i + 1$
11. **od**
12. **if** $i = k+2$ **then**
13. $\mathcal{F}ree_{k+1} := \mathcal{F}ree_{k+1} \cup \{X\}$
14. **if** $|g(X)| > \gamma + \delta$ **then**
15. $\mathcal{G}en_{k+1} := \mathcal{G}en_{k+1} \cup \{X\}$
16. **od**
17. $k := k + 1$
18. **od**
19. **return** $\bigcup_{i=1}^{k-1} \mathcal{F}ree_i$

[1] The word "fat" is also used to refer to large data sets as indicated for instance by D. Hand during his invited talk at PKDD'04.

Let $\mathcal{F}ree_k$ be the set of the free patterns at the level k whose frequency is greater than or equal to γ and $\mathcal{G}en_k$ be the set of generators at the level k i.e., the patterns in $\mathcal{F}ree_k$ with a frequency greater than $\gamma + \delta$.

The first step is the initialization of $\mathcal{F}ree_1$ and $\mathcal{G}en_1$. One scan on the database enables to compute the extension of the items and to determine whether they are γ-frequent and δ-free or not and $\mathcal{F}ree_1$ is obtained (Line 1). The initialization of $\mathcal{G}en_1$, using the pruning Criterion 1, stands in Line 2 and $\mathcal{G}en_1$ contains the γ-frequent δ-free patterns which have a frequency greater than $\gamma + \delta$.

The main loop begins in Line 4: it stops when there is no generators left at the considered level. For generating one candidate X at the level $k+1$ (Line 5), two patterns having a common prefix Y of length $k-1$ are joined (Line 6). The computation of the extension of X by intersecting the extensions of its generators is performed Line 7 using Property 2. In Line 8, Definition 1 is used to test whether the candidate X is γ-frequent thanks to its extension.

The loop begining at Line 10 considers every subset of X of length k. For each one (except for the two generators) the algorithm checks if it belongs to $\mathcal{G}en_k$ (i.e., if it is γ-frequent, δ-free and if its frequency is greater than $\gamma + \delta$) and if its frequency is greater than the frequency of the candidate plus δ. If X satisfies all the tests (Line 12), it is added in $\mathcal{F}ree_{k+1}$. Moreover, X is also a generator if its frequency is greater than $\gamma + \delta$ using Criterion 1.

Theorem 2 shows that FTMINER is correct and complete.

Theorem 2. *The algorithm* FTMINER *extracts all the γ-frequent and δ-free patterns from the database r.*

Proof (Correctness). Let us prove that a pattern X in $\mathcal{F}ree_k$ is a γ-frequent δ-free pattern. We test at Line 8 if $|g(X)| \geq \gamma$, what ensures that X is γ-frequent. At Line 10, we establish that X is δ-free using the condition $|g(X)| + \delta < |g(X \backslash \{x_i\})|$ (cf. Definition 2).

Proof (Completeness). The algorithm FTMINER covers the whole attribute search space thanks to the principle of the levelwise algorithms. The accuracy of the used pruning criteria (properties of anti-monotonous constraints and Criterion 1) entails the completeness of FTMINER.

5 Experiments

The aim of the experiments is to show the run-time benefit brought by FTMINER and emphasizes that FTMINER is able to mine frequent δ-free patterns in situations where prototypes (even taking benefit from condensed representations) fail. In Section 5.1 we compare on benchmarks FTMINER to MVMINER. The latter is a common prototype to extract condensed representations composed of δ-free patterns[2]. Let us note that it is equivalent to ACMINER implemented by

[2] MVMINER has been implemented by François Rioult (GREYC).

A. Bykowski (LIRIS) [5]. To the best of our knowledge, there exists no other prototype to mine frequent δ-free patterns. Section 5.2 widens the comparison to real gene expression data sets.

All the tests were performed on a 2.20 GHz Pentium IV processor with Linux operating system by using 3Go of RAM memory.

5.1 Results on Benchmarks

The benchmarks come from the UCI repository[3].

Benchmarks with a Lot of Attributes. In order to get large benchmarks, we transposed the `CMC` and `ABALONE` data sets. Thus, in the following, the used data sets have 30 rows and 1474 columns for `CMC`, 30 rows and 4178 columns for `ABALONE`. Figure 1 plots the comparison between FTMINER and MVMINER on run-times during the computation of frequent 0-free patterns according to the frequency threshold γ. γ ranges from 10 to 6 (37 to 20 percent) for `CMC`, 9 to 6 (30 to 20 percent) for `ABALONE`.

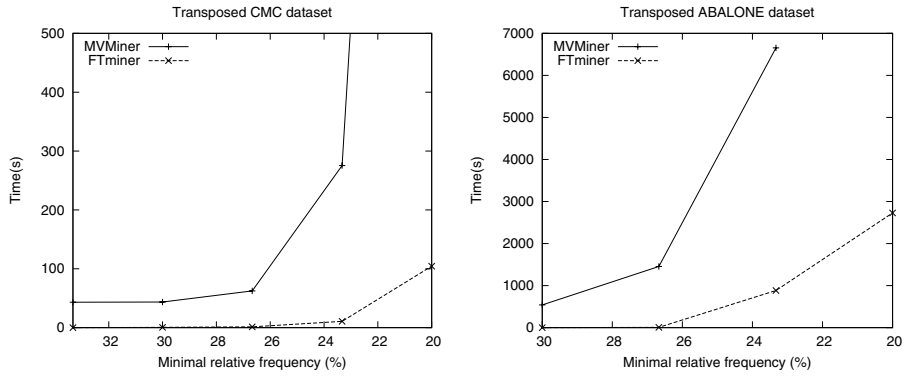

Fig. 1. Run-time performances according to the frequency on large data

As expected, the run-time increases when γ decreases. FTMINER clearly outperforms MVMINER (Figure 1). The latter fails when γ is equal to 5 (17%) for lack of memory while FTMINER ends in 2420 s on `CMC`. MVMINER also fails on `ABALONE` when γ is equal to 6 (20%).

Benchmarks with Usual Dimensions. Out of curiosity, we test FTMINER on data with usual dimensions i.e. having much more rows than attributes. We used the benchmarks `MUSHROOM` and `PUMSB` (from UCI repository). `MUSHROOM` is a 8124×120 data and `PUMSB` a 49046×7118 data. Figure 2 indicates that FTMINER runs faster than MVMINER even if there is an important number of

[3] `http://www.ics.uci.edu/~mlearn/MLSummary.html`

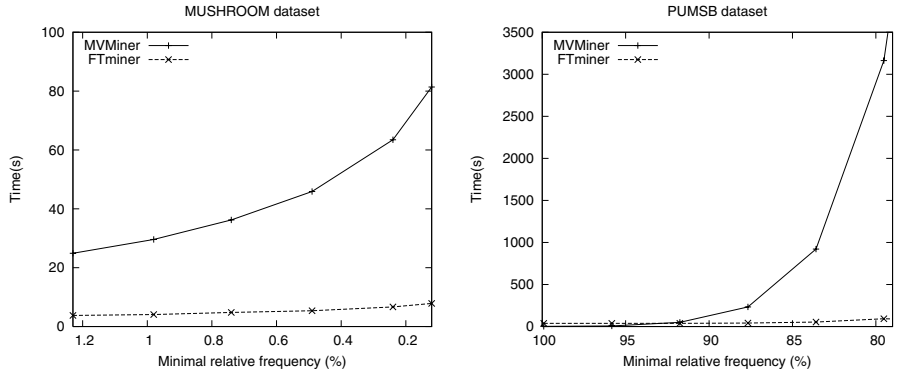

Fig. 2. Run-time performances according to the frequency on data having usual dimensions

objects in MUSHROOM and PUMSB. However, in one situation (PUMSB benchmark with a relative frequency threshold of 75.5%), FTMINER was lacking of memory due to the size of the extensions while MVMINER ends in 8829 seconds. This result was expected because the benefit of using the extension on large data (i.e., few patterns of objects) might not be reached on huge data with usual dimensions.

5.2 Results on Gene Expression Data Sets

We performed similar comparisons on a publicly available human Serial Analysis of Gene Expression (SAGE) data set[4] SAGE is an experimental technique designed to quantify gene expression. SAGE data provide expression values for given biological situations and given genes. These data sets are characterized by a large number of columns and few biological situations. For instance, the data set used for these experiments gathers 27,679 gene expressions for 90 biological situations.

Figure 3 (left) plots the run-times for mining the 3-free patterns with γ varying from 30 to 24 (33 to 27 percent). We used a logarithmically scaled ordinate axis. With a relative frequency threshold of 33.3%, FTMINER spends 30 seconds whereas one day is needed for MVMINER. With a threshold of 32%, FTMINER spends 50 seconds and MVMINER more than two days. Such results show the efficiency of FTMINER on large data.

Another aim was to experimentally quantify the efficiency of the new pruning criterion (Criterion 1). Figure 3 (right) plots the run-times of the extractions with and without this pruning criterion according to the number of exceptions. The run-time benefit is important: for $\gamma = 27$ (corresponding to 30%) and $\delta = 5$, it spends 31 seconds to extract the frequent δ-free patterns using Criterion 1 and 527 seconds without. In average, the run-time is divided by 7 thanks to

[4] This data set comes from the CGMC laboratory (CNRS UMR 5534) and has been prepared by Olivier Gandrillon and Sylvain Blachon.

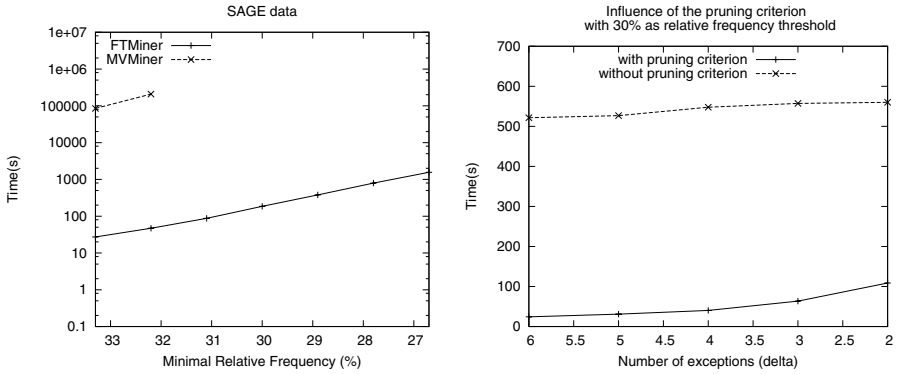

Fig. 3. Run-time performances on SAGE data

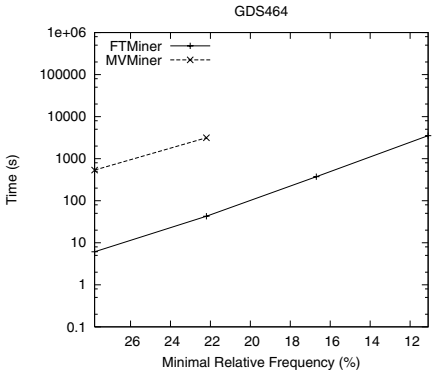

Fig. 4. Run-time performances on Dual-Channel data (GDS464)

Criterion 1. This can be explained by the large number of candidates additionally pruned thanks to this criterion: when $\gamma = 27$ and $\delta = 5$, this number is divided by 52, from 732,557,270 to 14,056,991.

Obviously this approach runs on other gene expression data sets. Out of curiosity, we ran our prototype on the gene expression data `GDS464` from the Gene Expression Omnibus repository[5] This data (collected in Dual-Channel experiments) gives the expression of 7085 genes in 90 biological situations. Figure 4 shows the run-times for mining the 2-free patterns according to γ (logarithmically scaled ordinate axis). The extraction becomes intractable with MVMINER when γ is less than 20%.

These experiments show that using both the extensions and the new pruning criterion enables to mine frequent δ-free patterns in large data whereas other approaches fail.

[5] Publicly available at URL `http://www.ncbi.nlm.nih.gov/projects/geo/gds/gds_browse.cgi?gds=464`

5.3 Discussion

These experiments prove the practical interest of the use of the extension in the case of large data sets. Nevertheless, Section 5.1 shows that the use of extension could also be efficient when mining data with usual dimensions (i.e., a large number of objects and few attributes). Furthermore, even in such data, the computational cost of the closures is more expensive than the one of the extension. It may be explained by the fact that the computing of a closure requires to intersect all the objects containing a given pattern in the data set. The computing of an extension is purely limited to the intersection of two objects as explained in Section 3.3.

6 Conclusion

Mining patterns in large data is a difficult task due to the large number of attributes. It is an important challenge because a lot of data sets have such geometrical dimensions and patterns like frequent δ-free are required by the owners of the data for several uses like classes characterization or classification. In this paper, we have proposed a new method based on a efficient way to compute the extension of a pattern and a pruning criterion to mine frequent δ-free patterns in large databases. A key point of the success of this approach is that the extensions in large data gather small object patterns easy to store. Experiments on benchmarks and a real gene expression data set show the practical use of this approach. Further work deals with the use of the extension to improve the extraction of patterns satisfying other constraints.

Acknowledgements. The authors thank the CGMC Laboratory (CNRS UMR 5534, Lyon) for providing the gene expression database. We thank François Rioult and Arnaud Soulet for fruitful discussions. This work has been partially funded by the ACI "masse de données" (MD 46, Bingo).

References

[1] R. Agrawal, H. Mannila, R. Srikant, and H. Toivonen. Fast discovery of associations rules. In *Advances in Knowledge Discovery and Data Mining*, 1996.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In J. B. Bocca, M. Jarke, and C. Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 12–15 1994.

[3] R. J. Bayardo. The hows, whys, and whens of constraints in itemset and rule discovery. In *proceedings of the workshop on Inductive Databases and Constraint Based Mining*, 2005.

[4] J.-F. Boulicaut and A. Bykowski. Frequent closures as a concise representation for binary data mining. In *proceedings of the Fourth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'00)*, volume 1805 of *Lecture notes in artificial intelligence*, pages 62–73, Kyoto, Japan, 2000. Springer-Verlag.

[5] J.-F. Boulicaut, A. Bykowski, and C. Rigotti. Approximation of frequency queries by means of free-sets. In *The Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases*, Lyon, 2000.

[6] 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.

[7] B. Crémilleux and J.-F. Boulicaut. Simplest rules characterizing classes generated by delta-free sets. In *22nd Int. Conf. on Knowledge Based Systems and Applied Artificial Intelligence (ES'02)*, pages 33–46, Cambridge, UK, December 2002.

[8] T. Eiter and G. Gottlob. Identifying the minimal transversals of a hypergraph and related problems. *SIAM Journal on Computing archive*, 24(6):1278–1304, 1995.

[9] R. E. Jamison and J. L. Pfaltz. Closure spaces that are not uniquely generated. In *Ordinal and Symbolic Data Analysis*, Brussels, 2000.

[10] B. Jeudy and F. Rioult. Database transposition for constrained closed pattern mining. In *Third International Workshop on Knowledge Discovery in Inductive Databases*, 2004.

[11] H. Mannila and H. Toivonen. Multiple uses of frequent sets and condensed representations. In *proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96)*, pages 189–194, Portland, Oregon, 1996.

[12] H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.

[13] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. *Lecture Notes in Computer Science*, 1999.

[14] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Efficient mining of association rules using closed itemset lattices. *Data Mining and Knowledge Discovery journal*, 24(1):25–46, 1999.

[15] J. Pei, J. Han, and R. Mao. CLOSET: An efficient algorithm for mining frequent closed itemsets. In *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 21–30, 2000.

[16] J. L. Pfaltz and C. Taylor. Closed set mining of biological data. In *Workshop on Data Mining and Bioinformatics*, 2002.

[17] F. Rioult, J.-F. Boulicaut, B. Crémilleux, and J. Besson. Using transposition for pattern discovery from microarray data. In M. J. Zaki and C. C. Aggarwal, editors, *proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'03)*, pages 73–79, San Diego, CA, 2003.

[18] M. J. Zaki. Generating non-redundant association rules. In *proceedings of the 6th International Conference on Knowledge Discovery and Data Mining (ACM SIGKDD'00)*, pages 34–43, Boston, MA, 2000.

[19] M. J. Zaki and C. J. Hsiao. CHARM: an efficient algorithm for closed itemset mining. In *proceedings of the 2th SIAM international conference on Data Mining*, pages 33–43, Arlington, 2002.