ABS: Adaptive Borders Search of frequent itemsets

Frédéric Flouvat[1], Fabien De Marchi[2], Jean-Marc Petit[1]

[1] Laboratoire LIMOS, UMR CNRS 6158
Université Blaise Pascal - Clermont-Ferrand II,
24 avenue des Landais, 63 177 Aubière cedex, France
flouvat@isima.fr, jmpetit@math.univ-bpclermont.fr

[2] Laboratoire LIRIS, FRE CNRS 2672
Université Claude Bernard Lyon 1
8, boulevard Niels Bohr, 69 622 Villeurbanne cedex France
fabien.demarchi@liris.cnrs.fr

Abstract

In this paper, we present an ongoing work to discover maximal frequent itemsets in a transactional database. We propose an algorithm called ABS for Adaptive Borders Search, which is in the spirit of algorithms based on the concept of dualization. From an abstract point of view, our contribution can be seen as an improvement of the basic APRIORI algorithm for mining maximal frequent itemsets. The key point is to decide dynamically at which iteration, if any, the dualization has to be made to avoid the enumeration of all subsets of large maximal itemsets. Once the first dualization has been done from the current negative border, APRIORI is no longer used and instead, another dualization is carried out from the positive border known so far. The process is repeated until no change occurs anymore in the positive border in construction.

Experiments have been done on FIMI datasets from which tradeoffs on adaptive behavior have been proposed to guess the best iteration for the first dualization. Far from being the best implementation wrt FIMI'03 contributions, performance evaluations of ABS exhibit better performance than IBE, the only public implementation based on the concept of dualization.

1 Introduction

In this paper, we present an ongoing work to discover maximal frequent itemsets in a transactional database. We propose to adapt an algorithm originally devised for mining inclusion dependencies in databases [8]. This algorithm is called ABS for Adaptive Borders Search, and is in the spirit of algorithms based on the concept of dualization [14, 21].

The basic idea of our proposition is to combine the strength of both levelwise algorithm [1, 18] and *Dualize and Advance* algorithm [14] in such a way that:

- "small" maximal frequent itemsets are efficiently generated with levelwise strategies.
- "large" maximal frequent itemsets may be found efficiently by dualization.

The dualization performed is quite similar to that proposed in the *Dualize and Advance* algorithm. Nevertheless, instead of starting from some subset of maximal frequent itemsets as *Dualize and Advance* algorithm does, we use infrequent itemsets to perform the dualization. As a consequence, we obtain the so-called *optimistic positive border* of maximal frequent itemsets. The set of such candidates corresponds exactly to k-uniform hypergraph clique proposed in [22]. As a consequence, our proposition contributes to clarify some related contributions [22, 17, 3, 12, 7]) since it gives an exact characterization of the optimistic positive border of maximal frequent itemsets from some subset of infrequent itemsets.

From an abstract point of view, our contribution can be seen as an improvement of the basic APRIORI algorithm for mining maximal frequent itemsets. The key point is to decide dynamically at which iteration, if any, the dualization has to be made to avoid the enumeration of all subsets of large maximal itemsets. Once the first dualization has been done from the current negative border available at that iteration, APRIORI is no longer used and instead, another dualization is carried out from the positive border known so far. The process is repeated until no change occurs anymore in the positive border in construction.

Experiments have been done on FIMI datasets [10]. The adaptive behavior of our algorithm has been tuned from results gathered from these experiments. For the tested dataset, we were able to guess dynamically the best iteration for the first dualization, a key parameter of our algorithm.

Far from being the best implementation wrt FIMI'03 contributions [11], performance evaluations of ABS exhibit better performance than IBE [21], the only public implementation based on the concept of dualization.

2 Preliminaries

Let R be a set of symbols called *items*; a *line* is a subset of R, and a *binary relation* r over R is a multiset of lines. We suppose the reader is familiar with the notions of itemsets, support, and with the main aspects of frequent itemsets mining problem in a binary relation, given a threshold $minsup$ (see e.g. [1] for details). We recall the notion of borders of a set of itemsets [18]. Given F a set of itemsets over R, the *positive border* of F denoted by $\mathcal{B}d^+(F)$ is defined by $\mathcal{B}d^+(F) = max_{\subseteq}\{X \in F\}$. The negative border of F is defined by $\mathcal{B}d^-(F) = min_{\subseteq}\{Y \subseteq R \mid \forall X \in F, Y \not\subseteq X\}$. If FI is the set of all itemsets frequent in r, then $\mathcal{B}d^+(FI)$ is called the set of *maximal frequent itemsets* in r.

We will use the concepts of hypergraph and minimal transversal of a hypergraph, whose definition is pointed out here (see for example [4] for more details). Given V a finite set of elements. A subset E of V defines a *hypergraph* $\mathcal{H} = (V, E)$, where elements of V are called *vertices* of $\mathcal{H}$ and elements of E *edges* of $\mathcal{H}$. A *transversal* T of $\mathcal{H} = (V, E)$ is a subset of V that intersect all the elements of E. T is *minimal* if no other transversal of $\mathcal{H}$ are included in T. The set of all minimal transversals of $\mathcal{H}$ is noted $Tr(\mathcal{H})$.

The relationship between the notion of borders and minimal transversals of hypergraph has been exhibited in [18]. Indeed, any set of itemsets can be seen as a hypergraph; if FI is the set of frequent itemsets in a binary relation r, we have: $Tr(\overline{FI}) = \mathcal{B}d^-(FI)$, where $\overline{FI} = \{R - X \mid X \in FI\}$.

3 Method description

3.1 Starting with a levelwise algorithm

The algorithm $Apriori$ [1] was initially devoted to frequent itemset mining; Nevertheless, it has been proved to be still competitive for maximal frequent itemsets mining in many cases [11], when the size of elements to discover remain small.

Our goal is to exploit the efficiency of $Apriori$, but to automatically detect when it will fall into troubles and stop its execution. Then we propose to exploit the knowledge mined so far to initialize a different search, based on the concept of dualization between positive and negative borders; each border is updated and used to compute the corresponding dual border, until a fix point is reached.

3.2 From negative to positive border

In the sequel, let r be a binary database over a set of items R, $minsup$ a minimum support, and FI the set of frequent itemsets in r. After the levelwise part, our method is still iterative; at each iteration i, new elements of the positive and negative borders are expected to be discovered. We denote by $\mathcal{B}d_i^+$ (resp. $\mathcal{B}d_i^-$) the subset of $\mathcal{B}d^+(FI)$ (resp. $\mathcal{B}d^-(FI)$) discovered until the i^{th} iteration. In other words, $\forall i < j, \mathcal{B}d_i^+ \subseteq \mathcal{B}d_j^+$ and $\mathcal{B}d_i^- \subseteq \mathcal{B}d_j^-$. Roughly speaking, candidates for $\mathcal{B}d_i^+$ are obtained from elements of $\mathcal{B}d_i^-$, and candidates for $\mathcal{B}d_{i+1}^-$ are obtained from elements of $\mathcal{B}d_i^+$.

The following definitions and results have been proposed in [8] for inclusion dependency discovery problem in relational databases. We recall them in the context of maximal frequent itemsets mining, only the proofs are omitted.

We first define the notion of *Optimistic positive border*.

Definition 1*(Optimistic positive border) Given a set F of itemsets, the optimistic positive border of F is: $\mathcal{F}opt(F) = max_{\subseteq}\{X \subseteq R \mid \forall Y \in F, Y \not\subseteq X\}$.*

The next theorem gives a constructive characterization of $\mathcal{F}opt(F)$.

Theorem 1*[8] $\mathcal{F}opt(F) = \overline{Tr(F)}$*

Therefore, the idea is to compute the optimistic positive border for $\mathcal{B}d_i^-$ to obtain exactly the largest itemsets which do not contain any infrequent itemset discovered so far.

Proposition 1 *Let $X \in \mathcal{F}opt(\mathcal{B}d_i^-)$. If $sup(X) \geq minsup, X \in \mathcal{B}d^+(FI)$.*

Proof *Since X is maximal in the definition of $\mathcal{F}opt(\mathcal{B}d_i^-)$, each of its superset contains at least one element of $\mathcal{B}d_i^-$, and is infrequent by anti-monotonicity.* □

Then, $\mathcal{B}d_i^+$ is exactly made up of all the frequent itemsets in $\mathcal{F}opt(\mathcal{B}d_i^-)$.

3.3 From positive to negative border

In a dual way, the set $\mathcal{B}d_i^+$ is then used to compute its negative border $\mathcal{B}d^-(\mathcal{B}d_i^+)$, to finally update the negative border in construction and obtain $\mathcal{B}d_{i+1}^-$.

The next theorem gives a constructive characterization of $\mathcal{B}d^-(F)$, for any set F of frequent itemsets.

Theorem 2*[18] $\mathcal{B}d^-(F) = Tr(\overline{F})$*

Proposition 2 *Let $X \in \mathcal{B}d^-(\mathcal{B}d_i^+)$. If $sup(X) < minsup, X \in \mathcal{B}d^-(FI)$.*

Proof *Let X be an element of $\mathcal{B}d^-(\mathcal{B}d_i^+)$. By the definition of the negative cover of a set, each subset of X is included in an element of $\mathcal{B}d^+(FI)$ and then is frequent.* □

Then, $\mathcal{B}d_{i+1}^-$ is exactly made up of all the infrequent itemsets in $\mathcal{B}d^-(\mathcal{B}d_i^+)$.

3.4 The Algorithm ABS

Algorithm 1 computes the positive and negative borders of frequent itemsets in a given binary database. Within the framework of levelwise algorithms, ABS decides at each level whether or not the levelwise approach has to be stopped. In that case, the levelwise approach is halted, and the two borders are incrementally updated as described previously. The functions $GenPosBorder$ and $GenNegBorder$ compute respectively the optimistic positive and negative borders, using characterizations in theorems 1 and 2. The algorithm terminates when all elements of the optimistic positive border currently computed are frequent. It is worth noting that no dualization may occur at all: in this case, ABS is reduced to $APRIORI$. The proposition 3 ensures the correctness of ABS.

The behavior of the function $IsDualizationRelevant$ is described in section 3.5.

Proposition 3 *The algorithm ABS returns $\mathcal{B}d^+(FI)$ and $\mathcal{B}d^-(FI)$.*

Proof *If the test performed by $IsDualizationRelevant()$ is never true, the demonstration is obvious.*
If not, in line 15, from propositions 1 and 2, we have $\mathcal{B}d_i^+ \subseteq \mathcal{B}d^+(FI)$ and $\mathcal{B}d_{i-1}^- \subseteq \mathcal{B}d^-(FI)$
Moreover, the termination condition ensures that $\mathcal{B}d_i^+ = GenPosBorder(\mathcal{B}d_{i-1}^-)$; all elements in $\mathcal{B}d_i^+$ are frequent and all elements in $\mathcal{B}d_{i-1}^-$ are infrequent. Suppose that $\exists X \in \mathcal{B}d^-(FI) \mid X \notin \mathcal{B}d_{i-1}^-$. Then:

- *if $\exists Y \in \mathcal{B}d_{i-1}^- \mid Y \subset X$, since Y is infrequent, $X \notin \mathcal{B}d^-(FI)$ and there is a contradiction*
- *if $\nexists Y \in \mathcal{B}d_{i-1}^- \mid Y \subseteq X$, then from the definition of the optimistic positive border $\exists Z \in \mathcal{B}d_i^+ \mid X \subseteq Z$, which contradict the fact that X is infrequent.*

Thus $\mathcal{B}d_{i-1}^- = \mathcal{B}d^-(FI)$. An identical reasoning leads to $\mathcal{B}d_i^+ = \mathcal{B}d^+(FI)$.

□

Algorithm 1 ABS: Adaptive Border Search

```
Require: a binary database $r$, a integer $minsup$
Ensure: $\mathcal{B}d^+(FI)$ and $\mathcal{B}d^-(FI)$
1: $F_1 = \{A \in R \mid \text{sup}(A) \geq minsup\}$
2: $C_2 = AprioriGen(F_1)$
3: $i = 2$; $\mathcal{B}d_1^- = R - F_1$; $\mathcal{B}d_0^+ = \emptyset$
4: while $C_i \neq \emptyset$ do
5:    $F_i = \{X \in C_i \mid sup(X) \geq minsup\}$
6:    $\mathcal{B}d_i^- = \mathcal{B}d_{i-1}^- \cup (C_i - F_i)$
7:    $\mathcal{B}d_{i-1}^+ = \mathcal{B}d_{i-2}^+ \cup \{X \in F_{i-1} \mid \forall Y \in F_i, X \nsubseteq Y\}$
8:    if $IsDualizationRelevant(i, |\mathcal{B}d_i^-|, |F_i|, |C_i|) = TRUE$ then
9:       $\mathcal{B}d_i^+ = \{X \in GenPosBorder(\mathcal{B}d_i^-) \mid |X| \geq minsup\}$
10:      while $\mathcal{B}d_i^+ \neq \mathcal{B}d_{i-1}^+$ do
11:         $\mathcal{B}d_i^- = \{X \in GenNegBorder(\mathcal{B}d_i^+) \mid |X| \leq minsup\}$
12:         $\mathcal{B}d_{i+1}^+ = \{X \in GenPosBorder(\mathcal{B}d_i^-) \mid |X| \geq minsup\}$
13:         $i = i + 1$
14:      end while
15:      Return $\mathcal{B}d_i^+$ and $\mathcal{B}d_{i-1}^-$ and exit
16:   end if
17:   $C_{i+1} = AprioriGen(F_i)$
18:   $i = i + 1$
19: end while
20: $\mathcal{B}d_{i-1}^+ = \mathcal{B}d_{i-2}^+ \cup F_{i-1}$
21: Return $\mathcal{B}d_{i-1}^+$ and $\mathcal{B}d_{i-1}^-$
```

3.5 Adaptive aspects of ABS

The main adaptive aspect of ABS is conveyed by the function $IsDualizationRelevant$, line 8 of algorithm 1. As mentioned, its goal is to estimate if it is interesting to dualize the current negative border to the optimistic positive border.

We have identified four parameters specific to a given iteration of the levelwise algorithm, which can be obtained dynamically without any overhead:

- The current level i. No jump is allowed until a given integer threshold; we set the threshold equal to 4, since $Apriori$ is very efficient in practice to explore the levels 1 to 4. In our experiments, dualizing before this level incurs no improvement.
- $|\mathcal{B}d_i^-|$, the size of the current negative border. A simple remark can be made here: if this parameter is very large (more than 100000) the minimal transversals computation become prohibitive. We are not aware of existing implementations of minimal transversals

computation able to handle such input hypergraphs [1]. Moreover, such cases are likely to correspond to best scenario for $Apriori$.

- $|F_i|$, the number of frequent i-itemsets and $|\mathcal{B}d^-_i|$ have to be compared. Indeed, a small value of $|\mathcal{B}d^-_i|$ wrt $|F_i|$ is likely to give a successful dualization.
- $|F_i|$ and $|C_i|$, the number of candidates in level i, can also be compared. If $|F_i|/|C_i|$ is close to 1, we can suspect to be in a "dense" part of the search space, and thus the levelwise search should be stopped.

3.6 Practical aspects

3.6.1 Candidate generation from the current positive border

From [18], candidate generation of a levelwise algorithm for a problem representable as sets can be formulated using dualization: At the ith iteration, we have

$$C_{i+1} = Tr(\cup_{j\leq i}\overline{F_j}) - \cup_{j\leq i}C_j$$

It is shown in [18] that candidate itemsets of C_{i+1} are exactly of size $i+1$, which allows to improve candidate generation.

In the setting of this paper, we can see C_{i+1} as the set $\mathcal{B}d^-_{i+1} - \mathcal{B}d^-_i$, and thus we get:

$$C_{i+1} = Tr(\overline{\mathcal{B}d^+_i}) - \cup_{j\leq i}C_j$$

Here, the major difference with a pure levelwise approach is that $\mathcal{B}d^+_i$ may contain some elements of size greater than $i+1$.

One may question about the size of the largest elements of C_{i+1}: does there exist elements of size strictly greater than $i+1$? The answer is yes as shown in the following non trivial example.

Example 1
*Let r be the binary relation over a schema **R** $= \{A,B,C,D,E,F,G,H,I\}$ represented in Table 1. For a minsup equals to 1, the borders of frequent itemsets in r are $\mathcal{B}d^- = \{AE,BF,CG,DH,ABCDI\}$ and $\mathcal{B}d^+ = \{ABCHI, ABDGI, ABGHI, ACDFI, ACFHI, ADFGI, AFGHI, BCDEI, BCEHI, BDEGI, BEGHI, CDEFI, CEFHI, DEFGI, EFGHI, ABCD\}$.*
After a levelwise pass until level two, the four NFI of size two have been discovered, i.e. $\mathcal{B}d^-_2 = \{AE,BF,CG,DH\}$. Suppose the algorithm decides here to stop the pure levelwise search. Then, these sets are used to compute the optimistic positive border from level 2. It is made up of 16 itemsets of size 5, among which the only non frequent itemset is $ABCDI$. Thus, at this time, $\mathcal{B}d^+_2 = \{ABCHI, ABDGI, ABGHI, ACDFI, ACFHI, ADFGI, AFGHI, BCDEI, BCEHI, BDEGI, BEGHI, CDEFI, CEFHI, DEFGI, EFGHI\}$. We obtain $\mathcal{B}d^-(\mathcal{B}d^+_2) = \{ABCD\}$ of size 4, being understood that no elements of size 3 does exist.

A	B	C	D	E	F	G	H	I
1	1	1					1	1
1	1		1			1		1
1	1					1	1	1
1		1	1		1			1
1		1			1		1	1
1			1		1	1		1
1					1	1	1	1
	1	1	1	1				1
	1	1		1			1	1
	1		1	1		1		1
	1			1		1	1	1
		1	1	1	1			1
		1		1	1		1	1
			1	1	1	1		1
				1	1	1	1	1
1	1	1	1					

Table 1. Example database

In our first implementation, computing the set $\mathcal{B}d^-(\mathcal{B}d^+_i)$ using minimal transversals had a quite prohibitive cost on large hypergraph instances. Therefore, we made the choice to restrict $\mathcal{B}d^-(\mathcal{B}d^+_i)$ to its $(i+1)$-itemsets for *efficiency reasons*. This choice has no effect on the correctness of the algorithm, since the termination condition is always the same[2].

3.6.2 Dealing with "almost frequent" large candidate itemsets

Let us consider the case of a candidate itemset obtained after a dualization from the current negative border. Let X be this candidate. Two main cases do exist: either X is frequent, or X is infrequent. In that case, we propose to estimate a degree of error in order to "qualify the jump".

Given a new user-defined threshold δ, and a minimal support $minsup$, an *error measure*, noted

[1]Experiments conducted in [16, 2] only consider hypergraphs with not more than 32000 edges.

[2]We suspect the algorithm $PincerSearch$ [17] to be *not complete*. Indeed, the search strategy of $PincerSearch$ is very close to our proposition: if we only consider $(i+1)$-itemsets in $\mathcal{B}d^-(\mathcal{B}d^+_i)$, they correspond exactly to the candidate set C_{i+1} of $PincerSearch$. Since $PincerSearch$ stops as soon as $C_{i+1} = \emptyset$, some elements could be forgotten. From the example 1, after the level 2, C_3 is empty, and therefore the maximal set $ABCD$ seems to be never generated by $PincerSearch$.

$error(X, minsup)$, can be defined as the ratio between the minsup $minsup$ and the support of the infrequent itemset X, i.e. $error(X, minsup) = 1 - \frac{support(X)}{minsup}$.

Two sub-cases are worth considering:

- either $error(X, minsup) \leq \delta$: the "jump" was not successful but solutions should exist among the nearest subsets of X.
- or $error(X, minsup) > \delta$: In that case, the jump was over-optimistic and probably, no solution does exist among the nearest generalizations of X.

Note that this error measure is decreasing, i.e. $X \subset Y \Rightarrow error(X, minsup) \leq error(Y, minsup)$

In our current implementation, these almost frequent large itemsets are first considered as frequent to enable more pruning in subsequent passes. Afterward, they are considered at the very end of our algorithm. A pure top-down levelwise approach has been implemented to find out their subsets which can be maximal frequent itemsets.

4 Implementation details and experimental results

4.1 Implementation details

An implementation of the algorithm has been performed in C++/STL. Two existing implementations available from the FIMI repository website [10] were borrowed: the $Apriori$ code of C. Borgelt [5] and the prefix-tree implementation of B. Goethals using C++/STL [10].

To keep coherence with this implementation, we use a similar data structure for the new parts of the algorithm. The itemsets and the transactions are stored in a prefix-tree [1, 6].

Minimal Transversals computation For the minimal transversals computation, we implemented the algorithm proposed in [9] using a prefix-tree in order to handle relatively large hypergraph instances. Its incremental aspect is very interesting in our case, since the negative border is itself incremental. Note that improvements have been performed by exploiting the knowledge of previous dualizations. We do not give more details here.

4.2 Experimental results

We conducted experiments on a pentium 4.3GHz Processor, with 1Go of memory. The operating system was Redhat Linux 7.3 and we used gcc 2.96 for the compilation. We used four datasets available on the FIMI'03 repository.

We first evaluate the influence of the level from which the levelwise approach is stopped on the performances of ABS. Then, the impact of "almost frequent" large itemsets is studied for different threshold values for the error measure. Finally, we compare ABS with four maximal frequent itemsets mining algorithms: $Apriori$ and $Eclat$ [12] implemented by C.Borgelt [5], $Fpmax$ [13] based on $FP-trees$ [15] and IBE [21].

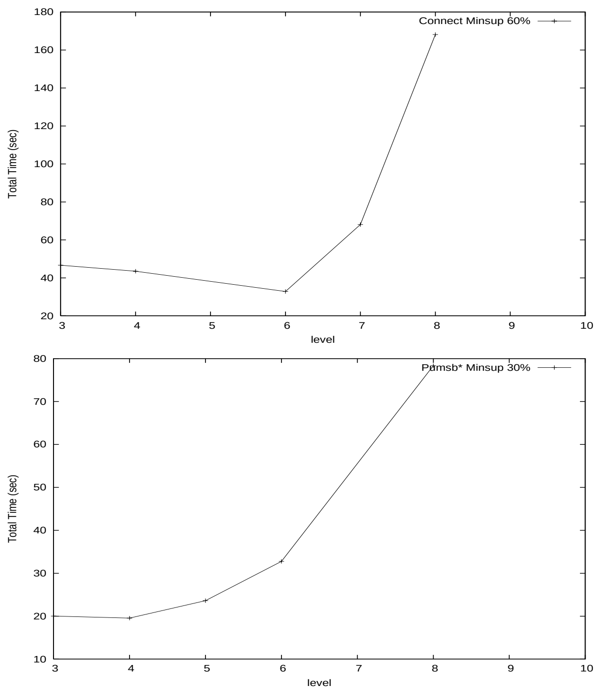

Figure 1. Forcing the first dualization at level k for connect (top) and pumsb* (bottom)

In figure 1, we forced the first dualization for different levels (from 3 to 8), on the connect dataset with a $minsup$ of 80 % and the pumsb* dataset with a $minsup$ of 30%. The results confirm the necessity to fix dynamically this parameter, and then justify an adaptive approach. Second, for all tested datasets, our function $IsDualizationRelevant$ has dynamically determined the best level to begin dualization.

The optimization based on the error measure is evaluated on figure 2. From pumsb dataset (on the top), this optimization appears to be interesting with a threshold value near 0.002. Nevertheless, on the connect dataset (bottom) no improvements is achieved. This comes from the fact that the proposed error measure is not strictly decreasing; and the equivalence classes induced by closed frequent itemsets are large. Our top down levelwise approach is prone to fail on this kind of databases .

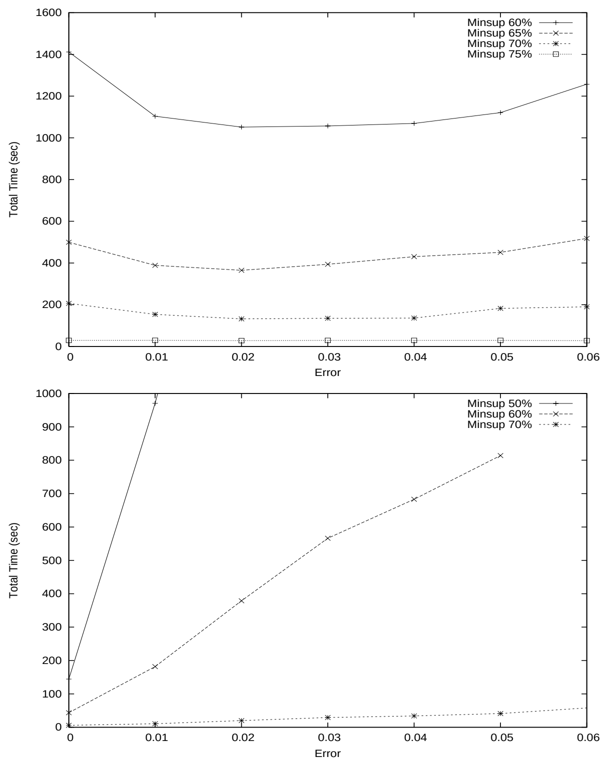

Figure 2. Exec. times for pumsb (top) and connect (down) wrt different error measure thresholds

From figures 3, 5 and 6, ABS is far to compete with best known implementations but tends to outperform IBE for most of our experimentations. Recall that IBE is the unique implementation based on the concept of dualization available from FIMI'03. We believe that this is due to the number of dualization performed by IBE, which is in the size of the positive border.

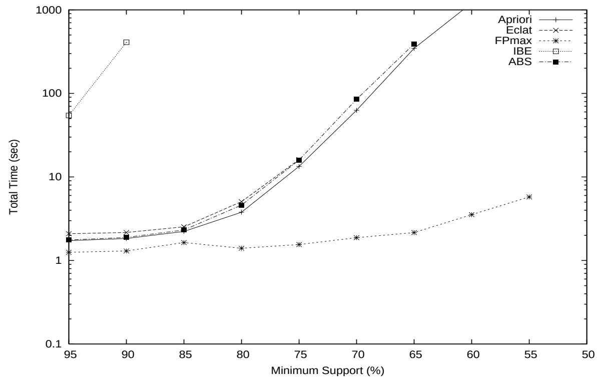

Figure 3. Execution times for database Pumsb

From figure 4, IBE exhibits better performances than ABS for low support thresholds (less than 20%). This is due to the fact that while the size of the positive border remains small (less than 5000 elements) the size of the negative border exceeds 10^6 elements, where some elements appear to have a very large size. This seems to be the worst case for ABS.

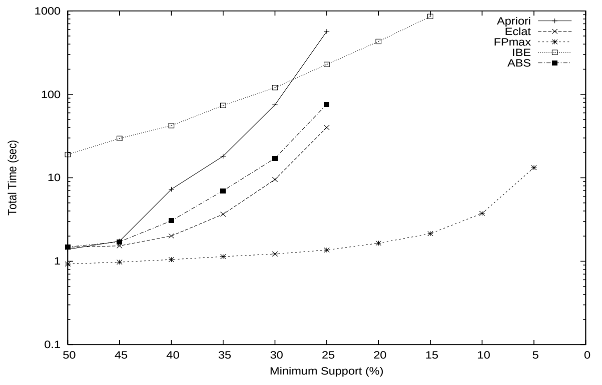

Figure 4. Execution times for database Pumsb*

From figure 5, ABS behaves like $Apriori$ as expected. Indeed, the positive border of retail is made up of "small" itemsets, and $Apriori$ turns out to be the best implementation for this kind of datasets.

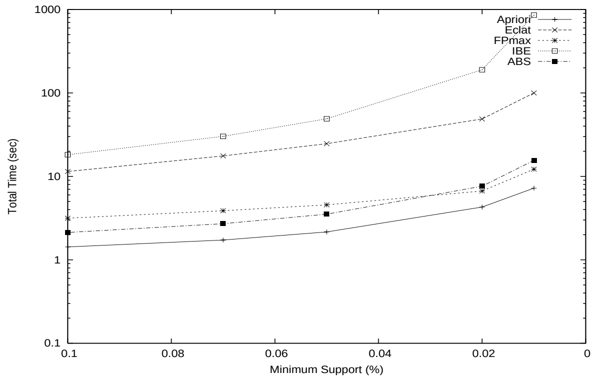

Figure 5. Execution times for database Retail

From figure 6, ABS is not as efficient as best known implementations (e.g. $fpmax$), but improves $Apriori$ by a factor of ten and beats $Eclat$ and IBE.

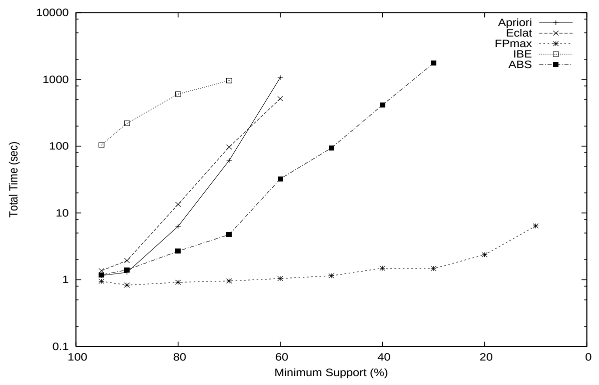

Figure 6. Execution times for database Connect

To sum up, two main reasons explain our mitigate results: 1) the cost of dualization remains high on very large hypergraph instances and 2) candidate generation and support counting seems to be not enough efficient in our current implementation.

The main parameter influencing performance of ABS turns out to be around the negative border. If for a given minsup, the negative border does not become too huge and its largest element remains "small" with respect to the largest maximal frequent itemset, ABS should have good performance.

5 Related works

Several algorithms exist for discovering maximal frequent itemsets mining in a transactional database (see FIMI'03). The goal is always to avoid an exponential search by characterizing as fast as possible largest frequent itemsets without exploring their subsets. $MaxMiner$ [3] uses a levelwise approach to explore the candidate itemsets, using the Rymon's enumeration system [20] - in which itemsets are arranged in a non redundant tree. But when a candidate X is counted over the database, the greatest candidate in the subtree of X is also counted; if it is frequent, then all the subtree can be pruned by anti-monotony of the "is frequent" property. Jumps done by $MaxMiner$ depend on the ordering of items used to build the tree and are therefore quiet different from jumps proposed in this paper. The algorithms $Mafia$ [7] and $GenMax$ [12] use the same principle as $MaxMiner$ with efficient optimizations, e.g. vertical bitmaps.

The $Pincer - Search$ Algorithm [17] uses a search strategy very close to ours. After a levelwise initialization, the principle is also to look at the largest not yet eliminated candidates. However, these large candidates are not characterized in a formal way.

In [14], the authors propose the *Dualize and Advance* algorithm. In their approach, the positive border in construction is always a subset of the positive border to be discovered. At each step, from some elements of the positive border already discovered, they generate the corresponding negative border. If one element of the negative border appears to be satisfied, they generate a specialization of it which belongs to the positive border and they re-iterate the process until each element of the negative border is indeed not satisfied. An implementation of a variant of *Dualize and Advance* has been proposed in [21] with an irredundant dualization. Their code is available from the FIMI'03 website.

Some algorithms like $Mafia$ [7] or DCI [19] can adapt themselves to mine frequent itemsets, with respect to the dataset density and some architectural characteristics (e.g. available memory). Even if these aspects improve performances, it only concerns choices for data structures; the mentioned algorithms do not really adapt their *strategy* to explore the search space.

6 Conclusion and future works

In this paper, we have proposed an ongoing effort toward the discovery of maximal frequent itemsets. Our contribution takes its roots from the algorithm ZigZag devised for inclusion dependency discovery in databases. Even if this two data mining problems fit into the same theoretical framework [18], they widely differ in practice which is not a surprise. Indeed, while ZigZag performed very well in our experiments[3], ABS does not exhibit such a good behavior for maximal frequent itemsets mining. Many reasons explain this result, for instance the availability of public datasets allowing thorough experimentations, the intrinsic properties of each problem, and may be the more important reason lies in the cost of a database access, in-memory resident data vs data stored into a DBMS.

Many improvements can be brought to our current implementation. Some are specific to our algorithm like for instance minimal transversal computation on large hypergraph instances or taking into account large equivalence classes induced by closed frequent itemsets during candidate generation from "almost frequent" itemsets. Some others belong to "the state of the art" of maximal frequent itemsets implementation : managing huge set of set, support counting... Complexity issues need also to be addressed.

To end up, we want to quote a personal note on the main objective of the FIMI workshop. We believe that frequent, closed and maximal itemsets mining are key data mining tasks since algorithms devised to solve these tasks are likely to be used in other contexts under some conditions [18]. Roughly speaking, for every problem *representable as sets*

[3]Note that dynamic parameters were quiet different, e.g. the first dualization was always performed at the second level.

with an anti-monotone predicate as for instance with functional dependency inference or simply anti-monotone predicates on itemsets other than "is frequent", the algorithms devised for FIMI should be useful to answer these tasks. Nevertheless, it seems rather optimistic to envision the application of many FIMI'03 [11] implementations to another data mining problem representable as sets. Indeed, even if the development of efficient data structures for managing huge sets of set is definitely useful, loading the database in main memory using sophisticated data structure specially devised for the anti-monotone predicate to be mined turns out to give very efficient algorithms but deserve other data mining tasks.

References

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In J. B. Bocca, M. Jarke, and C. Zaniolo, editors, *International Conference on Very Large Data Bases (VLDB'94), Santiago de Chile, Chile*, pages 487–499. Morgan Kaufmann, 1994.

[2] J. Bailey, T. Manoukian, and K. Ramamohanarao. A fast algorithm for computing hypergraph transversals and its application in mining emerging patterns. In *International Conference on Data Mining (ICDM'03), Floride, USA*, pages 485–488, 2003.

[3] R. J. Bayardo. Efficiently mining long patterns from databases. In L. M. Haas and A. Tiwary, editors, *ACM SIGMOD Conference, Seattle, USA*, pages 85–93, 1998.

[4] C. Berge. *Graphs and Hypergraphs*. North Holland, Amsterdam, 1973.

[5] C. Borgelt. Efficient implementations of apriori and eclat. In *FIMI'03 Workshop on Frequent Itemset Mining Implementations*, November 2003.

[6] C. Borgelt and R. Kruse. Induction of association rules : Apriori implementation. 2002.

[7] D. Burdick, M. Calimlim, and J. Gehrke. Mafia: A maximal frequent itemset algorithm for transactional databases. In *International Conference on Data Engineering (ICDE'01), Heidelberg, Germany*, pages 443–452. IEEE CS, 2001.

[8] F. De Marchi and J.-M. Petit. Zigzag : a new algorithm for discovering large inclusion dependencies in relational databases. In *International Conference on Data Mining (ICDM'03), Melbourne, Florida, USA*, pages 27–34. IEEE Computer Society, 2003.

[9] J. Demetrovics and V. Thi. Some remarks on generating armstrong and inferring functional dependencies relation. *Acta Cybernetica*, 12(2):167–180, 1995.

[10] B. Goethals. Frequent itemset mining implementations repository, http://fimi.cs.helsinki.fi/, 2003.

[11] B. Goethals and M. Zaki, editors. *Workshop on Frequent Itemset Mining Implementations*. CEUR Workshop Proceedings, 2003.

[12] K. Gouda and M. J. Zaki. Efficiently mining of maximal frequent itemsets. In N. Cercone, T. Y. Lin, and X. Wu, editors, *International Conference on Data Mining (ICDM'01), San Jose, USA*. IEEE Computer Society, 2001.

[13] G. Grahne and J. Zhu. Efficiently using prefix-trees in mining frequent itemsets. In *FIMI'03 Workshop on Frequent Itemset Mining Implementations*, November 2003.

[14] D. Gunopulos, R. Khardon, H. Mannila, S. Saluja, H. Toivonen, and R. S. Sharma. Discovering all most specific sentences. *ACM Transaction on Database System*, 28(2):140–174, 2003.

[15] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *ACM SIGMOD'00, Dallas, Texas, USA*, 2000.

[16] D. J. Kavvadias and E. C. Stavropoulos. Evaluation of an algorithm for the transversal hypergraph problem. In J. S. Vitter and C. D. Zaroliagis, editors, *Algorithm Engineering, International Workshop, WAE '99, London, UK*, volume 1668, 1999.

[17] D.-I. Lin and Z. M. Kedem. Pincer search: A new algorithm for discovering the maximum frequent set. In H.-J. Schek, F. Saltor, I. Ramos, and G. Alonso, editors, *Extending Database Technology (EDBT'98), Valencia, Spain*, volume 1377 of *Lecture Notes in Computer Science*, pages 105–119. Springer, 1998.

[18] H. Mannila and H. Toivonen. Levelwise Search and Borders of Theories in Knowledge Discovery. *Data Mining and Knowledge Discovery*, 1(1):241–258, 1997.

[19] S. Orlando, P. Palmerini, R. Perego, and F. Silvestri. Adaptive and resource-aware mining of frequent sets. In *International Conference on Data Mining (ICDM'02), Maebashi City, Japan*, pages 338–345, 2002.

[20] R. Rymon. Search through systematic set enumeration. In B. Nebel, C. Rich, and W. R. Swartout, editors, *International Conference on Principles of Knowledge Representation and Reasoning (KR'92), Cambridge, USA*, pages 539–550. Morgan Kaufmann, 1992.

[21] T. Uno and K. Satoh. Detailed description of an algorithm for enumeration of maximal frequent sets with irredundant dualization. In B. Goethals and M. Zaki, editors, *ICDM 2003 Workshop on Frequent Itemset Mining Implementations (FIMI '03), Melbourne, Florida, USA*, volume 90 of *CEUR Workshop Proceedings*, 2003.

[22] M.-J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. In *International Conference on Knowledge Discovery and Data Mining (KDD-97), Newport Beach, California, USA*, pages 283–286. AAAI Press, 1997.