Mining Free Itemsets under Constraints

Jean-François Boulicaut   Baptiste Jeudy
Institut National des Sciences Appliquées de Lyon
Laboratoire d'Ingénierie des Systèmes d'Information
Bâtiment 501
F-69621 Villeurbanne cedex, France
{Jean-Francois.Boulicaut,Baptiste.Jeudy}@lisi.insa-lyon.fr

Abstract

Computing frequent itemsets and their frequencies from large boolean matrices (e.g., to derive association rules) has been one of the hot topics in data mining. Levelwise algorithms (e.g., the APRIORI *algorithm) have been proved effective for frequent itemset mining from sparse data. However, in many practical applications, the computation turns to be intractable for the user-given frequency threshold and the lack of focus leads to huge collections of frequent itemsets. The last three years, two promising issues have been investigated: the use of user defined constraints and closed sets mining. To the best of our knowledge, combining these two frameworks has not been studied yet. In this paper, we show that the benefit of these two approaches can be combined into levelwise algorithms. An experimental validation related to the discovery of association rules with negations is reported.*

1. Introduction

One of the obvious hot topics of data mining research in the past years has been frequent set discovery from large boolean matrices (millions of rows and hundreds of columns). It concerns the discovery of sets of columns that are true within a same row often enough. The user defines the desired frequency threshold and when every frequent itemset has to be found with its frequency (for instance, when association rules [1] are to be derived), it gives rise to challenging algorithmic issues due to the exponential size of the search space.

Levelwise algorithms, e.g., the well-know APRIORI algorithm [2], have been proved effective for frequent itemset mining when the matrix is sparse and the data is lowly correlated. A prototypical application domain where it works is the popular basket analysis problem. However, in most of the other applications we know, the extraction is not always tractable for the user-given frequency thresholds. This happens when the data is dense and/or highly correlated, i.e., when the number of frequent itemsets explodes. Furthermore, even if it is tractable, the size of the output can be huge and is often larger than the size of the original data. The lack of focus leads to huge collections of frequent itemsets from which too many uninteresting patterns or rules will be derived.

During the last three years, two promising issues have been investigated to tackle these problems.

First, one can assume that only a subset of the collection of frequent itemsets is interesting: it leads to *constraint-based extraction of the frequent itemsets* [18, 13, 10, 7]. These studies have considered various kinds of constraints, including "syntactic" constraints (e.g., an item must not appear in the itemsets) and constraints related to the so-called objective measures of itemset interestingness (e.g., the itemsets must be frequent). Using constraints enables to decrease the size of the output while improving user guidance. The problem is to "push" efficiently the constraint checking step during itemset extraction, i.e., not to apply a simple "generate and test" strategy. Nice results have been discovered concerning the so-called anti-monotone, succinct and monotone constraints [13, 7], i.e., a wide range of constraints. This framework has been also studied for other kinds of properties like rules [10] or correlations [9].

Another promising approach concerns the *condensed representation of frequent itemsets* [11]. The intuition is that instead of mining all the frequent patterns, one can extract a particular subset of the frequent pattern collection such that it is possible to regenerate from it the whole collection. Ideally, this subset is much smaller than the original collection and can be extracted more efficiently, while allowing a fast regeneration of the whole collection of frequent patterns. Several researchers have investigated the use of closed frequent sets as a valuable condensed representation [15, 4, 6, 17, 19].

To the best of our knowledge, combining these two frameworks has not been studied yet.

In this paper, we show that the benefit of these two approaches can be combined into levelwise algorithms. Doing so, new mining tasks can be considered like frequent itemset mining for low frequency thresholds or the discovery of frequent generalized itemsets (sets that combine positive and negative items). An experimental validation related to the discovery of association rules with negations is reported.

In Section 2, we recall the APRIORI algorithm and outline the effective processing for anti-monotone constraints. In Section 3, we discuss the use of monotone constraints in order to get effective levelwise algorithms for a rather wide class of constraints. In Section 4, we revisit the CLOSE algorithm [15] that computes closed frequent sets and we discuss its extension towards the constraint-based discovery of itemsets. Section 5 points out a practical validation of this combined framework and Section 6 is a short conclusion.

2. Understanding the effective use of constraints in APRIORI-like algorithms

2.1. Problem settings and notations

We consider transactional databases. Given a finite set `Items` of symbols (denoted by capital letters: `Items`$= \{\mathtt{A}, \mathtt{B}, \mathtt{C}, \ldots\}$) a *transaction* t is a subset of `Items`. A *transactional database* T is a finite and non empty multiset $T = \{t_1, t_2, \ldots, t_n\}$ of transactions. An *itemset* is a subset of `Items` and a k-itemset is an itemset of size k. A transaction t *supports* an itemset S iff $S \subseteq t$. The *support* (denoted $Support(S)$) of an itemset S is the multiset of all transactions of T that support S (e.g. $Support(\emptyset) = T$). The *frequency* of an itemset S is defined by $\mathcal{F}(S) = |Support(S)|/|Support(\emptyset)|$ where $|.|$ denote the cardinality of the multiset (each transaction is counted with its multiplicity). An itemset S is *γ-frequent* in T if $\mathcal{F}(S) \geq \gamma$. Figure 1 provides an example of a transactional database and the supports and the frequencies of some itemsets. Notice that we often use a string notations for sets, e.g., AB for $\{\mathtt{A}, \mathtt{B}\}$.

$$T = \begin{array}{c|l} t_1 & \mathtt{ABCD} \\ t_2 & \mathtt{AC} \\ t_3 & \mathtt{AC} \\ t_4 & \mathtt{ABCD} \\ t_5 & \mathtt{BC} \\ t_6 & \mathtt{ABC} \end{array}$$

Itemset	Support	Frequency
A	$\{t_1, t_2, t_3, t_4, t_6\}$	0.83
B	$\{t_1, t_4, t_5, t_6\}$	0.67
AB	$\{t_1, t_4, t_6\}$	0.5
AC	$\{t_1, t_2, t_3, t_4, t_6\}$	0.83
CD	$\{t_1, t_4\}$	0.33
ACD	$\{t_1, t_4\}$	0.33

Figure 1. Supports and frequencies of some itemsets in a transactional database T.

```
APRIORI algorithm
1.  $C_1^g$ := Items$_1$; $\mathcal{L}_0 = \{\emptyset\}$
2.  $k := 1$
3.  while $C_k^g \neq \emptyset$ do
4.      Phase 1 - candidate safe pruning
        $C_k$ := safe-pruning-on($C_k^g, \mathcal{L}_{k-1}$)
5.      Phase 2 - frequency constraint - it needs a data scan
        $\mathcal{L}_k := SAT_{\mathcal{C}_{freq}}(C_k)$
6.      Phase 3 - candidate generation for level k+1
        $C_{k+1}^g$ := generate$_{apriori}(\mathcal{L}_k)$
7.      $k := k + 1$
    od
8.  output $\bigcup_{i=0}^{k-1} \mathcal{L}_i$
```

Definition 1 (constraint) *If $\mathcal{T}$ denotes the set of all transactional databases and $2^{\mathtt{Items}}$ the set of all itemsets, a constraint $\mathcal{C}$ is a predicate over $2^{\mathtt{Items}} \times \mathcal{T}$. We say that an itemset $S \in 2^{\mathtt{Items}}$* satisfies *a constraint $\mathcal{C}$ in the database $T \in \mathcal{T}$ iff $\mathcal{C}(S, T) = true$. When it is clear from the context, we write $\mathcal{C}(S)$. Given a subset I of `Items`, we define $SAT_{\mathcal{C}}(I) = \{S \in I,\ S$ satisfies $\mathcal{C}\}$. $SAT_{\mathcal{C}}$ denotes $SAT_{\mathcal{C}}(2^{\mathtt{Items}})$.*

Let $\mathcal{C}_{freq}(S) \equiv \mathcal{F}(S) \geq \gamma$ be the constraint that is true iff S is γ-frequent in T.

Example 1 *Consider the dataset of Figure 1 where `Items`$= \{\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}\}$. If $\mathcal{C}_{freq}$ specifies that an itemset must be 0.6-frequent, then $SAT_{\mathcal{C}_{freq}} = \{\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{AC}, \mathtt{BC}\}$. Assume that $\mathcal{C}_{size}(S) \equiv |S| \leq 2$ and $\mathcal{C}_{miss}(S) \equiv \mathtt{B} \notin S$, then $SAT_{\mathcal{C}_{size} \wedge \mathcal{C}_{miss}} = \{\mathtt{A}, \mathtt{C}, \mathtt{D}, \mathtt{AC}, \mathtt{AD}, \mathtt{CD}\}$ while $SAT_{\mathcal{C}_{freq} \wedge \mathcal{C}_{size} \wedge \mathcal{C}_{miss}} = \{\mathtt{A}, \mathtt{C}, \mathtt{AC}\}$.*

Definition 2 (constrained itemset mining task) *Given a transactional database T and a constraint $\mathcal{C}$, the constrained itemset mining task is the computation of the collection of the itemsets that verify $\mathcal{C}$ (i.e., $SAT_{\mathcal{C}}$) together with their frequencies. It provides $\mathcal{R}_{\mathcal{C}} = \{(S, \mathcal{F}(S)),\ S \in SAT_{\mathcal{C}}\}$.*

2.2. Sketching the APRIORI algorithm

We consider an abstract definition of the APRIORI algorithm [2] to support our discussion on the effective use of constraints. This algorithm performs the constrained itemset mining task when $\mathcal{C}$ is $\mathcal{C}_{freq}$.

In this algorithm, and in the following ones, the frequency of the itemsets are not explicit for the sake of clarity (e.g., Line 5 of the algorithm should be $\mathcal{L}_k := \{(S, \mathcal{F}(S)),\ S \in SAT_{\mathcal{C}_{freq}}(C_k)\}$ since APRIORI outputs the frequency of each frequent itemset).

APRIORI is a levelwise exploration of the lattice of itemsets (w.r.t. set inclusion). During the first pass (when $k = 1$), it computes frequent 1-itemsets and then it generates candidate 2-itemsets from frequent 1-itemsets. In the second pass ($k = 2$), it prunes some candidate 2-itemsets (those that contain an infrequent subset), it computes their frequencies and it generates candidate 3-itemsets from frequent 2-itemsets, ...

C_k^g denotes the k-itemsets that can be frequent. During Phase 1, some of these k-itemsets are pruned. safe-pruning-on($C_k^g, \mathcal{L}_{k-1}$) eliminates the candidates for which a subset of length k is not frequent. This can be justified by the so-called APRIORI trick: if S is not frequent, every superset of S is not frequent and S can be safely pruned.

During Phase 2, a database scan is performed to compute the frequency of the candidate itemsets. The frequent ones are stored in $\mathcal{L}_k$ together with their frequencies.

In Phase 3, frequent k-itemsets are used to compute candidate $k+1$-itemsets. $\texttt{generate}_{apriori}(\mathcal{L}_k)$ provides the candidates by fusion of two elements from $\mathcal{L}_k$ that share the same $k-1$ first items: $\texttt{generate}_{apriori}(\mathcal{L}_k)$=$\{A \cup B$, where $A, B \in \mathcal{L}_k$, A and B share the $k-1$ first items (in lexicographic order)$\}$. This generation procedure is the key of the efficiency of the algorithm: it ensures that large portions of the itemset lattice are pruned and that no frequent itemset is missed.

Example 2 *Let us focus on an extract from the execution of* APRIORI *on the data of Figure 1 with a frequency threshold $\gamma = 0.5$. At Iteration 2 ($k = 2$) Phase 2, one can verify that $\mathcal{L}_2 = \{\texttt{AB}, \texttt{AC}, \texttt{AD}, \texttt{BC}, \texttt{CD}\}$. Then Phase 3 provides the collection of candidates $C_3^g = \{\texttt{ABC}, \texttt{ABD}, \texttt{ACD}\}$ (BCD is not generated since $\texttt{BD} \notin \mathcal{L}_2$ and therefore BCD cannot be frequent). At Iteration 3 Phase 1 the pruning step provides $C_3 = \{\texttt{ABC}, \texttt{ACD}\}$ since $\texttt{BD} \notin \mathcal{L}_2$ (and therefore ABD cannot be frequent).*

It can be proved by induction on k that APRIORI is correct and complete, i.e., $\cup_{i=0}^{k-1} \mathcal{L}_i = SAT_{\mathcal{C}_{freq}}$.

2.3. Effective use of anti-monotone constraints

It is well-known that the completeness of APRIORI (i.e., it does not prune any frequent itemset) relies on the anti-monotonicity of $\mathcal{C}_{freq}$.

Definition 3 (Anti-monotonicity) *An* anti-monotone *constraint is a constraint $\mathcal{C}$ such that for all itemsets S, S': ($S' \subseteq S \wedge S$ satisfies $\mathcal{C}$) $\Rightarrow S'$ satisfies $\mathcal{C}$.*

Notice that a disjunction or a conjunction of anti-monotone constraints is an anti-monotone constraint.

Example 3 *$\mathcal{C}_{freq}$, $\mathcal{C}(S) \equiv \texttt{A} \notin S$, $S \subseteq \{\texttt{A}, \texttt{B}, \texttt{C}\}$ and $S \cap \{\texttt{A}, \texttt{B}, \texttt{C}\} = \emptyset$ are examples of anti-monotone constraints. Many other anti-monotone constraints are presented in [13].*

Let $\mathcal{C}_{am}$ be an anti-monotone constraint eventually including $\mathcal{C}_{freq}$ (i.e., $\mathcal{C}_{am}$ could contain $Cfreq$ or not): if S does not satisfy $\mathcal{C}_{am}$, every superset of S does not satisfy $\mathcal{C}_{am}$. Therefore, if Step 5 of the APRIORI algorithm is replaced by $\mathcal{L}_k := SAT_{\mathcal{C}_{am}}(C_k)$, it is still correct and complete. It means that APRIORI can be used to mine constrained itemsets when the given constraint is anti-monotone.

3. Testing Monotone Constraints

If the effective use of anti-monotone constraints is easy to understand, it is far more complex in the general case. In other terms, given an arbitrary constraint $\mathcal{C}$, it is not possible to use it in APRIORI by simply replacing Step 5 with $\mathcal{L}_k := SAT_{\mathcal{C}}(C_k)$. Doing this leads to the loss of the completeness of APRIORI. Indeed, there is two problems: the generation step and the pruning step. The generation step must be complete, i.e., it must not miss any itemset satisfying $\mathcal{C}$, and also the pruning step (Phase 1) must be correct, i.e., it must not prune an itemset that verify the constraint.

Example 4 *Assume the constraint is $\mathcal{C}(S) \equiv \texttt{C} \in S$. The itemset ABC should be generated by $\texttt{generate}_{apriori}$ from AB and AC but since $\mathcal{C}(\texttt{AB}) = false$, ABC is not generated whereas $\mathcal{C}(\texttt{ABC}) = true$.*

If the constraint is $\mathcal{C}(S) \equiv \texttt{A} \in S$. The itemset ABC is then correctly generated by $\texttt{generate}_{apriori}$ from AB and AC but since $\mathcal{C}(\texttt{BC}) = false$, ABC is incorrectly pruned whereas $\mathcal{C}(\texttt{ABC}) = true$.

To overcome these problems, we propose a generation procedure and a pruning procedure which allow to push conjunctions of anti-monotone and monotone constraints, i.e., when $\mathcal{C}$ can be written as $\mathcal{C}_{am} \wedge \mathcal{C}_m$. A *monotone constraint* $\mathcal{C}_m$ is the negation of an anti-monotone constraint and it motivates the notation $\neg\mathcal{C}'_{am}$ for $\mathcal{C}_m$. The main property of a monotone constraint is: $S \subseteq \texttt{Items}$, $\mathcal{C}_m(S)$ is true $\Rightarrow \forall S' \supset S, \mathcal{C}_m(S')$ is true.

Example 5 *Continuing our running example, $\{\texttt{A}, \texttt{B}, \texttt{C}, \texttt{D}\} \subset \texttt{S}$ and $S \cap \{\texttt{A}, \texttt{B}, \texttt{C}\} \neq \emptyset$ are monotone constraints. Assuming that items in S are correlated is monotone too [9].*

Generation procedure The next theorem gives a new complete generation procedure when the constraint is the conjunction of a monotone and an anti-monotone constraint. We use the concept of negative border [12]. If $\mathcal{C}_{am}$

denotes an anti-monotone constraint, $\mathcal{B}d^-_{\mathcal{C}_{am}}$ is the collection of the minimal (w.r.t. the set inclusion) itemsets that do not satisfy $\mathcal{C}_{am}$.

Let us now consider two generating procedures, $\texttt{generate}_1(\mathcal{L}_k) = \{A \cup B$, where $A \in \mathcal{L}_k$ and B is a 1-itemset$\}$ and $\texttt{generate}_2(\mathcal{L}_k) = \{A \cup B$, where $A, B \in \mathcal{L}_k\}$. Notice that a naive algorithm that computes $\texttt{generate}_2$ will produce many duplicates (see [8]).

Our new generation procedure is denoted $\texttt{generate}_m$ and is defined by the next theorem.

Theorem 1 *Assume* $\mathcal{C} = \mathcal{C}_{am} \wedge \neg\mathcal{C}'_{am}$ *and* $ms = \textit{Max}_{\,S \in \mathcal{B}d^-_{\mathcal{C}'_{am}}} |S|$.
If $\texttt{generate}_m$ *is defined by :*
$\texttt{generate}_m(\mathcal{L}_0) = \mathcal{B}d^-_{\mathcal{C}'_{am}} \cap \texttt{Items}_1$ *and, for* $k \geq 1$,
if $k < ms$, $\texttt{generate}_m(\mathcal{L}_k) = \texttt{generate}_1(\mathcal{L}_k) \cup (\mathcal{B}d^-_{\mathcal{C}'_{am}} \cap \texttt{Items}_{k+1})$*;*
if $k = ms$, $\texttt{generate}_m(\mathcal{L}_k) = \texttt{generate}_1(\mathcal{L}_k)$*;*
if $k > ms$, $\texttt{generate}_m(\mathcal{L}_k) = \texttt{generate}_2(\mathcal{L}_k)$*;*
then this candidate generation procedure is complete and ensures that every candidate itemset verifies $\neg\mathcal{C}'_{am}$.

The fact that every candidate itemset verify $\mathcal{C}_m = \neg\mathcal{C}'_{am}$ makes useless any verification of this constraint after the candidate generation step.

Safe pruning procedure Let us now introduce a correct and complete pruning algorithm.

```
Pruning: algorithm prune_m
    for all $S \in C^g_{k+1}$ and for all $S' \subset S$ such that $|S'| = k$
        do if $S' \notin \mathcal{L}_k$ and $\mathcal{C}_m(S') = true$
            then delete $S$ from $C^g_{k+1}$ od
```

Theorem 2 *The pruning algorithm* $\texttt{prune}_m$ *is correct and complete.*

This algorithm is correct because it does not prune any itemset that verify $\mathcal{C} = \mathcal{C}_{am} \wedge \mathcal{C}_m$. Its completeness means that if an itemset is not pruned then every proper subset of that itemset verify $\mathcal{C}_{am}$. Intuitively, it is not possible to prune more itemsets without affecting the completeness.

Generic algorithm We can now give a generic algorithm for a constraint $\mathcal{C} = \mathcal{C}_{am} \wedge \mathcal{C}_m = \mathcal{C}_{am} \wedge \neg\mathcal{C}'_{am}$ using the structure of APRIORI and our procedures $\texttt{generate}_m$ and $\texttt{pruning}_m$.

```
generic algorithm
1.  $C^g_1$ :=$\mathcal{B}d^-_{\mathcal{C}'_{am}} \cap \texttt{Items}_1$ ; $\mathcal{L}_0 = \{\emptyset\}$
2.  $k$ := 1
3.  while $C^g_k \neq \emptyset$ do
4.      Phase 1 - candidate safe pruning
        $C_k$ := $\texttt{pruning}_\texttt{m}(C^g_k, \mathcal{L}_{k-1})$
5.      Phase 2 anti-monotone constraint checking
        $\mathcal{L}_k$ := $SAT_{\mathcal{C}_{am}}(C_k)$
6.      Phase 3 - candidate generation for level k+1
        $C^g_{k+1}$ := $\texttt{generate}_m(\mathcal{L}_k)$
7.      $k$ := $k + 1$
    od
8.  output $\bigcup_{i=0}^{k-1} \mathcal{L}_i$
```

Note that it is not necessary to check $\mathcal{C}_m$ during Phase 2 since Theorem 1 ensures that every generated itemset verifies it.

Related work Our generic algorithm is inspired by several algorithms [18, 8, 13] and can be considered as a generalization of them. Conjunctions of monotone and anti-monotone constraints encompass every kind of constraints that have been "pushed" inside a levelwise algorithm (another kind of interesting constraint, the convertible constraints [16], can be pushed in depth-first exploration algorithms). The framework of succinct constraints introduced in [13] allows to find an effective generation procedure (i.e., an effective computation of the negative border $\mathcal{B}d^-_{\mathcal{C}'_{am}}$ of Theorem 1).

4. Revisiting the CLOSE Algorithm

It is now interesting to revisit the algorithms CLOSE [15], Charm [19] and MIN-EX [4]. These algorithms compute frequent closed itemsets, i.e., condensed representations of frequent itemsets. They allow tractable frequent itemset extractions from dense and highly-correlated data, i.e., tractable extractions for thresholds on which APRIORI is clearly intractable.

4.1. How CLOSE algorithm has been defined so far

The APRIORI algorithm explores the itemset lattice to find all the frequent itemsets. However, the number of frequent itemsets can be exponential in the size of $\texttt{Items}$. If the size of $\texttt{Items}$ is n, the size of the itemset lattice is 2^n and many of these itemsets can be frequent for the given frequency threshold. This is the case in highly-correlated data like for instance census data.

The CLOSE algorithm (and related algorithms) operates on a different lattice: the *closed itemset* lattice.

Definition 4 (closed itemset lattice) *The closure of an itemset* S *(denoted by* $\texttt{closure}(S)$*) is the maximal (for set inclusion) superset of* S *which has the same support as* S*. A closed itemset is an itemset that is equal to its closure. The set of closed itemset is a lattice called the closed itemset lattice.*

In CLOSE, the exploration of this lattice is done in a levelwise manner like APRIORI. The efficiency of these algorithms comes from the fact that this lattice is generally several order of magnitude smaller than the itemset lattice.

These algorithms output the set of frequent closed itemsets. Frequent closed itemsets are interesting for several reasons:

- they are far less numerous than frequent itemsets (and therefore faster to compute, easier to store and manipulate),
- if necessary, it is possible to generate efficiently all frequent itemsets (and their frequencies) from the closed ones,
- it is possible to derive (non redundant) association rules directly from closed frequent itemsets without generating all frequent non-closed ones (see, e.g., [19, 14]).

4.2. A new constraint

We show how to consider the CLOSE algorithm as an exploration of the classical itemset lattice with a new constraint $\mathcal{C}_{Free}$. Then in Section 4.3, we will be able to use this constraint in our generic algorithm together with other constraints and therefore achieve constrained free-set mining. We first define a constraint $\mathcal{C}'_{Free}$.

Definition 5 (A constraint for CLOSE) $\mathcal{C}'_{Free}(S) \equiv S' \subset S \Rightarrow S \not\subseteq \mathtt{closure}(S')$.

The itemsets which verify this constraint are exactly the *0-free sets* introduced in [6] and it motivates the chosen name of the constraint.

Definition 6 (Free itemsets) *Free itemsets are itemsets that are not included in any closure of their proper sub-set. Equivalently, free itemsets are itemsets that verify* $\mathcal{C}'_{Free}$.

A fundamental property of free itemesets is that no logical rule (i.e., association rule with a confidence of 1) holds between their attributes. In other words, if X is a free itemset, then there does not exist two distinct subset Y and Z of X with $Z \neq \emptyset$ such that the rule $Y \Rightarrow Z$ has a confidence of 1. Also, the frequency of itemsets that are not free can be inferred from the frequency of free itemsets [6].

Example 6 *Let us compute* $\mathtt{closure}(\mathtt{AB})$ *on our running example. Items* A *and* B *are simultaneously in transactions 1, 4 and 6. We notice that item* C *is the only other item that is also present in these three transactions, thus* $\mathtt{closure}(\mathtt{AB}) = \mathtt{ABC}$. *We also have* $\mathtt{closure}(\mathtt{A}) = \mathtt{AC}$ *and* $\mathtt{closure}(\mathtt{B}) = \mathtt{BC}$, *so* $\mathtt{AB} \not\subseteq \mathtt{closure}(\mathtt{A})$ *and* $\mathtt{AB} \not\subseteq \mathtt{closure}(\mathtt{B})$. *Therefore* $\mathcal{C}'_{Free}(\mathtt{AB})$ *is true. If the frequency threshold is* $\gamma = 1/2$, $SAT_{\mathcal{C}_{freq} \wedge \mathcal{C}'_{Free}} = \{\emptyset^{\mathtt{C}}, \mathtt{A}^{\mathtt{C}}, \mathtt{B}^{\mathtt{C}}, \mathtt{D}^{\mathtt{AC}}, \mathtt{AB}^{\mathtt{C}}\}$ *where the notation* $\mathtt{AB}^{\mathtt{C}}$ *means that* $\mathcal{C}'_{Free}(\mathtt{AB})$ *is true and that* $\mathtt{closure}(\mathtt{AB}) = \mathtt{ABC}$.

Notice that when the closure of an itemset X is a proper superset of X, say Y, it means that an association rule $X \Rightarrow Y$ holds with confidence 1. In other terms, it means that the more you have such correlations in your data, the less you have free itemsets and thus the less you have to count for frequencies when looking for frequent itemsets.

Proposition 1 *The* $\mathcal{C}'_{Free}$ *constraint is anti-monotone.*

This constraint is another example of an anti-monotone constraint which needs a database pass to be checked (a database pass is needed to compute the closure of an itemset). Checking this constraint seems expensive if the closure of every subset of S has to be computed. We can use an equivalent constraint $\mathcal{C}_{Free}(S) \equiv (S' \subset S \wedge |S'| = |S| - 1) \Rightarrow S \not\subseteq \mathtt{closure}(S')$. The equivalence means that $\mathcal{C}'_{Free}(S)$ is true iff $\mathcal{C}_{Free}(S)$ is true.

So we only need the closure of every subset of S of size $|S|-1$. We are now able to test the constraint on $S \in \mathcal{L}_{k+1}$: for each $S' \subset S$ such that $|S'| = |S| - 1$ we must know $\mathtt{closure}(S')$. In the CLOSE algorithm, the closure of each candidate itemset of size k and its frequency are computed during the database pass at level k. If the closure of S' is not computed, it means that S' does not verify $\mathcal{C}_{freq} \wedge \mathcal{C}_{Free}$ (i.e., an anti-monotone constraint) and therefore S cannot verify $\mathcal{C}_{freq} \wedge \mathcal{C}_{Free}$. Finally, either the closure of S' is known and we can check if $S \subseteq \mathtt{closure}(S')$ or it is not known and it means that $\mathcal{C}_{freq}(S) \wedge \mathcal{C}_{Free}(S)$ is false. This strategy which uses the anti-monotonicity of $\mathcal{C}'_{Free}$ enables to test the constraint with only a little extra cost during the database pass.

4.3. Incorporating constraints

Now, it seems straightforward to search for itemsets which verify a constraint $\mathcal{C} = \mathcal{C}_{Free} \wedge \mathcal{C}_{am} \wedge \mathcal{C}_m$ using the generic algorithm (since $\mathcal{C}_{Free}$ is anti-monotone). However, there are two problems. First (due to $\mathcal{C}_m$), the closures of some candidates of level k are not computed thus making the $\mathcal{C}_{Free}$ checking impossible at level $k+1$ (it is not possible to check if an itemset of size $k+1$ is included in the closure of one of its proper subset). Second, we loose an important property of CLOSE: $SAT_{\mathcal{C}_{Free} \wedge \mathcal{C}_{am} \wedge \mathcal{C}_m}$ will no longer enables to compute $SAT_{\mathcal{C}_{am} \wedge \mathcal{C}_m}$.

Assume we replace $\mathcal{C}'_{Free}$ with $\mathcal{C}'_{Free \wedge \mathcal{C}_m}(S) \equiv (S' \subset S \wedge \mathcal{C}_m(S')) \Rightarrow S \not\subseteq \mathtt{closure}(S')$ and $\mathcal{C}_{Free}$ with: $\mathcal{C}_{Free \wedge \mathcal{C}_m}(S) \equiv (S' \subset S \wedge |S'| = |S| - 1 \wedge \mathcal{C}_m(S')) \Rightarrow S \not\subseteq \mathtt{closure}(S')$. Then we have the following theorem.

Theorem 3 *The constraints $\mathcal{C}_{Free \wedge \mathcal{C}_m}$ and $\mathcal{C}'_{Free \wedge \mathcal{C}_m}$ are equivalent and anti-monotone. The set $SAT_{\mathcal{C}_{am} \wedge \mathcal{C}_m}$ can be efficiently computed using the same method as in* CLOSE *using $SAT_{\mathcal{C}_{Free \wedge \mathcal{C}_m} \wedge \mathcal{C}_{am} \wedge \mathcal{C}_m}$, i.e., the output of the generic algorithm with the constraint $\mathcal{C} = \mathcal{C}_{Free \wedge \mathcal{C}_m} \wedge \mathcal{C}_{am} \wedge \mathcal{C}_m$.*

This theorem means that we can find free-itemsets that verify conjunctions of anti-monotone and monotone constraints.

4.4. MIN-EX algorithm

The MIN-EX algorithm is an extension of the CLOSE algorithm [4]. The concept of closure is extended, providing new possibilities for pruning. However, we must trade this efficiency improvement against precision: the frequency of the frequent itemsets are only known within and bounded error. If δ is an integer, let $\texttt{closure}_\delta$(S) be the maximal (for the set inclusion) superset Y of S such that for every item $A \in Y - S$, $|\texttt{Support}(S \cup \{\texttt{A}\})|$ is at least $|\texttt{Support}(S)| - \delta$ (with $\delta = 0$, it is the same closure operator than CLOSE i.e., $\texttt{closure}_0 = \texttt{closure}$). Larger values of δ leads to more efficiency improvement and larger errors on the frequencies of itemsets. By replacing this new closure operator in the definition of $\mathcal{C}_{Free}$ we define $\mathcal{C}_{\delta-Free}$ and $\mathcal{C}_{\delta-Free \wedge \mathcal{C}_m}$ and Theorem 3 is true with these new constraints. The sets that fulfill $\mathcal{C}_{\delta-Free}$ are the so-called δ-free sets from [6]. Here again, one can say that the more you have almost logical association rules that hold in your data (rules with confidence close to 1 since only δ exceptions are allowed), the less you have δ-free sets. It has been shown that when the frequency of a frequent itemset is approximated by using the frequency of a δ-free set, the error on frequency can remain very low in practice [6].

5. An experimental validation

We consider an experiment motivated by the search for association rules with negations [5]. Only some results concerning the discovery of generalized sets (from with association rules with negations are derived) are given here.

Notations Let $\texttt{Items}^+ = \{\texttt{A}, \texttt{B}, ...\}$ be a finite set of symbols called the positive items and a set $\texttt{Items}^-$ of same cardinality as $\texttt{Items}^+$ whose elements are denoted $\overline{\texttt{A}}, \overline{\texttt{B}}, \ldots$ and called the negative items. Given a transactional database $\mathcal{T}$ over $\texttt{Items}^+$, let us define a complemented transactional database over $\texttt{Items} = \texttt{Items}^+ \cup \texttt{Items}^-$ as follows: for a given transaction $t \in \mathcal{T}$, we add to t negative items corresponding to positive items not present in t. Generalized itemsets are subsets of $\texttt{Items}$ and can contain positive and negative items.

Constraints We want to extract frequent itemsets ($\mathcal{C}_{freq}$) that do not involve only negative items ($\mathcal{C}_{alpp}$). $\mathcal{C}_{alpp}(S)$ is true when S involves at least p positive items. This is obviously a monotone constraint. First experiments have shown that it was interesting to relax such a monotone constraint (i.e., accepting more sets) in order to give rise to more pruning (see [5] for a complete discussion). Following that guideline, instead of $\mathcal{C}_{alpp}$, we used the constraint $\mathcal{C}_{alppoam1n} = \mathcal{C}_{alpp} \vee \mathcal{C}_{am1n}$

This constraint enforces $\underline{a}$t $\underline{l}$east $\underline{p}$ $\underline{p}$ositive attributes (a monotone constraint) $\underline{o}$r $\underline{a}$t $\underline{m}$ost $\underline{1}$ $\underline{n}$egative attribute (an anti-monotone constraint).

Let us introduce the collection of constraints that have been used.

$\mathcal{C}_{freq}$
$\mathcal{C}_1 = \mathcal{C}_{freq} \wedge \mathcal{C}_{al1poam1n}$
$\mathcal{C}_2 = \mathcal{C}_{freq} \wedge \mathcal{C}_{al2poam1n}$
$\mathcal{C}_3 = \mathcal{C}_{freq} \wedge \mathcal{C}_{al3poam1n}$
$\mathcal{C}_{f1} = \mathcal{C}_{freq} \wedge \mathcal{C}_{Free \wedge \mathcal{C}_m} \wedge \mathcal{C}_{al1poam1n}$
$\mathcal{C}_{f2} = \mathcal{C}_{freq} \wedge \mathcal{C}_{Free \wedge \mathcal{C}_m} \wedge \mathcal{C}_{al2poam1n}$
$\mathcal{C}_{f3} = \mathcal{C}_{freq} \wedge \mathcal{C}_{Free \wedge \mathcal{C}_m} \wedge \mathcal{C}_{al3poam1n}$
$\mathcal{C}_{d1} = \mathcal{C}_{freq} \wedge \mathcal{C}_{\delta-Free \wedge \mathcal{C}_m} \wedge \mathcal{C}_{al1poam1n}$
$\mathcal{C}_{d2} = \mathcal{C}_{freq} \wedge \mathcal{C}_{\delta-Free \wedge \mathcal{C}_m} \wedge \mathcal{C}_{al2poam1n}$
$\mathcal{C}_{d3} = \mathcal{C}_{freq} \wedge \mathcal{C}_{\delta-Free \wedge \mathcal{C}_m} \wedge \mathcal{C}_{al3poam1n}$

With these constraints, we are able to compare different approaches :

- using only the frequency constraint;
- using the frequency constraint and $\mathcal{C}_{alppoam1n}$ constraint with p = 1, 2, 3 ($\mathcal{C}_{f1}$, $\mathcal{C}_{f2}$ and $\mathcal{C}_{f3}$);
- using $\mathcal{C}_{freq}$ and $\mathcal{C}_{alppoam1n}$ in conjunction with $\mathcal{C}_{Free \wedge \mathcal{C}_m}$ or $\mathcal{C}_{\delta-Free \wedge \mathcal{C}_m}$.

Datasets We studied the use of these constraints on two dataset. The first one is a benchmark, the so-called `mushroom` data. This dataset is a binary matrix of 8124 rows. Each row contains 23 discrete attributes. Theses attributes are binarized into exclusive attribute-value pairs. This leads to a binary matrix with 119 columns and 23 "1" per row. When encoding negative items, it leads to a matrix with 238 columns whose each row contains 119 "1".

The second dataset is from the French national institute of statistics (INSEE). In this dataset, each row represents a French town and each column represents a kind of service (e.g., bank, insurance company, etc), a "1" in "bank" column means that there is at least one bank in the town. In this dataset, there are about 37000 rows and 59 columns with an average number of "1" per row of 4. When encoding negative items, it leads to a matrix with 118 columns whose each row contains 59 "1".

The former dataset is quite small but it is known to be tough due to the high correlation between the attributes and its density (for positive attributes). The latter dataset is larger but it is sparse (4 “1” per row on average for positive attributes) and less correlated. These two different datasets let us compare our approach on different types of datasets. Indeed the results show a great difference between these two experiments.

Experiments The experiments were conducted on a 500 MHz Pentium III with 768 MB of memory. The value of the frequency threshold (γ) is changed over experiments in order to observe the trend. Logarithmically scaled axes are used. The value of the δ parameter in the $\mathcal{C}_{\delta-Free}$ constraint is set to 200 for the mushroom database and to 100 for the INSEE database (in this latter database there was only a slight difference in execution time between $\delta = 100$ and $\delta = 200$ but $\delta = 100$ gives more accurate results as explained in Section 4.4).

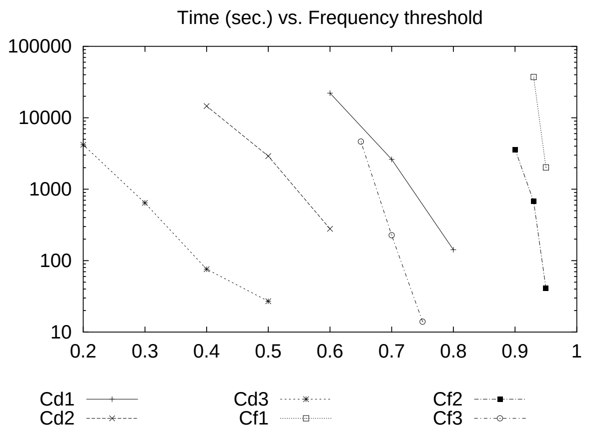

Figure 2. Mining generalized sets from mushroom dataset.

Figure 2 shows the results of the experiments on the mushroom dataset with the constraints $\mathcal{C}_{f1}$, $\mathcal{C}_{f2}$, $\mathcal{C}_{f3}$, $\mathcal{C}_{d1}$, $\mathcal{C}_{d2}$ and $\mathcal{C}_{d3}$. The extractions using the other constraints ($Cfreq$, $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_3$) were intractable even at high frequency threshold (95%) and with the strongest requirement on the number of positive items ($\mathcal{C}_3$). On this dataset the use of $\mathcal{C}_{\delta-Free \wedge \mathcal{C}_m}$ as opposed to $\mathcal{C}_{\delta-Free \wedge \mathcal{C}_m}$ clearly improves the results. With $\mathcal{C}_{f3}$, a frequency threshold of 65% is reached whereas using $\mathcal{C}_{d3}$ allows to reach a frequency of 20% within about the same time. Finally, on the mushroom dataset,

- $\mathcal{C}_{freq}$ combined with the most favorable case of $\mathcal{C}_{alpp}$ still leads to an intractable extraction;
- $\mathcal{C}_{freq}$ combined with only $\mathcal{C}_{\delta-Free}$ or $\mathcal{C}_{Free}$ does not allow mining at low threshold (even with $\mathcal{C}_{al1p}$ i.e., $\mathcal{C}_{d1}$ and $\mathcal{C}_{f1}$, we only reach 60%).

Therefore, to mine at reasonable frequency thresholds, the conjunction of both techniques (using constraints on itemsets with the $\mathcal{C}_{alppoam1n}$ family of constraints and looking for δ-free-sets with $\mathcal{C}_{Free \wedge \mathcal{C}_m}$ or $\mathcal{C}_{\delta-Free \wedge \mathcal{C}_m}$) appears mandatory.

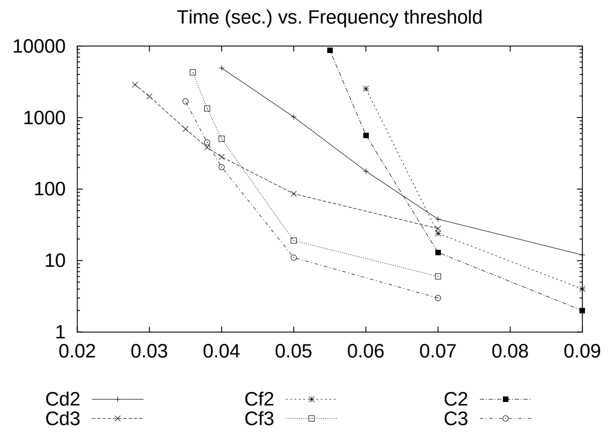

Figure 3. Mining generalized sets from INSEE dataset.

Figure 3 shows the results of the experiments on the INSEE dataset. With the constraints $\mathcal{C}_{freq}$, $\mathcal{C}_1$, $\mathcal{C}_{f1}$ and $\mathcal{C}_{d1}$ it is not possible to reach a frequency threshold less than 34%. For this frequency threshold, there is only one frequent positive attribute. This means that all the itemsets mined at this threshold are composed of the same positive attribute and several negative ones.

On this dataset, we notice that the $\mathcal{C}_{fi}$ family of constraint is surprisingly less efficient than the less constraining $\mathcal{C}_i$ family. We analyzed the output of the algorithm and found that almost no logical rule (association rule with a confidence of 1) holds in this dataset. The optimization of $\mathcal{C}_{fi}$ over $\mathcal{C}_i$ is based on the presence of these rules.

Even if the use of δ-free-sets (with $\mathcal{C}_{d2}$ or $\mathcal{C}_{d3}$) does not allow to mine at significantly lower thresholds, however, it speeds up the extraction by an order of magnitude with respect to $\mathcal{C}_2$ or $\mathcal{C}_3$ at lower frequency thresholds. Finally, with this dataset too the approach turns to be valuable.

6. Conclusion

We study itemset mining under constraints within levelwise algorithms. Several interesting results have been already published the last three years, e.g., about the effective use of anti-monotone constraints or the interest of monotone constraints. The generic algorithm we give in this paper is a simple generalization of several related algorithms and enable to emphasize the potential of optimization when considering conjunctions of anti-monotone and monotone constraints. Furthermore, we provide new results concerning the computation of free sets under constraints. We discussed under which conditions it was possible to extend an algorithm like CLOSE for an effective use of constraints. An experimental validation has confirmed the added-value of this approach.

A recent work uses a different approach and proposes to mine frequent itemsets without candidate generation [17]. Integrating this new algorithm within our study seems promising. Furthermore, frequent sets discovery, and more generally data mining, is not limited to independent mining tasks (or queries). Knowledge discovery in databases is an iterative process and there are still lots of work to do to optimize sequences of queries. There is a major trade-off between fully optimizing each individual query and finding a strategy that makes use of previous mined patterns [8, 3]. This strategy may be less effective for the first queries but may win for long sequences of related queries, i.e., the way people actually proceed.

Acknowledgement The authors thank Artur Bykowski for his contribution to the experimental validation.

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'93)*, pages 207 – 216, Washington, D.C., USA, May 1993. ACM.

[2] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pages 307 – 328. AAAI Press, Menlo Park, CA, 1996.

[3] E. Baralis and G. Psaila. Incremental refinement of mining queries. In *Proceedings of the First International Conference on Data Warehousing and Knowledge Discovery (DaWaK'99)*, volume 1676 of *Lecture Notes in Computer Science*, pages 173 – 182, Florence, I, Sept. 1999. Springer-Verlag.

[4] J.-F. Boulicaut and A. Bykowski. Frequent closures as a concise representation for binary data mining. In *Proceedings of the Fourth Pacif-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'00)*, volume 1805 of *Lecture Notes in Artificial Intelligence*, pages 62 – 73, Kyoto, JP, Apr. 2000. Springer-Verlag.

[5] J.-F. Boulicaut, A. Bykowski, and B. Jeudy. Mining association rules with negations. Technical report, INSA Lyon, LISI, F-69621 Villeurbanne, Nov. 2000.

[6] J.-F. Boulicaut, A. Bykowski, and C. Rigotti. Approximation of frequency queries by mean of free-sets. In *Proceedings of the Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases PKDD'00*, volume 1910 of *Lecture Notes in Artificial Intelligence*, pages 75 – 85, Lyon, F, Sept. 2000. Springer-Verlag.

[7] J.-F. Boulicaut and B. Jeudy. Using constraint for itemset mining: should we prune or not? In *Proceedings Bases de Données Avançées" BDA'00*, pages 221 – 237, Blois, F, Oct. 2000.

[8] B. Goethals and J. van den Bussche. On implementing interactive association rule mining. In *Proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'99)*, Philadelphia, USA, May 30 1999.

[9] G. Grahne, L. V. S. Lakshmanan, and X. Wang. Efficient mining of constrained correlated sets. In *Proc. ICDE 2000*, pages 512 – 521, San Diego, USA, 2000.

[10] L. V. Lakshmanan, R. Ng, J. Han, and A. Pang. Optimization of constrained frequent set queries with 2-variable constraints. In *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'99)*, pages 157 – 168, Philadelphia, USA, 1999. ACM Press.

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

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

[13] R. Ng, L. V. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained associations rules. In *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'98)*, pages 13 – 24, Seattle, Washington, USA, 1998. ACM Press.

[14] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Closed set based discovery of small covers for association rules. In *Proc. BDA'99*, pages 361 – 381, Bordeaux, F, Oct. 1999.

[15] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Efficient mining of association rules using closed itemset lattices. *Information Systems*, 24(1):25 – 46, Jan. 1999.

[16] J. Pei, J. Han, and L. V. S. Lakshmanan. Mining frequent itemsets with convertible constraints. In *Proc. ICDE'01*, Heidelberg, Germany, 2001.

[17] J. Pei, J. Han, and R. Mao. CLOSET an efficient algorithm for mining frequent closed itemsets. In *Proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'00)*, Dallas, USA, May 2000.

[18] R. Srikant, Q. Vu, and R. Agrawal. Mining association rules with item constraints. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD'97)*, pages 67 – 73, Newport Beach, California, USA, 1997. AAAI Press.

[19] M. J. Zaki. Generating non-redundant association rules. In *Proc. KDD'00*, pages 34 – 43, Boston, USA, 2000.