ExAnte: Anticipated Data Reduction in Constrained Patterns Mining

Francesco Bonchi, Fosca Giannotti
ISTI - C.N.R.
Area della Ricerca di Pisa
Via Giuseppe Moruzzi, 1 - 56124 Pisa, Italy
bonchi@di.unipi.it,
f.giannotti@cnuce.cnr.it

Alessio Mazzanti, Dino Pedreschi [*]
Dipartimento di Informatica
Università di Pisa
Via F. Buonarroti 2, 56127 Pisa, Italy
mazzanta@cli.di.unipi.it,
pedre@di.unipi.it

ABSTRACT

Constraint pushing techniques have been proven to be effective in reducing the search space in the frequent pattern mining task, and thus in improving efficiency. But while pushing anti-monotone constraints in a level-wise computation of frequent itemsets has been recognized to be always profitable, the case is different for monotone constraints. In fact, monotone constraints have been considered harder to push in the computation and less effective in pruning the search space. In this paper, we show that this prejudice is not founded and introduce ExAnte, a pre-processing data reduction algorithm which reduces dramatically both the search space and the input dataset in constrained frequent patterns mining. Experimental results show a reduction of orders of magnitude, thus enabling a much easier mining task. ExAnte can be used as a pre-processor with any constrained patterns mining algorithm.

Keywords: Data Mining, Constrained Frequent Patterns Discovery, Pre-Processing.

ACM Computing Classification System: H.2.8 Database Application - Data Mining

1. INTRODUCTION

Constrained itemsets mining is a hot research theme in data mining [3, 6, 7, 8, 9, 10, 11, 12]. The most studied constraint is the frequency constraint, whose anti-monotonicity is used to reduce the exponential search space of the problem. Exploiting the anti-monotonicity of the frequency constraint is known as *apriori trick* [1, 2]: it dramatically reduces the search space making the computation feasible. Frequency is not only computationally effective, it is also semantically important since frequency provides "support" to any discovered knowledge. For these reasons frequency is the base constraint of what is generally referred to as *frequent itemsets mining*. However, many other constraints can facilitate user-focussed exploration and control, as well as reduce the computation. For instance, a user could be interested in mining all frequently purchased itemsets having a total price greater than a given threshold and containing at least two products of a given brand. Among these constraints, classes have been individuated which exhibit nice properties. The class of anti-monotone constraints is the most effective and easy to use in order to prune the search space. Since any conjunction of anti-monotone constraints is in turn anti-monotone, we can use the *apriori trick* to exploit completely the pruning power of the conjunction: the more anti-monotone constraints, the more selective the *apriori trick* will be.

The dual class, monotone constraints, has been considered more complicated to exploit and less effective in pruning the search space. As highlighted by Boulicaut and Jeudy in [3], pushing monotone constraints can lead to a reduction of anti-monotone pruning. Therefore, when dealing with a conjunction of monotone and anti-monotone constraints we face a tradeoff between anti-monotone and monotone pruning. Our observation is that the above consideration holds only if we focus completely on the search space of all itemsets, which is the approach followed by the work done so far.

In this paper we show that the most effective way of attacking the problem is to reason on both the itemsets search space and the transactions input database *together*. In this way, pushing monotone constraints does not reduce anti-monotone pruning opportunities, on the contrary, such opportunities are boosted. Dually, pushing anti-monotone constraints boosts monotone pruning opportunities: the two components strengthen each other recursively. We prove our previous statement by introducing ExAnte, a pre-processing data reduction algorithm which reduces dramatically both the search space and the input dataset in constrained frequent patterns mining.

[*] All authors are member of the Pisa KDD Laboratory: http://www-kdd.cnuce.cnr.it/
The present research is founded by "Fondazione Cassa di Risparmio di Pisa" under the "WebDigger Project"

ExAnte can exploit any constraint which has a monotone component, therefore also succinct monotone constraints [9] and convertible monotone constraints [10, 11] can be used to reduce the mining computation. Being a preprocessing algorithm, ExAnte can be coupled with any constrained patterns mining algorithm, and it is always profitable to start any constrained patterns computation with an ExAnte preprocess. The correctness of ExAnte is formally proven in this paper, by showing that the reduction of items and transaction database does not affect the set of constrained frequent patterns, which are solutions to the given problem, as well as their support. We discuss a through experimentation of the algorithm, which points out how effective the reduction is, and which potential benefits it offers to subsequent frequent pattern computation.

Our contributions:

Summarizing, the data reduction algorithm proposed in this paper is characterized by the following:

- ExAnte uses for the first time, the real synergy of monotone and anti-monotone constraints to prune the search space and the input dataset: the total benefit is greater than the sum of the two individual benefits.
- ExAnte can be used with any constraint which has a monotone component: therefore also succinct monotone constraints and convertible monotone constraints can be exploited.
- ExAnte maintains the exact support of each solution itemsets: a necessary condition if we want to compute Association Rules.
- ExAnte can be used to make feasible the discovery of particular patterns which can be discovered only at very low support level, for which the computation is unfeasible for traditional algorithms.
- Being a pre-processing algorithm, ExAnte can be coupled with any constrained patterns mining algorithm, and it is always profitable to start any constrained patterns computation with an ExAnte preprocess.
- ExAnte is efficient and effective: even a very large input dataset can be reduced of an order of magnitude in a small computation.
- A thorough experimental study has been performed with different monotone constraints on various datasets (both real world and synthetic datasets), and the results are described in details.

2. PROBLEM DEFINITION

Let $Items = \{x_1, ..., x_n\}$ be a set of distinct literals, usually called **items**. An **itemset** X is a non-empty subset of $Items$. If $k = |X|$ then X is called a **k-itemset**. A **transaction** is a couple $\langle tID, X\rangle$ where tID is the transaction identifier and X is the content of the transaction (an itemset). A **transaction database** TDB is a set of transactions. An itemset X is **contained** in a transaction $\langle tID, Y\rangle$ if $X \subseteq Y$. Given a transaction database TDB the subset of transaction which contain an itemset X is named $TDB[X]$. The **support** of an itemset X, written $supp_{TDB}(X)$ is the cardinality of $TDB[X]$. Given a user-defined **minimum support** δ, an itemset X is called **frequent** in TDB if $supp_{TDB}(X) \geq \delta$. This the definition of the frequency constraint $\mathcal{C}_{freq}[TDB]$: if X is frequent we write $\mathcal{C}_{freq}[TDB](X)$ or simply $\mathcal{C}_{freq}(X)$ when the dataset is clear from the context.

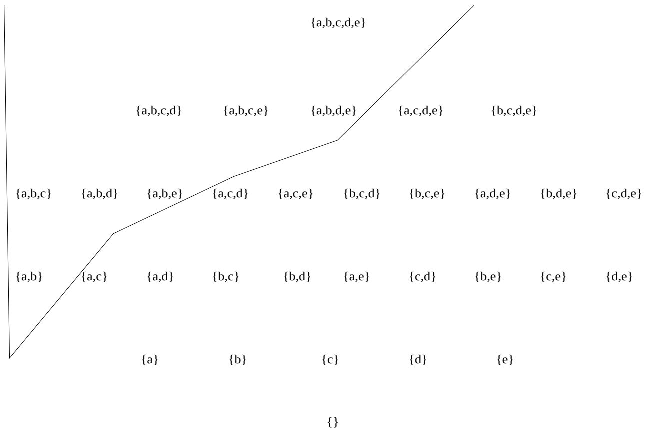

Figure 1: Itemset lattice for $Items = \{a, b, c, d, e\}$. The portion of lattice of supersets of $\{a, b\}$ is highlighted.

Let $Th(\mathcal{C}) = \{X|\mathcal{C}(X)\}$ denotes the set all itemsets X that satisfy constraint $\mathcal{C}$. The *frequent itemset mining problem* requires to compute the set of all frequent itemsets $Th(\mathcal{C}_{freq})$. In general given a conjunction of constraints $\mathcal{C}$ the *constrained itemset mining problem* requires to compute $Th(\mathcal{C})$; the *constrained frequent itemsets mining problem* requires to compute $Th(\mathcal{C}_{freq}) \cap Th(\mathcal{C})$.

We now formally define the notion of anti-monotone and monotone constraints.

Definition 1. Given an itemset X, a constraint $\mathcal{C}_{AM}$ is anti-monotone if

$$\forall Y \subseteq X : \mathcal{C}_{AM}(X) \Rightarrow \mathcal{C}_{AM}(Y)$$

If $\mathcal{C}_{AM}$ holds for X then it holds for any subset of X.

The frequency constraint is clearly anti-monotone. This property is used by the APRIORI algorithm with the following heuristic: if an itemset X does not satisfy $\mathcal{C}_{freq}$, then no superset of X can satisfy $\mathcal{C}_{freq}$, and hence they can be pruned. This pruning can affect a large part of the search space, since itemsets form a lattice as shown in Figure 1. Therefore the APRIORI algorithm operates in a level-wise fashion moving bottom-up on the itemset lattice, and each time it finds an infrequent itemset it prunes away all its supersets.

Definition 2. Given an itemset X, a constraint $\mathcal{C}_M$ is monotone if:

$$\forall Y \supseteq X : \mathcal{C}_M(X) \Rightarrow \mathcal{C}_M(Y)$$

independently from the given input transaction database. If $\mathcal{C}_M$ holds for X then it holds for any superset of X.

Monotone constraint	$\mathcal{C}_M \equiv$
cardinality	$\lvert X\rvert \geq n$
sum of prices	$sum(X.prices) \geq n$
maximum price	$max(X.prices) \geq n$
minimum price	$min(X.prices) \leq n$
range of prices	$range(X.prices) \geq n$

Table 1: Monotone constraints considered in our analysis.

Note that in the last definition we have required a monotone constraint to be satisfied independently from the given input transaction database. This is necessary since we want to distinguish between simple monotone constraints and global constraints such as the *"infrequency constraint"*:

$$supp_{TDB}(X) \leq \delta.$$

This constraint is still monotone but has different properties since it is dataset dependent and it requires dataset scans in order to be computed. Obviously, since our pre-processing algorithm reduces the transaction dataset, we want to exclude the infrequency constraint from our study. Thus, our study focuses on "local" monotone constraints, in the sense that they depend exclusively on the properties of the itemset (as those ones in Table1), and not on the underlying transaction database.

The general problem that we consider in this paper is the mining of itemsets which satisfy a conjunction of monotone and anti-monotone constraints:

$$Th(\mathcal{C}_{AM}) \cap Th(\mathcal{C}_M).$$

Since any conjunction of anti-monotone constraints is an anti-monotone constraint, and any conjunction of monotone constraints is a monotone constraint, we just consider two constraints: one per class. In particular, we choose frequency ($\mathcal{C}_{AM} \equiv supp_{TDB}(X) \geq \delta$) as anti-monotone constraint, in conjunction with various simple monotone constraints (see Table1).

$$Th(\mathcal{C}_{freq}) \cap Th(\mathcal{C}_M).$$

Problem Characterization

The concept of border is useful to characterize the solution space of the given problem. However, this notion is not directly exploited by ExAnte.

Definition 3. Given an anti-monotone constraint $\mathcal{C}_{AM}$ and a monotone constraint $\mathcal{C}_M$ we define their borders as:

$$B(\mathcal{C}_{AM}) = \{X \mid \forall Y \subset X : \mathcal{C}_{AM}(Y) \wedge \forall Z \supset X : \neg\, \mathcal{C}_{AM}(Z)\}$$

$$B(\mathcal{C}_M) = \{X \mid \forall Y \supset X : \mathcal{C}_M(Y) \wedge \forall Z \subset X : \neg\, \mathcal{C}_M(Z)\}$$

Moreover, we distinguish between positive and negative borders. Given a general constraint $\mathcal{C}$ we define:

$$B^+(\mathcal{C}) = B(\mathcal{C}) \cap Th(\mathcal{C})$$

$$B^-(\mathcal{C}) = B(\mathcal{C}) \cap Th(\neg\, \mathcal{C})$$

In Figure 2 we show the borders of two constraints: the anti-monotone constraint $supp(X) \geq 2$, and the monotone one $sum(X.prices) \geq 14$. In the given situation the borders are:

$$B^+(\mathcal{C}_M) = \{e, acd, bcd\} \quad B^+(\mathcal{C}_{AM}) = \{bcd, bce, ade\}$$
$$B^-(\mathcal{C}_M) = \{cd, abc, abd\} \quad B^-(\mathcal{C}_{AM}) = \{ab, ac, bde, cde\}$$

The solutions to our problem are the itemsets that lie in between the two borders: under the anti-monotone border and over the monotone border:

$$R = \{e, ae, be, ce, de, bcd, bce, ade\}$$

The next theorem proves algebraically what we have just seen graphically.

Theorem 1.

$$X \in (Th(\mathcal{C}_{AM}) \cap Th(\mathcal{C}_M)) \Leftrightarrow \exists Y \in B^+(\mathcal{C}_{AM}) : X \subseteq Y \quad \wedge$$
$$\exists Z \in B^+(\mathcal{C}_M) : X \supseteq Z$$

Proof. Trivially by definition of border:

$$X \in Th(\mathcal{C}_{AM}) \Leftrightarrow \mathcal{C}_{AM}(X) \Leftrightarrow \exists Y \in B^+(\mathcal{C}_{AM}) : X \subseteq Y$$

$$X \in Th(\mathcal{C}_M) \Leftrightarrow \mathcal{C}_M(X) \Leftrightarrow \exists Z \in B^+(\mathcal{C}_M) : X \supseteq Z$$

□

The ExAnte algorithm proposed in this paper allows to reduce the transaction database and the relevant 1-itemsets without affecting the set of solutions $Th(\mathcal{C}_{freq}) \cap Th(\mathcal{C}_M)$ and without exploiting the notion of border.

3. SEARCH SPACE AND INPUT DATA REDUCTION

As already stated, if we focus only on the itemsets lattice, pushing monotone constraint can lead to a less effective anti-monotone pruning. Consider again Figure 1 and suppose that $\{a, b\}$ have been removed from the search space because it does not satisfy some monotone constraints $\mathcal{C}_M$. This pruning avoids checking support for $\{a, b\}$. But it may be that if we check support, $\{a, b\}$ could result to be infrequent, and thus all its supersets could be pruned away. By monotone pruning $\{a, b\}$ we risk to loose anti-monotone pruning opportunities given from $\{a, b\}$ itself. The tradeoff is clear [3]: pushing monotone constraint can save tests on anti-monotone constraints, however the results of these tests could have lead to more effective pruning.

In order to obtain a real amalgam of the two opposite pruning strategies we have to consider the constrained frequent patterns problem in its whole: not focussing only on the itemsets lattice but considering it together with the input database of transactions. In fact, as proved by the theorems in the following section, monotone constraints can prune away transactions from the input dataset *without loosing solutions.* This monotone pruning of transactions has got another positive effect: while reducing the number of transactions in input it reduces the support of items too, hence the total number of frequent 1-itemsets. In other words,

tID	itemset
1	a,b,c
2	d
3	b,c,e
4	b,c,d,e
5	e
6	a,d,e
7	b,c,d
8	a,d,e

item	price
a	4
b	3
c	5
d	6
e	15

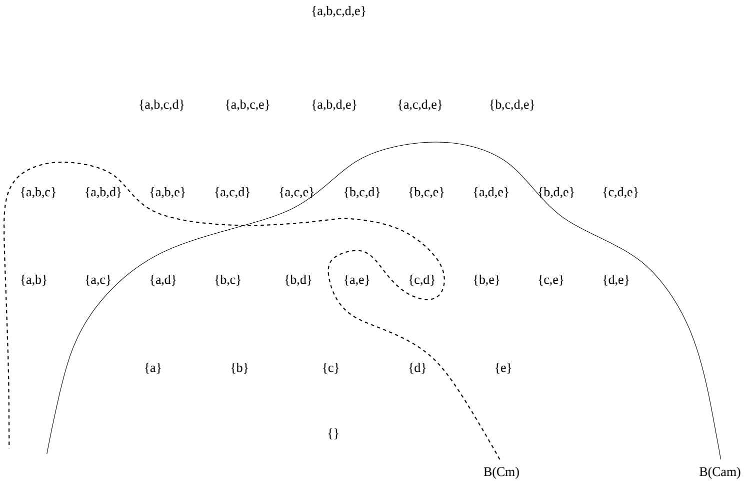

Figure 2: The borders $B(\mathcal{C}_M)$ and $B(\mathcal{C}_{AM})$ for the transaction database and the price table on the left with $\mathcal{C}_M \equiv sum(X.prices) \geq 14$ and $\mathcal{C}_{AM} \equiv supp_{TDB}(X) \geq 2$.

the monotone pruning of transactions strengthens the anti-monotone pruning.

Moreover, infrequent items can be deleted by the computation and hence pruned away from the transactions in the input dataset. This anti-monotone pruning has got another positive effect: reducing the size of a transaction which satisfies a monotone constraint can make the transaction violates the monotone constraint. Therefore a growing number of transactions which do not satisfy the monotone constraint can be found. We are clearly inside a loop where two different kinds of pruning cooperates to reduce the search space and the input dataset, strengthening each other step by step until no more pruning is possible (a fix-point has been reached). This is precisely the idea underlying ExAnte.

3.1 ExAnte Properties

In this section we formalize the basic ideas of ExAnte. First we define the two kinds of reduction, than we prove the completeness of the method. In the next section we provide the pseudo-code of the algorithm.

Definition 4. [μ-reduction] Given a transaction database TDB and a monotone constraint $\mathcal{C}_M$, we define the μ-reduction of TDB as the dataset resulting from pruning the transactions that do not satisfy $\mathcal{C}_M$.

$$\mu[TDB]_{\mathcal{C}_M} = Th(\mathcal{C}_M) \cap TDB$$

(Recall here that a transaction is an itemset).

Definition 5. [α-reduction] Given a transaction database TDB, a transaction $\langle tID, X\rangle$ and a frequency constraint $\mathcal{C}_{freq}[TDB]$, we define the α-reduction of $\langle tID, X\rangle$ as the subset of items in X that satisfy $C_{freq}[TDB]$.

$$\alpha[\langle tID, X\rangle]_{\mathcal{C}_{freq}[TDB]} = F_1 \cap X$$

Where: $F_1 = \{I \in Items | \{I\} \in Th(C_{freq}[TDB])\}$. We define the α-reduction of TDB as the dataset resulting from the α-reduction of all transactions in TDB.

Example:

$Items = \{a, b, c, d, e, f, g\} \quad X = \{a, c, d, f, g\}$
$Th(C_{AM}) = \{\{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$
$F_1 = \{a, b, c\}$
$\alpha[\langle tID, X\rangle]_{\mathcal{C}_{AM}} = F_1 \cap X = \{a, c\}$

The following two key theorems state that we can always μ-reduce and α-reduce a dataset without reducing the support of solution itemsets. Moreover, since satisfaction of $\mathcal{C}_M$ is independent from the transaction dataset, all solution itemsets will still satisfy it. Therefore, we can always μ-reduce and α-reduce a dataset without loosing solutions.

THEOREM 2 (μ-REDUCTION CORRECTNESS). *Given a transaction database TDB, a monotone constraint $\mathcal{C}_M$, and a frequency constraint $\mathcal{C}_{freq}$, we have that:*

$$\forall X \in Th(\mathcal{C}_{freq}[TDB]) \cap Th(\mathcal{C}_M) :$$
$$supp_{TDB}(X) = supp_{\mu[TDB]_{\mathcal{C}_M}}(X).$$

PROOF. Since $X \in Th(\mathcal{C}_M)$, all transactions containing X will also satisfy $\mathcal{C}_M$ for the monotonicity property. In other words: $TDB[X] \subseteq \mu[TDB]_{\mathcal{C}_M}$. This implies that:

$$supp_{TDB}(X) = supp_{\mu[TDB]_{\mathcal{C}_M}}(X).$$

□

THEOREM 3 (α-REDUCTION CORRECTNESS). *Given a transaction database TDB, a monotone constraint $\mathcal{C}_M$, and a frequency constraint $\mathcal{C}_{freq}$, we have that:*

$$\forall X \in Th(\mathcal{C}_{freq}[TDB]) \cap Th(\mathcal{C}_M) :$$
$$supp_{TDB}(X) = supp_{\alpha[TDB]_{\mathcal{C}_{freq}}}(X).$$

PROOF. Since $X \in Th(\mathcal{C}_{freq})$, all subsets of X will be frequent (by the anti-monotonicity of frequency). Therefore no subset of X will be α-pruned (in particular, no 1-itemsets in X). This implies that:

$$supp_{TDB}(X) = supp_{\alpha[TDB]_{C_{freq}}}(X).$$

□

3.2 ExAnte Algorithm

The two theorems above suggest a fix-point computation. ExAnte starts the first iteration as any frequent patterns mining algorithm: counting the support of singleton items. Items that are not frequent are thrown away once and for all. But during this first count only transactions that satisfy C_M are considered. The other transactions are signed to be pruned from the dataset (μ-reduction). Doing so we reduce the number of interesting 1-itemsets. Even a small reduction of this number represents a huge pruning of the search space. At this point ExAnte deletes from alive transactions all infrequent items (α-reduction). This pruning can reduce the monotone value (for instance, the total sum of prices) of some alive transactions, possibly resulting in a violation of the monotone constraints. Therefore we have another opportunity of μ-reducing the dataset. But μ-reducing the dataset we create new opportunities for α-reduction, which can turn in new opportunities for μ-reduction, and so on, until a fix-point is reached. the pseudo-code of ExAnte algorithm follows:

```
Procedure: ExAnte($TDB, \mathcal{C}_M, min\_supp$)
$I := \emptyset$;
forall tuples $t$ in $TDB$ do
    if $\mathcal{C}_M(t)$ then forall items $i$ in $t$ do
        $i.count$++; if $i.count \geq min\_supp$ then $I := I \cup i$;
$old\_number\_interesting\_items := |Items|$;
while $|I| < old\_number\_interesting\_items$ do
    $TDB := \alpha[TDB]_{C_{freq}}$;
    $TDB := \mu[TDB]_{C_M}$;
    $old\_number\_interesting\_items := |I|$;
    $I := \emptyset$;
    forall tuples $t$ in $TDB$ do
        forall items $i$ in $t$ do
        $i.count + +$;
        if $i.count \geq min\_supp$ then $I := I \cup i$;
end while
```

Clearly, a fix-point is eventually reached after a finite number of iterations, as at each step the number od alive items strictly decreases.

3.3 Run-through Example

Suppose that the transaction and price dataset in Table 2 are given. Suppose that we want to compute frequent itemsets ($min\_supp = 4$) with a sum of prices ≥ 45.

During the first iteration the total price of each transaction is checked to avoid using transactions which do not satisfy the monotone constraint. All transaction with a sum of prices ≥ 45 are used to count the support for the singleton items. Only the fourth transaction is discarded. At the end of the count we find items a, e, f and h to be infrequent. Note that, if the fourth transaction had not been discarded, items a and e would have been counted as frequent. At this point we perform an α-reduction of the dataset: this means removing a, e, f and h from all transactions in the dataset. After the α-reduction we have more opportunities to μ-reduce the dataset. In fact transaction 2, which at the beginning has a total price of 63, now has its total price reduced to 38 due to the pruning of a and e. This transaction can now be pruned away. The same reasoning holds for transactions number 7 and 9. At this point ExAnte counts once again the support of alive items with the reduced dataset. The item g which initially has got a support of 5 now has become infrequent (see Table 3 for items support iteration by iteration). We can α-reduce again the dataset, and then μ-reduce. After the two reductions transaction number 5 does not satisfy anymore the monotone constraint and it is pruned away. ExAnte counts again the support of items on the reduced datasets but no more items are found to have turned infrequent.

item	price
a	5
b	8
c	14
d	30
e	20
f	15
g	6
h	12

tID	Itemset	Total price
1	b,c,d,g	58
2	a,b,d,e	63
3	b,c,d,g,h	70
4	a,e,g	31
5	c,d,f,g	65
6	a,b,c,d,e	77
7	a,b,d,f,g,h	76
8	b,c,d	52
9	b,e,f,g	49

Table 2: Run-through Example: price table and transaction database.

Supports			
Items	1_{st}	2_{nd}	3_{rd}
a	3	†	†
b	7	4	4
c	5	5	4
d	7	5	4
e	3	†	†
f	3	†	†
g	5	3	†
h	2	†	†

Table 3: Run-through Example: items and their supports iteration by iteration.

The fix-point has been reached at the third iteration: the dataset has been reduced from 9 transactions to 4 transactions (number 1,3,6 and 8), and interesting itemsets have shrunk from 8 to 3 (b, c and d). At this point any constrained frequent pattern algorithm would find very easily the unique solution to problem which is the 3-itemset $\langle b, c, d \rangle$.

4. EXPERIMENTAL RESULTS

In this section we deeply describe the experimental study that we have conducted with different monotone constraints on various datasets. In particular, the monotone constraints used in the experimentation are in Table 1.

Dataset	Transactions	Items	Max Trans Size	Avg Trans Size
POS	515,597	1657	164	6.5
IBM	8,533,534	100,000	37	11.21
FoodMart	54537	1560	28	4.6
Italian	186,824	4800	31	10.42

Table 4: Characteristics of datasets used in the experiments.

Dataset	Min Price	Max Price	Avg Price
FoodMart	0.5	3.98	2
Italian	100	900,000	6454.87

Table 5: Characteristics of the price datasets used in the experiments.

Iteration	Transactions	1-itemsets
0	8533534	2367
1	1033508	398
2	323519	296
3	280186	289
4	278288	289
Execution time: 45.6 sec		

Table 6: Execution of ExAnte on Dataset "IBM" with min_supp = 7000 and cardinality $\geq$ 6

Iteration	Transactions	1-itemsets
0	17306	2010
1	13167	1512
2	11295	1205
3	10173	1025
4	9454	901
5	9005	835
6	8730	785
7	8549	754
8	8431	741
9	8397	736
10	8385	734
11	8343	729
12	8316	726
13	8312	724
14	8307	722
15	8304	722
Execution time: 1.5 sec		

Table 7: Execution of ExAnte on Dataset "Italian" with min_supp = 40 and sum of prices $\geq$ 100000

In addition, we have experimented a harder to exploit constraint: $avg(X.prices) \geq n$. This constraint is clearly neither monotone nor anti-monotone, but can exhibit a monotone (or anti-monotone) behavior if items are ordered by ascending (or descending) price, and frequent patterns are computed following a prefix-tree approach. This class of constraints, named *convertible*, has been introduced in [10]. In our experiments the constraint $avg(X.prices) \geq n$ is treated by inducing a conjunction of two weaker monotone constraints: $sum(X.prices) \geq n$ and $max(X.prices) \geq n$.

Note that in every reported experiment we have chosen monotone constraints thresholds that are not very selective: there are always solutions to the given problem.

The test bed architecture used in our experiments was a Windows2000 based personal computer, equipped with a Pentium III processor running at 866MHz and 1GB RAM. ExAnte was implemented using Microsoft Visual C++ 6.0.

For a more detailed report of our experiments see [5].

4.1 Datasets used

In our experiments we have used four datasets with different characteristics (see Table 4). The first dataset, named "POS", was used in the KDD-Cup 2000 competition and it is described in [13]. The dataset is available from the KDD-Cup 2000 home page[1]. "POS" is a real world dataset containing several years worth of point-of-sale data from a large electronic retailer, aggregated at the product category level.

"IBM" is a synthetic dataset obtained with the most commonly adopted dataset generator, available from IBM Almaden[2]. We have generate a very large dataset since we have not been able to find a real-world dataset over one million transactions.

"FoodMart" has been obtained from the FoodMart2000 database which is provided as demo with Microsoft SQL Server. We have constructed transactions grouping by CustomerID and TimeID at the product level. We have chosen this database in order to have real transactions together with real products prices. In Table 5 characteristics of the price database are reported.

"Italian" is another real-world dataset obtained from an Italian supermarket chain within a market-basket analysis project conducted by our research lab, few years ago (note that the prices are in the obsolete currency Italian Lira).

4.2 Data reduction

In Table 6 and 7 two typical executions of ExAnte are reported. The first one exhibits a data reduction of one order of magnitude on both the number of transactions and the number of interesting 1-itemsets. The fix point is reached at the fourth iteration with a very efficient computation (considering the size of the input dataset). The second execution exhibits a progressive data reduction without huge gaps, which terminates at the fifteenth iteration. The resulting number of interesting 1-itemsets is around one third of the initial number: recall that even a small reduction of interesting 1-itemsets represents a very large reduction of the search space as shown in the next section. These graphs confirm that stronger monotone and anti-monotone constraints yield more effective items and transaction reduction by means of ExAnte.

In Figure 3 the reduction of the number of transactions w.r.t

[1] http://www.ecn.purdue.edu/KDDCUP/

[2] http://www.almaden.ibm.com/cs/quest/syndata.html#assocSynData

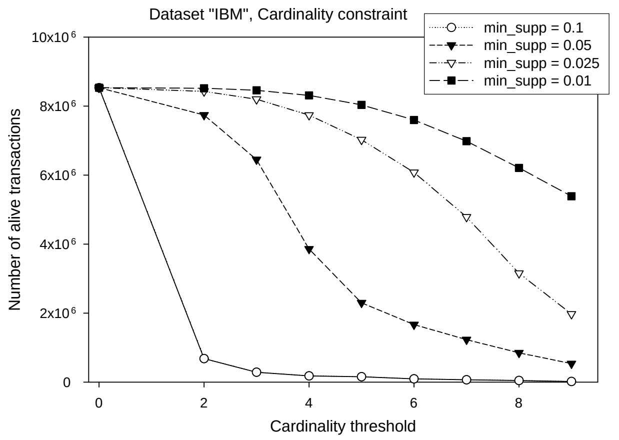

Figure 3: Transactions reduction on dataset "IBM"

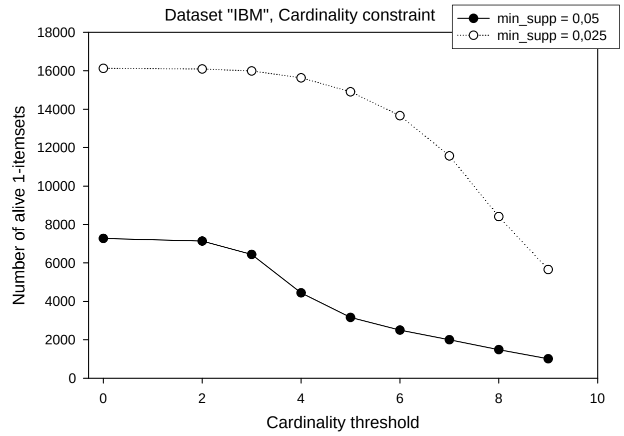

Figure 4: Reduction of interesting 1-itemsets on dataset "IBM".

the cardinality threshold is shown for four different support thresholds on the synthetic dataset. When the cardinality threshold is equals to zero the number of transactions is obviously as the total number of transactions in the database, since there is no monotone pruning. Already for a low support threshold as 0.1% with a cardinality constraint equals to 2 the number of transactions decreases dramatically. Figure 4 describes the reduction of number of interesting 1-itemsets on the same dataset.

4.3 Search space reduction

As already stated, even a small reduction in the number of relevant 1-itemsets represents a very large pruning of the search space. In our experiments, as a measure of the search space explored, we have considered the number of candidate itemsets generated by a levelwise algorithm such as Apriori. In Figure 5 is reported a comparison of the number of candidate itemsets generated by Apriori and by *ExAnteApriori* (ExAnte pre-processing followed by Apriori) on the "Italian" dataset with various constraints. The dramatic search space reduction is evident, and it will be confirmed by computation time reported in the next section. The same comparison on dataset "POS" with the cardinality constraint is reported in Figure 6.

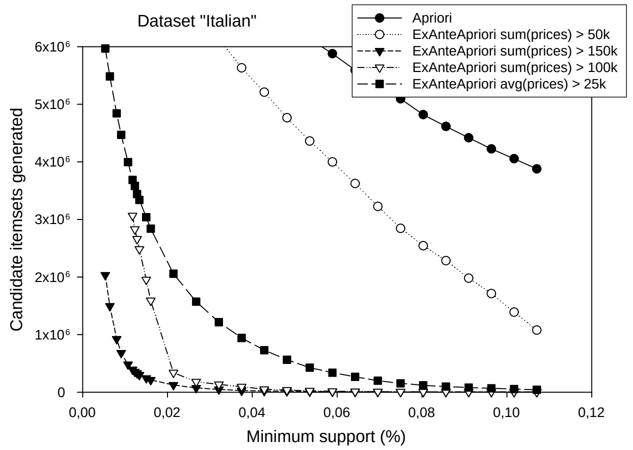

Figure 5: Search space reduction on dataset "Italian".

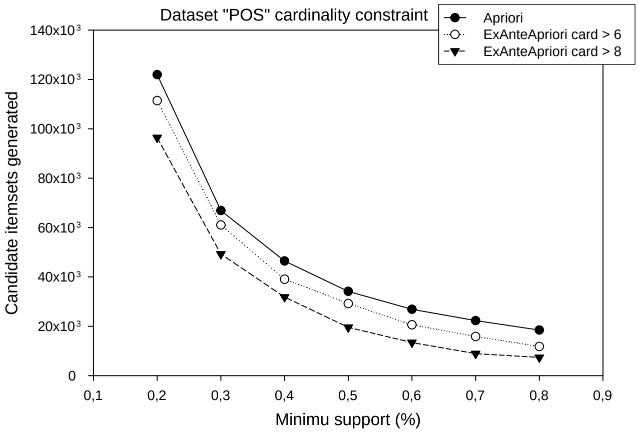

Figure 6: Search space reduction on dataset "POS".

How the number of candidate itemsets shrinks by increasing strength of the monotone constraint is shown in Figure 7. This figure also highlights another interesting feature of ExAnte: even at very low support level (min_supp = 5 on a dataset of 186,824 transactions) the frequent patterns computation is feasible if coupled with a monotone constraint. Therefore, ExAnte can be used to make feasible the discovery of particular patterns which can be discovered only at very low support level, for instance:

- extreme purchasing behaviors (such as patterns with a very high average of prices);
- very long patterns (using the cardinality constraint coupled with a very low support threshold).

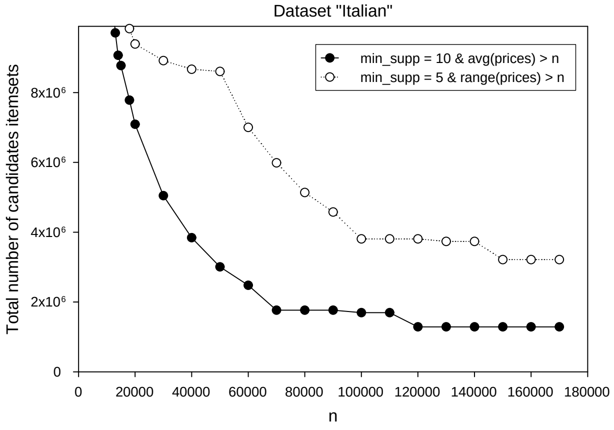

Figure 7: Search space reduction with different constraints.

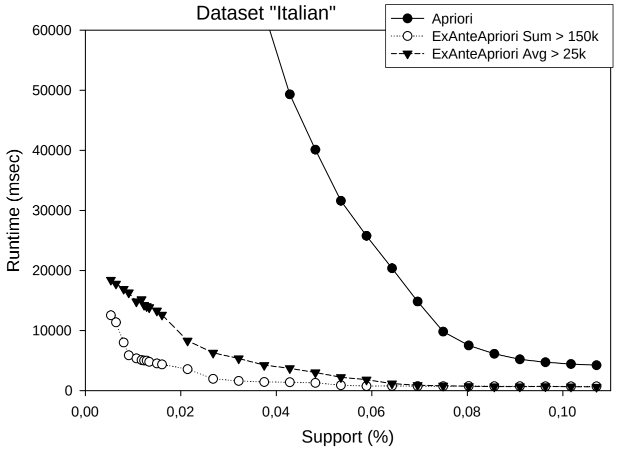

Figure 8: Runtime comparison between Apriori and ExAnteApriori with various constraints.

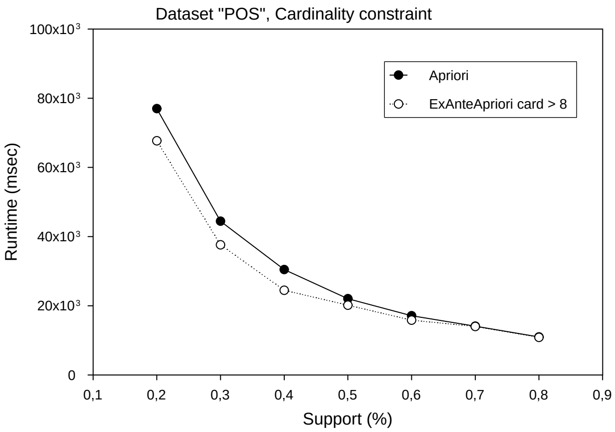

Figure 9: Runtime comparison between Apriori and ExAnteApriori with Cardinality constraint.

4.4 Time reduction

In this section we report time comparison between Apriori and ExAnteApriori (ExAnte pre-processing followed by Apriori). We have chosen Apriori as the "standard" frequent pattern mining algorithm. Recall that every frequent pattern mining algorithm can be coupled with ExAnte preprocessing obtaining similar benefits. Execution time is reported in Figure 8 (dataset "Italian", sum and average constraints) and Figure 9 (dataset "POS", cardinality constraint). The large search space pruning reported in the previous section is here confirmed by the execution time.

5. RELATED WORK

Being a pre-processing algorithm, ExAnte can not be directly compared with any previously proposed algorithm for constrained frequent pattern mining. However, it would be interesting to couple ExAnte data reduction with those algorithms and to measure the improve in efficiency. Among constrained frequent pattern mining algorithms, we would like to mention $\mathcal{FIC}^{\mathcal{M}}$ [11] and the recently proposed DualMiner [4].

6. CONCLUSIONS AND FUTURE WORK

In this paper we have introduced ExAnte, a pre-processing data reduction algorithm which reduces dramatically the search space the input dataset, and hence the execution time, in constrained frequent patterns mining. We have proved experimentally the effectiveness of our method, using different constraints on various datasets. Due to its capacity in focussing any particular instance of the problem, ExAnte exhibits very good performance also when one of the two constraints (the anti-monotone or the monotone) is not very selective. This feature makes ExAnte useful to discover particular patterns which can be discovered only at very low support level, for which the computation is unfeasible for traditional algorithms.

We are actually developing a new algorithm for constrained frequent pattern mining, which will take full advantage of ExAnte pre-processing. We are also interested to study in which other mining tasks ExAnte can be useful. We will investigate its applicability to constrained sequential patterns, and to the discovery of anomalies and outliers in data cubes.

ExAnte executable can be downloaded by our web site:
http://www-kdd.cnuce.cnr.it/

7. REFERENCES

[1] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., 26–28 May 1993.

[2] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In *Proceedings of the Twentieth International Conference on Very Large Databases*, pages 487–499, Santiago, Chile, 1994.

[3] J.-F. Boulicaut and B. Jeudy. Using constraints during set mining: Should we prune or not? In *Actes*

des Seizime Journes Bases de Donnes Avances BDA'00, Blois (F), pages 221–237, 2000.

[4] C. Bucila, J. Gehrke, D. Kifer, and W. White. Dualminer: A dual-pruning algorithm for itemsets with constraints. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002.

[5] F.Bonchi, F.Giannotti, A.Mazzanti, and D.Pedreschi. Exante: a preprocessing algorithm for constrained frequent pattern mining. Technical Report ISTI-B4-2003-07, ISTI, 2003.

[6] G. Grahne, L. Lakshmanan, and X. Wang. Efficient mining of constrained correlated sets. In *16th International Conference on Data Engineering (ICDE' 00)*, pages 512–524. IEEE, 2000.

[7] J. Han, L. V. S. Lakshmanan, and R. T. Ng. Constraint-based, multidimensional data mining. *Computer*, 32(8):46–50, 1999.

[8] L. V. S. Lakshmanan, R. T. Ng, J. Han, and A. Pang. Optimization of constrained frequent set queries with 2-variable constraints. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 28(2), 1999.

[9] R. T. Ng, L. V. S. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained associations rules. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD-98)*, volume 27,2 of *ACM SIGMOD Record*, pages 13–24, New York, June 1–4 1998. ACM Press.

[10] J. Pei and J. Han. Can we push more constraints into frequent pattern mining? In R. Ramakrishnan, S. Stolfo, R. Bayardo, and I. Parsa, editors, *Proceedinmgs of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'00)*, pages 350–354, N. Y., Aug. 20–23 2000. ACM Press.

[11] J. Pei, J. Han, and L. V. S. Lakshmanan. Mining frequent item sets with convertible constraints. In *(ICDE'01)*, pages 433–442, 2001.

[12] R. Srikant, Q. Vu, and R. Agrawal. Mining association rules with item constraints. In D. Heckerman, H. Mannila, D. Pregibon, and R. Uthurusamy, editors, *Proc. 3rd Int. Conf. Knowledge Discovery and Data Mining, KDD*, pages 67–73. AAAI Press, 14–17 Aug. 1997.

[13] Z. Zheng, R. Kohavi, and L. Mason. Real world performance of association rule algorithms. In F. Provost and R. Srikant, editors, *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 401–406, 2001.