Preprocessing for Frequent Pattern Mining through Data Reduction

Francesco Bonchi, Fosca Giannotti, Alessio Mazzanti, Dino Pedreschi

***Abstract*— Discovering frequent patterns in large datasets is one of the more pervasive data mining tasks. Albeit rooted in market basket analysis, frequent pattern mining can be adopted in many applications, and on data sources of different nature and structure; it also provides a basis for several other mining tasks, such as association rules, classification, and clustering. However, frequent pattern mining is inherently difficult, in that it handles typically too many input data, which typically yield too many patterns as a result – this is often an insuperable obstacle, both for performance limitations and for the impossibility to discern the interesting patterns from the many, mostly uninteresting, extracted ones. Preprocessing based on data reduction and user-specified constraints may be the solution to this problem: it may drive the mining process towards potentially interesting patterns, while enabling query optimizations at the same time. We show how this can be achieved on the basis of a simple yet powerful idea: combine constraints of different nature to the purpose of dramatically reducing the input database. The mining process after such preprocessing is strikingly optimized, both in terms of performance, and in capability of focussing on interesting patterns.**

***Index Terms*— Frequent pattern mining, constraints, preprocessing, data reduction.**

AS the information flood surrounds us in our everyday life, the need of techniques capable of filtering out uninformative data and focussing on the portion of data containing the patterns of interest, becomes more and more crucial. In fact, the knowledge we are seeking for is usually contained in a small set of local patterns, hidden into oceans of data that generate mountains of useless patterns. This situation – too many data yielding too many patterns – is harmful for two reasons. First, performance degrades: mining generally becomes inefficient or, often, simply unfeasible. Second, the identification of the fragments of interesting knowledge, blurred within a huge quantity of mostly useless patterns, is difficult.

We believe that the main objective of preprocessing should be to take our *size-large knowledge-sparse* data and give back to us *size-small knowledge-dense* data, possibly without losing any interesting pattern.

In the vision of *constraint-based data mining*, it is the user which specifies to the system what is *interesting* for the current application. She drives the mining session by providing to the mining system *constraints* which the extracted knowledge is expected to satisfy, in order to qualify as potentially interesting [9]. Constraints are user's defense against the information flood: on one hand they can shrink the search space and the input data of the knowledge-seeking problem, thus reducing time and resource requirements; on the other hand, they provide focus on the interesting knowledge, thus reducing the number of patterns extracted to those of potential interest.

Authors are member of KDD Lab, the *Knowledge Discovery and Delivery Laboratory*, Pisa, Italy.
URL: http://www-kdd.isti.cnr.it

One of the main research objectives at KDD Laboratory in Pisa is the definition of a data mining query language, supported by a system able to optimize such constraint-based data mining queries. We believe that the data analyst must have a high-level vision of the data mining system, without worrying about the details of the computational engine, in the very same way a database designer has not to worry about query optimization. The analyst must be provided with a set of primitives to be used to communicate with the data mining system, using a data mining query language. She just needs to declaratively specify in the data mining query how the desired patterns should look like and which conditions they should obey. Indeed, the task of composing all constraints and producing the most efficient mining strategy (execution plan) for the given data mining query should be left to an underlying *query optimizer*.

In this perspective, our Lab has invented *ExAnte* [4], a very simple albeit effective preprocessing technique for frequent patterns mining. ExAnte exploits constraints in order to reduce dramatically the data in analysis to that containing the patterns of interest. This data reduction, in turn, induces a strong reduction of the search space of candidate patterns, thus supporting dramatic performance improvements in subsequent mining and, sometimes, making feasible otherwise untractable mining tasks.

FREQUENT PATTERN MINING

Since frequency provides support to any extracted knowledge, it is the most used and maybe the most useful, measure of interest. For sure is the most studied: during the last decade a lot of researchers have focussed their studies on the computational problem of mining patterns which satisfy a user-defined minimum threshold of frequency [2], [7].

The simplest form of frequent pattern is the frequent itemset: given a database of transactions (a transaction is a set of items) we want to find those subsets of transactions (itemsets) which appear together frequently. Here frequently means a number of times no less than a given threshold. This computational problem is at the basis of the well known *Association Rules* mining.

The idea of mining association rules [1] originates from the analysis of *market-basket* data where we are interested in finding rules describing customers behavior in buying products. Their direct applicability to business problems together with their inherent understandability, even for non data mining

experts, made association rules a popular mining method, and made frequent itemsets mining one of the most hot research themes in data mining. However frequent itemsets are meaningful not only in the context of association rules mining: they can be used as basic element in many other kind of analysis, ranging from classification [10], [11] to clustering [13], [14].

Recently the research community has turn its attention to more complex kinds of frequent patterns extracted from more structured data: sequences, trees, and graphs. All these different kinds of pattern have different peculiarities and application fields, (i.e. sequences are particular well suited for business applications, frequent subtrees can be mined from a set of XML documents, and frequent substructures from graphs can be useful, for instance, in biological applications, in drug design and in Web-mining), but they all share the same computational aspects: an exponential search space and a nice property of frequency that dramatically prunes such search space.

The Antimonotonicity of Frequency

In classical frequent itemset mining, the popular *Apriori* algorithm [2] exploits an interesting property of frequency in order to prune the exponential search space of the problem: whenever the support of an itemset violates the frequency constraint (that is, its support falls below the user-defined minimum frequency threshold), then all its supersets can be pruned away from the search space, since they will violate the constraint too. This property of frequency is usually called *antimonotonicity*.

The Apriori algorithm explores the powerset of itemsets (the search space of the problem), testing frequency breadth-first, size-wise, from small itemsets to larger itemsets, and exploiting the antimonotonicity of frequency to prune the search space.

Frequency is not the unique antimonotone constraint: other constraints with the same nice property can be defined. For instance one could be interested in mining frequent itemsets with a total sum of prices $\leq$ $50\$$. In this case we have two antimonotone constraints in conjunction: the minimum frequency constraint and the maximum sum of prices which is antimonotone as well. In fact, if we have an itemset which violates such constraint (i.e. it has a sum of prices greater than $50\$$), we can be sure that all its supersets will violate the constraint. Adding more items to the given itemset will simply make it more expensive, so it will never satisfy the constraint. Similarly a constraint on the maximum size of interesting pattern is an antimonotone constraint: adding more items to an already-oversized itemset will never make it more appealing. Another user could be interested in frequent itemsets whose minimum price is greater than $10\$$. Even in this case we have an antimonotone constraint: if an itemset contain a cheap item, adding more items will not change the situation.

Such constraints can be pushed deeply down into the frequent pattern mining computation since they behave exactly as the frequency constraint: if they are not satisfiable at an early level (small pattern), they have no hope of becoming satisfiable later (larger pattern). Moreover, since any conjunction of antimonotone constraints is antimonotone as well, they can be exploited all together, in the same way of frequency, to prune the search space. The more antimonotone constraints the user specifies, the more selective the algorithm will be. This means stronger pruning, which in turn means faster computation and smaller number of patterns extracted. Indeed, it's always a very good thing to have antimonotone constraints.

cardinality	$\|X\| \geq n$
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 I
EXAMPLES OF MONOTONE CONSTRAINTS.

The case is more subtle for constraints which exhibit the opposite property to antimonotonicity.

The Tradeoff Between Antimonotone and Monotone Pruning

As already stated, the frequent pattern mining usually produces a huge quantity of patterns which are mainly uninteresting and very hard to read for the user. One reasonable user could decide to focus on frequent pattern which are large enough (more than a given threshold): she knows that she doesn't care about small patterns, which are too many and too little informative. Therefore she feeds her mining system with such constraint on size, forcing the search to focus on large patterns.

Another user more focussed on profit could ask to the mining system to mine only very profitable patterns, i.e. those patterns with a sum of prices no less than a given threshold. An essential question arises: "*How can the data mining system exploit such constraints in conjunction with the frequency constraint?*".

Before answering to this question we should highlight that these constraints behave exactly in the opposite way of frequency, and they are called *monotone constraints*. Whenever an itemsets satisfy a monotone constraint, so will do all its supersets (or the other way around: if an itemsets does not satisfy a monotone constraint, none of its subsets will satisfy the constraint as well). Since the frequency computation moves from small to large patterns, we can not push such constraints in it. At an early stage, if an itemset is too small or too cheap to satisfy a monotone constraint, we can not yet say nothing about its supersets. Perhaps, just adding a very expensive single item to the itemsets could raise the total sum of prices over the given threshold, thus making the resulting itemset satisfy the monotone constraint.

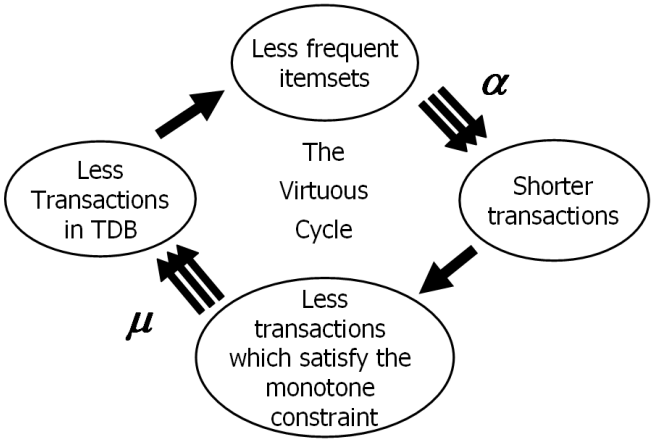
Therefore the answer to the question "*how to exploit monotone constraints in frequent pattern mining*" is not trivial. All the researchers which have approached the problem so far, have stated that there is an inherent difficulty in it [6], [12]. The difficulty is given by the tradeoff existing between antimonotone and monotone pruning.

Suppose that a pattern has been removed from the search space because it does not satisfy a monotone constraint. This pruning avoids checking support for this pattern, but on the

other hand, if we check its support and find it smaller than the frequency threshold, we may prune away all the supersets of this itemset. In other words, by monotone pruning we risk to lose antimonotone pruning opportunities. The tradeoff is clear: pushing monotone constraint can save frequency tests, however the results of these tests could have lead to more effective antimonotone pruning.

Therefore the answer to the question *"how to exploit monotone constraints in frequent pattern mining"* has been so far: *"maybe pushing monotone constraints is not always effective, and however it is not as effective as pushing antimonotone constraints"*.

This was the state of art before the introduction of ExAnte [4].



EXANTE: THE REAL SYNERGY OF ANTIMONOTONICITY AND MONOTONICITY

ExAnte is obtained by shifting attention from the pattern search space to the input data. Indeed, the tradeoff between antimonotone and monotone pruning exists only if we focus exclusively on the search space of the problem, which is the approach followed in the state-of-the-art. But if we take a step back and look at the overall problem, reasoning on *both* the search space and the input database *together* we can find the real synergy of antimonotonicity and monotonicity.

Instead of applying monotone constraints to the patterns, we apply them to the database in input.

Consider the problem of mining frequent itemsets which satisfy a monotone constraint, from a database of transactions. Recall that a transaction is nothing more than an itemset, therefore we can apply the monotone constraint to the transactions in input. The *ExAnte property* states that a transaction which does not satisfy the given monotone constraint can be deleted by the input database since it will never participate to the support count of any solution. In fact all subsets of a transaction which does not satisfy a monotone constraint, do not satisfy it too for the monotonicity property. No subset of the transaction is an interesting itemset, and thus the transaction is useless because it does not contain any interesting pattern. In other words, the transaction will never participate to the support count of an itemset which has some chances to be a solution to the given problem. Deleting such transactions will not reduce the real support of interesting patterns. We have named this pruning of transactions based on monotonicity *μ-reduction*.

A major consequence of reducing the input database in this way is that it implicitly reduces the support of a large amount of itemsets that do not satisfy the monotone constraint as well, resulting in a reduced number of candidate itemsets generated during the mining algorithm. Even a small reduction in the database can cause a huge cut in the search space, because all supersets of infrequent itemsets are pruned from the search space as well. In other words, monotonicity-based data-reduction of transactions strengthens the antimonotonicity-based pruning of the search space.

This is not the whole story, in fact, infrequent singleton items can not only be removed from the computation: for the same antimonotonicity property they can be deleted also from all transactions in the input database. This antimonotonicity-based data-reduction has been named *α-reduction*. Removing items from transactions has got another positive effect: reducing the size of a transaction which satisfies a monotone constraint can make the transaction violate the monotone constraint. Consider for instance the monotone constraint based on the minimum sum of prices: a transaction which satisfies the constraint having a total sum of prices greater than the given threshold, might end up not satisfying the constraint anymore after its infrequent items are removed, since its total sum of prices might go below the threshold. Therefore a growing number of transactions which do not satisfy the monotone constraint can be found. Obviously, we are inside a loop where two different kinds of data-reduction (α and μ) cooperate 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).

The situation is as follows. We have two data reduction techniques, the first one (*μ-reduction*) based on monotonicity, the second one (*α-reduction*) based on antimonotonicity. The

Procedure: $\mathbf{ExAnte}(TDB, \mathcal{C}_M, min\_supp)$:
where TDB is the transaction database in input, $\mathcal{C}_M$ is a monotone constraint, min_supp is the minimum frequency threshold, I represents the set of frequent singleton items, $Items$ represents the whole alphabet of items.

```
1)  I = ∅;
2)  forall transactions t in TDB do
3)      if C_M(t) then forall items i in t do
4)          i.count++;
5)          if i.count == min_supp then I = I ∪ {i};
6)  old_number_interesting_items = |Items|;
7)  while |I| < old_number_interesting_items do
8)      α-reduce[TDB];
9)      μ-reduce[TDB];
10)     old_number_interesting_items = |I|;
11)     I = ∅;
12)     forall transactions t in TDB do
13)         forall items i in t do
14)             i.count++;
15)             if i.count == min_supp then I = I ∪ {i};
16) end while
```

Fig. 1. The ExAnte algorithm pseudo-code.

item	price	Supports 1_{st}	2_{nd}	3_{rd}
a	5	3	†	†
b	8	7	4	4
c	14	5	5	4
d	30	7	5	4
e	20	3	†	†
f	15	3	†	†
g	6	5	3	†
h	12	2	†	†

(A)

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

(B) (C)

tID	Itemset	Total price
1	b,c,d,g	58
2	b,d	38
3	b,c,d,g	58
5	c,d,g	50
6	b,c,d	52
7	b,d,g	44
8	b,c,d	52
9	b,g	14

(D)

tID	Itemset	Total price
1	b,c,d,g	58
3	b,c,d,g	58
5	c,d,g	50
6	b,c,d	52
8	b,c,d	52

(E)

tID	Itemset	Total price
1	b,c,d	52
3	b,c,d	52
6	b,c,d	52
8	b,c,d	52

(F)

TABLE II

EXAMPLE: THE PRICE TABLE (A), ITEMS AND THEIR SUPPORTS ITERATION BY ITERATION (B), THE INITIAL TRANSACTION DATABASE (C), TWO INTERMEDIATE STATUS OF THE DATABASE (D) AND (E), AND THE FINAL PREPROCESSED DATABASE (F).

first one which deletes transactions from the input database, the second one which deletes items from the transactions. The good news is that they strengthen each other, and both give more power to the antimonotonicity-based pruning of the search space.

In Figure we have named this fruitful collaboration *"the virtuous cycle"*.

We are clearly inside a loop where two different kinds of data-reductions cooperates to reduce the search space and the input database, strengthening each other step by step until no more pruning is possible (a fix-point has been reached). This is the key idea of the ExAnte preprocessing technique, whose pseudo-code is reported in Figure 1.

THE EXANTE ALGORITHM

ExAnte starts the first iteration as any frequent pattern 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 $\mathcal{C}_M$ (the monotone constraint) are considered. The other transactions are signed to be deleted from the database (μ-reduction). Doing so we reduce the number of interesting singleton items. 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 constraint. Therefore we have another opportunity of μ-reducing the database. But μ-reducing the database 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 ExAnte preprocessing returns a much smaller database (often one order of magnitude) containing only the necessary data needed to compute all the interesting patterns which satisfy all given constraints. No information is lost in this reduction: all the interesting itemsets are still in the reduced database and with the initial frequency.

After the ExAnte preprocessing any frequent pattern mining algorithm can be run (for instance Apriori [2] or FP-growth [7]) on the reduced dataset produced by ExAnte, obtaining the whole set of interesting patterns with a much smaller (in time and memory) computation. In other words, the ExAnte preprocessing simply reduces the computational problem dimensions.

Let us go back to the initial question: *"how to exploit monotone constraints in frequent pattern mining?"*. We answer that there is no tradeoff between antimonotonicity and monotonicity. Actually exists a strong synergy between these two opposite components, and this synergy lies in data-reduction.

The next examples will clarify this very simple, yet very effective idea.

PREPROCESSING EXAMPLES

Suppose we have the transaction database in Table II (C) and the price database in Table II (A). 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 database: this means removing a, e, f and h from all transactions obtaining reduced database in Table II (D). After the α-reduction we have more opportunities to μ-reduce the database. 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. We obtain database in Table II (E). At this point ExAnte counts once again the support of alive items within the reduced database. The item g which initially has got a support of 5 now has become infrequent (see Table II (B) for items support iteration by iteration). We can α-reduce again the database, and then μ-reduce it. After the two reductions transaction number 5 does not satisfy anymore the monotone constraint and it is pruned away (see database in Table II (F)). ExAnte counts again the support of items within

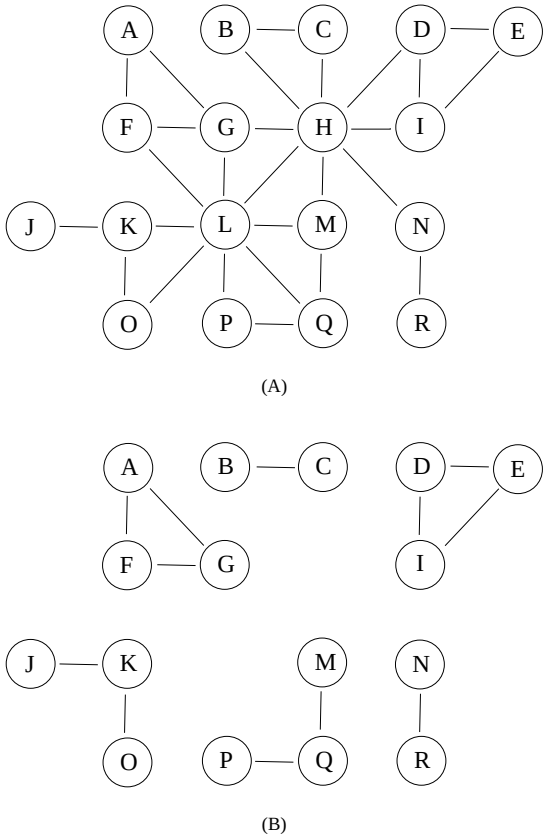


Fig. 2. Example of ExAnte data-reduction for the mining of frequent connected subgraphs.

the reduced database but no more items are found to have turned infrequent. The fix-point has been reached at the third iteration: the database 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). The final database contains only the unique solution to problem which is the 3-itemset $\{b, c, d\}$ with support 4 and sum of prices 52. Note that on this toy-example we even do not need to run a mining algorithm!

ExAnte data-reduction methodology works as well on other kinds of pattern. Consider for instance the problem of mining *frequent connected subgraphs* from a database of graphs [8], and suppose that the user considers interesting only subgraphs containing at least 4 vertices. This additional constraint is clearly monotone. Suppose that the graph in Figure 2 (A) is one of the graphs in the input database (that is the corresponding of a transaction in the frequent itemsets problem). This graph contains 18 vertices and its well far from not satisfying the monotone constraints. Suppose now that vertices H and L are found infrequent. We can α-reduce this graphs by pruning the two infrequent vertices. After this α-reduction we obtain the graph in Figure 2 (B). The graph now contains 16 nodes but there is no connected substructure of at least 4 vertices: this graphs will never participate to the support count of any interesting (solution) subgraph. Therefore we have discovered that our graph, even if it was quite large, does not contain any interesting pattern and can be deleted (μ-reduced) from the input database.

In this last example we had a stronger synergy between antimonotonicity and monotonicity, due to the requirement of being a *connected* subgraph.

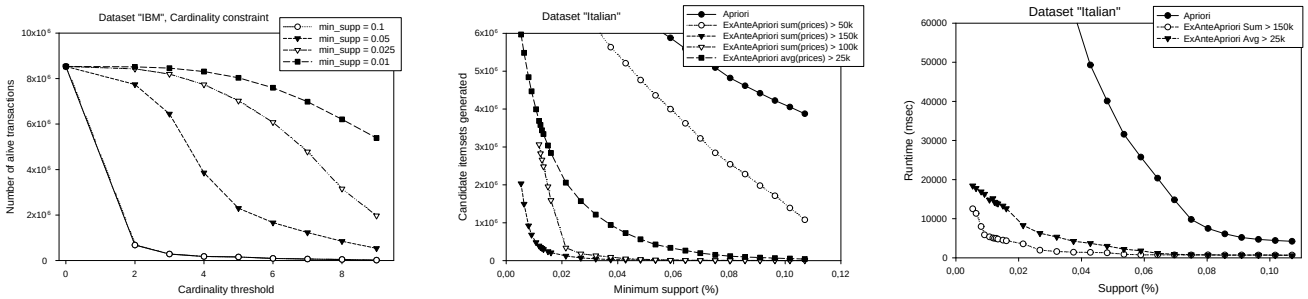
Data-Reduction Rate, Search Space Pruning and Efficiency Gain

ExAnte preprocessing reduces dramatically the input data (see Figure 3 (A)). This data reduction, in turn, induces a strong reduction of the search space (see Figure 3 (B)), thus supporting dramatic performance improvements in subsequent mining (see Figure 3 (C)).

In Figure 3 (A) we show the reduction of the number of transactions in a transactional database, thanks to the frequency antimonotone constraint conjoined with a minimum cardinality monotone constraint. On the Y axis we have the number of transactions alive in the database, and on the X axis we have different cardinality thresholds. We show the data-reduction rate for 4 different frequency thresholds. 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 μ*-reduction*. Already for a low support threshold as 0.1% with a cardinality constraint equals to 2 the number of transactions decreases dramatically. Obviously, the data-reduction becomes larger as the cardinality and frequency constraints become more selective.

The benefit of ExAnte preprocessing is not only in terms of input data-reduction: such data reduction in turn induces a strong pruning of the search space. In fact, as stated in the *virtuous cycle*, having a smaller number of transactions means to have a smaller number of frequent singleton items. But even a small reduction in the number of relevant singleton items 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 Apriori as possible solutions. In Figure 3 (B) is reported a comparison of the number of candidate itemsets generated by Apriori and by *ExAnteApriori* (ExAnte pre-processing followed by Apriori) with various constraints. The dramatic search space reduction is evident, and it will be confirmed by run-time comparisons. Note that in Figure 3 (B) we use prices in obsolete Italian Lire ($2000\ \ Lira \cong 1\ \ Euro$). Moreover in the same figure we have conjoined the frequency constraint even with a constraint ($avg(X.prices) \geq n$) which neither monotone nor antimonotone, and for this reason is harder to exploit in the frequent pattern computation. This kind of constraints is named in literature *convertible* [12]. In our experiments we have shown that even this kind of hard constraints can benefit form ExAnte preprocessing by inducing a weaker, but monotone, constraint from them (in this case $max(X.prices) \geq n$).

In Figure 3 (C) we report run-time comparisons. Note that the Apriori computation which benefits from the ExAnte preprocessing, thanks to the synergy between antimonotonicity and monotonicity, exhibits very good performance also when one of the two components is not very selective. For instance in the figure, when the support threshold is very low, while Apriori without preprocessing boosts due to lack of selectivity



(A) (B) (C)

Fig. 3. Data-reduction rate (A), search space pruning (B), run-time comparisons (C).

of frequency, *ExAnteApriori* (ExAnte pre-processing followed by Apriori) can keep the computation affordable thanks to the selectivity of the monotone constraint. This feature makes ExAnte useful to discover particular patterns which appear only at very low frequency levels (weak antimonotone pruning) for which the computation is unfeasible for traditional algorithms. For instance:

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

Further Exploiting the ExAnte Property

The first proposal of ExAnte was a preprocessing algorithm. However, the basic ExAnte property is a very general idea that can be exploited in many way.

Trying to generalize the technique from preprocessing to real mining, we have introduced a constrained frequent patterns mining algorithm, named *ExAMiner* [3]. ExAMiner exploits the basic idea of ExAnte generalizing it at all levels of an Apriori-like breadth-first computation. In this way, the antimonotonicity-monotonicity synergy is used at each iteration of the algorithm, resulting in significant performance improvements.

The main drawback of this proposal is that the benefit obtained from the data reduction is not always worth the cost of iteratively rewriting the reduced dataset to disk. A solution to that could be to store the database into main memory, where the reduction techniques can be applied much more efficiently. But in that case, it doesn't make much sense anymore to apply the breadth-first approach of Apriori. In [5] we have explored the opportunity of a depth-first approach. Currently, the FP-growth [7] algorithm is the best known algorithm that perform a depth-first search through the search space of all itemsets, by recursively *projecting* the database into smaller databases in which the itemsets sharing a common prefix can be found. The FP-growth algorithm stores the actual transactions from the database in a trie structure and every item has a linked list going through all transactions that contain that item. This data structure is denoted by *FP-tree* (Frequent-Pattern tree). FP-growth has been shown to outperform breadth-first algorithms given that the initial database, and all intermediate projected databases, fit into main memory. If this requirement cannot be met, this approach can simply not be applied anymore. Fortunately, the ExAnte property takes care of exactly that. Thanks to the recursive projecting approach of FP-growth, the ExAnte data-reduction is pervasive all over the computation. All the FP-trees built recursively during the FP-growth computation can be pruned extensively by using the ExAnte property, obtaining a computation with a smaller number of smaller trees. These tiny FP-trees, obtained by growing and pruning, are designated *FP-bonsai*.

The resulting method overcomes on one hand the main drawback of FP-growth, which is its memory requirements, and on the other hand, the main drawback of ExAMiner which is the I/O costs. Experimental results confirm that the proposed method outperforms the state-of-the-art algorithms. Our contribution is particularly important in given that the FP-growth approach is a general methodology which has been successfully instantiated to the mining of several kinds of frequent patterns including closed itemsets, maximal itemsets, and sequential patterns.

Conclusions

Preprocessing should take *size-large knowledge-sparse* data and give back *size-small knowledge-dense* data, possibly without losing any interesting patterns. Following this vision we have introduced ExAnte, a very simple albeit general and effective idea in preprocessing input data for mining frequent patterns of any kind: sets, sequences, trees or graphs. With the introduction of ExAnte we have shown that there is no tradeoff between antimonotonicity and monotonicity, belying a prejudice established in the research community, instead there's a strong synergy between the two opposite components.

ExAnte exploits such synergy in order to reduce dramatically the data in analysis to that containing the patterns of interest. This data reduction, in turn, induces a strong reduction of the search space of candidate patterns, thus supporting dramatic performance improvements in subsequent mining and, sometimes, making feasible otherwise untractable mining tasks. We have proved experimentally the effectiveness of our method, using different constraints on various datasets.

This data-reduction technique is one of the basic stones of a data mining system supporting the vision of a declarative mining paradigm, where is the user that specify what is interesting for the application.

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*, 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] F. Bonchi, F. Giannotti, A. Mazzanti, and D. Pedreschi. ExAMiner: Optimized level-wise frequent pattern mining with monotone constraints. In *Proceedings of the Third IEEE International Conference on Data Mining (ICDM'03)*, 2003.

[4] F. Bonchi, F. Giannotti, A. Mazzanti, and D. Pedreschi. ExAnte: Anticipated data reduction in constrained pattern mining. In *Proceedings of the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'03)*, 2003.

[5] F. Bonchi and B. Goethals. FP-Bonsai: the art of growing and pruning small FP-trees. In *The Eighth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'04)*, 2004.

[6] 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)*, 2000.

[7] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, 2000.

[8] A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *In Proceedings of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'00)*, 2000.

[9] 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.

[10] W. Li, J. Han, and J. Pei. CMAR: Accurate and efficient classification based on multiple class-association rules. In *In Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM'01)*, 2001.

[11] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *4th Int. Conf. Knowledge Discovery and Data Mining (KDD'98)*, 1998.

[12] J. Pei and J. Han. Can we push more constraints into frequent pattern mining? In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'00)*, 2000.

[13] J. Pei, X. Zhang, M. Cho, H. Wang, and P. Yu. Maple: A fast algorithm for maximal pattern-based clustering. In *Proceedings of the Third IEEE International Conference on Data Mining (ICDM'03)*, 2003.

[14] M. L. Yiu and N. Mamoulis. Frequent-pattern based iterative projected clustering. In *Proceedings of the Third IEEE International Conference on Data Mining (ICDM'03)*, 2003.

Dino Pedreschi Dino Pedreschi is a full professor at the Computer Science Department of the University of Pisa. He has been a visiting scientist and professor at the University of Texas at Austin (1989/90), at CWI Amsterdam (1993) and at UCLA (1995). His current research interests are in logic in databases, and particularly in data analysis, in deductive databases, in the integration of data mining and database querying, in spatio-temporal reasoning, and in formal methods for deductive computing. He has taught classes on programming languages and databases in universities in Italy and abroad, and is collaborating with F. Giannotti in course on data mining at the faculty of Economics at the University of Pisa. He participated in the scientific committee of various conferences in the area of Logic Programming and Databases, including the LID'96 Workshop on Logic in Databases, where he was program co-chair, and the LPNMR'99 Workshop on Logic Programming and Non Monotonic Reasoning, 1999. He is one of the program chairs for ECML-PKDD '04 conference. Contact him at pedre@di.unipi.it

Francesco Bonchi just received his Ph.D. in computer science from University of Pisa, with the thesis "Frequent Pattern Queries: Language and Optimizations". Actually he is a post-doc at Institute of Information Science and Technologies (ISTI) of the Italian National Research Council in Pisa where he is a member of the Knowledge Discovery and Delivery Laboratory. He has been a visiting fellow at the Kanwal Rekhi School of Information Technology, Indian Institute of Technology, Bombay (2000, 2001). His current research interests are data mining query language and optimization, frequent pattern mining, privacy-preserving data mining, web mining. He is one of the teachers for a course on data mining held at the faculty of Economics at the University of Pisa. He served as a referee at various national and international conferences on databases, data mining, logic programming and artificial intelligence. He is in the organizing committee for the 15th European Conference on Machine Learning (ECML) and the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD) will be co-located in Pisa, Italy, September 20-24, 2004. Contact him at bonchi@di.unipi.it; http://www-kdd.isti.cnr.it/ bonchi/

Fosca Giannotti is a senior researcher at Institute of Information Science and Technologies (ISTI) of the Italian National Research Council in Pisa where she leads the Knowledge Discovery and Delivery Laboratory. From 1982 to 1985 she was a research assistant, Dip. Informatica, Univ. Pisa. From 1985 to 1989 she was a Senior Researcher at R&D Lab. of Sipe Optimization, Pisa and at R&D Lab. of Systems and Management, Pisa. In 1989/90 she was a visiting researcher of MCC, Austin, Texas, USA, involved in the LDL (Logic Database Language) project. Her current research interests include data mining query languages, knowledge discovery support environment, web-mining, spatio-temporal reasoning, and spatio-temporal data mining. She has taught classes on databases and data mining at universities in Italy and abroad, and has coordinated a master course on data mining in collaboration with the faculty of Economics at the University of Pisa. She served in the organization and in the scientific committee of various conferences in the area of logic programming, databases and data mining. She is one of the program chairs for the ECML-PKDD '04 conference. Contact her at fosca.giannotti@isti.cnr.it

Alessio Mazzanti just received a Laurea degree from University of Pisa, with a thesis on constrained frequent pattern mining. He is a software architect at LIST S.p.A., Pisa. Contact him at mazzanta@cli.di.unipi.it; http://www.cli.di.unipi.it/ mazzanta/