Constraint-Based Discovery of a Condensed Representation for Frequent Patterns

Jean-François Boulicaut and Baptiste Jeudy *

Institut National des Sciences Appliquées de Lyon, Laboratoire d'Ingénierie des Systèmes d'Information, Bâtiment Blaise Pascal, 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 itemsets 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 the computation of condensed representations for frequent itemsets, e.g., frequent closed sets. We show that the benefit of these two approaches can be combined into a levelwise algorithm that computes the so-called δ-free sets under a conjunction of anti-monotone and monotone constraints. Applications of this algorithm are briefly discussed. Among others, it can be used for the discovery of association rules in difficult cases (dense and highly-correlated data).

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 itemsets 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,

* This research is part of the cInQ project (IST 2000-26469) that is partially funded by the European Commission IST Programme - Future and Emergent Technologies.

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* [19, 15, 12, 8]. These studies have considered various kinds of constraints, including "syntactic" constraints (e.g., an item must not appear) and constraints related to the so-called objective measures of 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 [15, 8]. This framework has been also studied for other kinds of properties like rules [12] or correlations [11].

Another promising approach concerns the *condensed representation of frequent itemsets* [13]. 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 itemsets as a valuable condensed representation of frequent itemsets [16, 4, 7, 18, 20].

To the best of our knowledge, combining these two frameworks has not been studied yet. In this paper, we show how the benefit of these two approaches can be combined into a levelwise algorithm and we emphasize the potential for applications. The concept of δ-free set introduced in [7] is related to the discovery of closed sets and constitutes our targeted condensed representation of frequent patterns. We provide an algorithm that computes such a representation when the δ-free sets fulfill a conjunction of anti-monotone and monotone constraints. Fortunately, many useful constraints fall in this category. Doing so, difficult mining tasks can be considered like frequent itemset mining for "low" frequency thresholds, the direct computation of representative association rules or the discovery of frequent generalized itemsets (sets that combine positive and negative items).

In Section 2, we provide the formal background about δ-free sets and outline an effective levelwise algorithm for constraint-based discovery of this representation. In Section 3, we discuss the applicability of the framework, pointing out some interesting mining tasks from the practitioner point of view. Section 4 is a short conclusion.

2 An Algorithm for Computing Frequent δ-Free Sets under Constraints

2.1 Problem Settings and Notations

Given a finite set `Items` of symbols called *attributes* (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. The set of k-itemsets is denoted $\mathtt{Items}_k$. 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 $|.|$ denotes the cardinality of the multiset (each transaction is counted with its multiplicity). An itemset S is γ*-frequent* in T if $\mathcal{F}(S)\geq\gamma$. Table 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 $\{\mathtt{A},\mathtt{B}\}$.

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

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

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	$\{t_1,t_2,t_3,t_4\}$	0.67
ABC	$\{t_1,t_4,t_6\}$	0.5

An *association rule* is a rule $X\Longrightarrow Y$ where X, $Y\subseteq$ `Items` and $X\cap Y=\emptyset$. The *frequency* of the association rule $X\Longrightarrow Y$ is $\mathcal{F}(X\Longrightarrow Y)=\mathcal{F}(X\cup Y)$. The *confidence* of $X\Longrightarrow Y$ is $\mathcal{C}onf(X\Longrightarrow Y)=\mathcal{F}(X\Longrightarrow Y)/\mathcal{F}(X)$. It is the conditional probability of having Y when X is supported by a transaction. An exception to an association rule $X\Longrightarrow Y$ is a transaction t such that t supports X and t does not support Y.

Let us formalize the concept of δ-free set introduced in [7]. For this we need to define the $\mathtt{closure}_\delta$ operator.

Definition 1 ($\mathtt{closure}_\delta$ operator). *Given a positive integer δ and an itemset S, $\mathtt{closure}_\delta(S)$ is the maximal (w.r.t. the set inclusion) superset Y of S such that for every item $A\in Y-S$, $|Support(S\cup\{\mathtt{A}\})|$ is at least $|Support(S)|-\delta$. Formally, $\mathtt{closure}_\delta(S)=S\cup\{\mathtt{A},|Support(S)|-|Support(S\cup\{\mathtt{A}\})|\leq\delta\}$.*

Definition 2 (δ-free sets). *Assume a positive integer δ, a δ-*free set* is an itemset S such that S is not included in the $\mathtt{closure}_\delta$ of any of its proper subset: $\forall Y\subset S$, $S\not\subseteq\mathtt{closure}_\delta(Y)$.*

In other terms, a δ-free set is a set such that no association rule with less than δ exceptions holds between its attributes. Notice that when $\delta = 0$, a 0-free set is such that no association rule with confidence 1 hold between subsets of X. A *closed itemset* X is an itemset such that $\mathtt{closure}_0(X) = X$.

Example 1. Considering the data in Table 1, $\mathtt{closure}_0(\mathtt{A}) = \mathtt{AC}$, $\mathtt{closure}_0(\mathtt{AC}) = \mathtt{AC}$ and $\mathtt{closure}_0(\mathtt{B}) = \mathtt{BC}$. With $\delta \neq 0$, $\mathtt{closure}_1(\mathtt{B}) = \mathtt{ABC}$ and $\mathtt{closure}_2(\mathtt{B}) = \mathtt{ABCD}$.

Let us explain why the collection of δ-free sets in a given matrix can be considered as a condensed representation of the frequent itemsets. First, there is much less δ-free sets than frequent itemsets for a given frequency threshold. Then, when the frequency of a δ-free set X is known, it can be used to approximate the frequency of any frequent itemset that is a superset of X [7]. A special case is that when $\delta = 0$, we can infer the exact frequency for every superset of X within the sub-lattice (w.r.t. the set inclusion) whose top is $\mathtt{closure}_0(\mathtt{X})$, the closure of X. For $\delta > 0$, we have a bounded error on the frequency value but it has been shown that, for small values of δ, this error is very low in practice [7]. So, every use of frequent itemsets can be done from condensed representation of frequent itemsets.

Definition 3 (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 $2^{\mathtt{Items}}$, we define $SAT_{\mathcal{C}}(I) = \{S \subseteq I,\ S \text{ satisfies } \mathcal{C}\}$. $SAT_{\mathcal{C}}$ 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 2. Assume the data in Table 1 ($\mathtt{Items}= \{\mathtt{A},\mathtt{B},\mathtt{C},\mathtt{D}\}$). If $\mathcal{C}_{freq}$ specifies that an itemset must be 0.6-frequent, then $\mathrm{SAT}_{\mathcal{C}_{freq}} = \{\mathtt{A,B,C,D,AC,AD,BC,CD,ADC}\}$. Assume that $\mathcal{C}_{size}(S) \equiv |S| \leq 2$ and $\mathcal{C}_{miss}(S) \equiv \mathtt{A} \notin S$, then $\mathrm{SAT}_{\mathcal{C}_{size} \wedge \mathcal{C}_{miss}} = \{\mathtt{B},\mathtt{C},\mathtt{D},\mathtt{BC},\mathtt{BD},\mathtt{CD}\}$ and $\mathrm{SAT}_{\mathcal{C}_{freq} \wedge \mathcal{C}_{size} \wedge \mathcal{C}_{miss}} = \{\mathtt{B},\mathtt{C},\mathtt{D},\mathtt{BC},\mathtt{CD}\}$.

Two kinds of constraints are considered in this paper, the anti-monotone and the monotone constraints.

Definition 4 (Monotonicity and anti-monotonicity). *An* anti-monotone *(resp.* 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 $\mathcal{C}$ (resp. $(S' \supset S \wedge S$ satisfies $\mathcal{C}) \Rightarrow S'$ satisfies $\mathcal{C}$).*

Example 3. $\mathcal{C}_{freq}$, $\mathtt{A} \notin S$, $S \subseteq \{\mathtt{A},\mathtt{B},\mathtt{C}\}$ and $S \cap \{\mathtt{A},\mathtt{B},\mathtt{C}\} = \emptyset$ are anti-monotone constraints. $\{\mathtt{A},\mathtt{B},\mathtt{C},\mathtt{D}\} \subset \mathtt{S}$ and $S \cap \{\mathtt{A},\mathtt{B},\mathtt{C}\} \neq \emptyset$ are monotone constraints. Assuming that items in S are correlated is monotone too [11].

Notice that a disjunction or a conjunction of anti-monotone (resp. monotone) constraints is an anti-monotone (resp. monotone) constraint and that the negation of an anti-monotone (resp. monotone) constraint is a monotone (resp. anti-monotone) constraint.

2.2 A Levelwise Algorithm

In this section, we provide an algorithm that discovers δ-free sets under constraints.

Assume a monotone constraint $\mathcal{C}_m = \neg\mathcal{C}'_{am}$ and an anti-monotone constraint $\mathcal{C}_{am}$. We want to mine itemsets that satisfy these two constraints and that are δ-free sets. Notice that generally, the $\mathcal{C}_{am}$ constraint contains the frequency constraint $\mathcal{C}_{freq}$ but this is not required.

First, we have to give a new definition for constrained δ-free sets.

Definition 5 (constrained δ-free sets). *Let δ be an integer and $\mathcal{C}_m$ a monotone constraint. A* constrained δ-free set *is an itemset S such that S is not included in the* `closure`$_\delta$ *of any of its proper subset* that satisfy $\mathcal{C}_m$*: $\forall Y \subset S$ $\mathcal{C}_m(Y) \Rightarrow S \not\subseteq \texttt{closure}_\delta(Y)$.*

This definition is needed because if we want to use the constraints during the extraction, the `closure`$_\delta$ of itemsets that do not satisfy the constraint $\mathcal{C}_{am} \wedge \mathcal{C}_m$ are not computed thus making the checking of the original δ-freeness impossible.

Let us now define the constraint associated with the constrained δ-free sets, i.e., a constraint that is true only on constrained δ-free sets.

Definition 6 (constraint for constrained δ-free sets). *The constrained δ-free sets are the ones that satisfy the constraint $\mathcal{C}_{Free \wedge \mathcal{C}_m}(S) \equiv (S' \subset S \wedge |S'| = |S| - 1 \wedge \mathcal{C}_m(S')) \Rightarrow S \not\subseteq \texttt{closure}_\delta(S')$*

An important property of this constraint is its anti-monotonicity. It means that within a levelwise algorithm, $\mathcal{C}_{Free \wedge \mathcal{C}_m}$ can be used to prune large parts of the lattice of itemsets. It is a key issue for explaining the efficiency of the technique.

We can give a generic algorithm for a constraint $\mathcal{C} = \mathcal{C}_{am} \wedge \mathcal{C}_{Free \wedge \mathcal{C}_m} \wedge \mathcal{C}_m = \mathcal{C}_{am} \wedge \mathcal{C}_{Free \wedge \mathcal{C}_m} \wedge \neg\mathcal{C}'_{am}$. We suppose that $\mathcal{C}_{am}$ and $\mathcal{C}_m$ are respectively anti-monotone and monotone constraints and that $\mathcal{C}_m \neq \mathcal{C}_{true}$ ($\mathcal{C}_{true}$ is the constraint that is always true), this is easy to check since $\mathcal{C}_m = \mathcal{C}_{true} \Leftrightarrow \mathcal{C}_m(\emptyset) = $ true.

This algorithm uses the functions `prune`$_\texttt{m}$ and `generate`$_m$ which are described in the next two subsections.

Generic algorithm

```
1. $C^g_1$ :=generate$_m(\emptyset, 0)$ ; $\mathcal{L}_0 = \emptyset$
2. $k := 1$
3. while $C^g_k \neq \emptyset$ do
4.     Phase 1 - candidate safe pruning
       $C_k$ := prune$_m(C^g_k, \mathcal{L}_{k-1})$
```

```
5.      Phase 2 anti-monotone constraint checking
        $\mathcal{L}_k := \text{SAT}_{\mathcal{C}_{am} \wedge \mathcal{C}_{Free} \wedge \mathcal{C}_m}(C_k)$
6.      Phase 3 - candidate generation for level k+1
        $C^g_{k+1} :=$ generate$_m(\mathcal{L}_k, k)$
7.      $k := k + 1$
    od
8.  output $\bigcup_{i=1}^{k-1} \mathcal{L}_i$
```

Like Apriori, this algorithm does a levelwise exploration of the lattice of itemsets (w.r.t. the set inclusion). During the first pass (when $k = 1$), it computes 1-itemsets that satisfy $\mathcal{C}$ and then it generates candidate 2-itemsets from them. In the second pass ($k = 2$), it prunes some candidate 2-itemsets (see the `prune`$_m$ function), discards those that do not satisfy $\mathcal{C}$ and generates candidate 3-itemsets from 2-itemsets that satisfy $\mathcal{C}$... The set C^g_k denotes the k-itemsets that can potentialy satisfy $\mathcal{C}$. During Phase 1, some of these k-itemsets are pruned. During Phase 2, a database scan is performed to compute the δ-closure and the frequency of the candidate itemsets. Those that satisfy $\mathcal{C}$ are stored in $\mathcal{L}_k$. Their δ-closures and frequencies are also stored. In Phase 3, k-itemsets in $\mathcal{L}_k$ are used to compute candidate $k+1$-itemsets. The generation function `generate`$_m$ is described later. This generation function 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.

Note that it is not necessary to check $\mathcal{C}_m$ during Phase 2 to ensure the correctness of this algorithm (our generation function ensure that all candidate itemsets satisfy $\mathcal{C}_m$).

Example 4. Given the data from Table 1, $\delta = 0$ and the frequency threshold 0.5, the output of the algorithm is $\{\emptyset^{\mathtt{C}}, \mathtt{A}^{\mathtt{C}}, \mathtt{B}^{\mathtt{C}}, \mathtt{D}^{\mathtt{AC}}, \mathtt{AB}^{\mathtt{C}}\}$ and the frequencies of these itemsets. The notation $\mathtt{D}^{\mathtt{AC}}$ means that `D` is a 0-free set and that $\mathtt{closure}_0(\mathtt{D}) = \mathtt{ACD}$. If we consider the constraint $\mathtt{D} \not\in S$ with $\delta = 1$ then it outputs $\{\emptyset^{\mathtt{A(-1)C}}, \mathtt{B}^{\mathtt{A(-1)C}}\}$. The notation $\mathtt{B}^{\mathtt{A(-1)C}}$ means that `B` is a 1-free set, $\mathtt{closure}_1(\mathtt{B}) = \mathtt{ABC}$, $\mathcal{F}(\mathtt{B}) = \mathcal{F}(\mathtt{BC})$, and $|Support(\mathtt{AB})| - |Support(\mathtt{B})| = -1$. Here, -1 is the *miss-count* of `A` with respect to `B`.

Theorem 1. *Assuming that $\mathcal{C}_{am}$ and $\mathcal{C}_m$ are respectively anti-monotone and monotone constraints and that $\mathcal{C}_m$ is not a trivial constraint, i.e., that $\mathcal{C}_m$ is not always true, this algorithm is correct and complete, i.e., it outputs exactly $SAT_{\mathcal{C}}$.*

2.3 Generation Function

In this section, the generation function for the generic algorithm is presented. We use the concept of negative border [14]. 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 $\mathcal{C}_{am}$.

This function makes use of an extended generation function. First, let us define two functions: $\texttt{generate}_1(\mathcal{L}_k) = \{A \cup B$, where $A \in \mathcal{L}_k$ and B is a 1-itemset$\}$ and $\texttt{generate}_2(\mathcal{L}_k) = \{A \cup B$, where $A, B \in \mathcal{L}_k$, A and B share $k-1$ items $\}$ and let $ms = \text{ Max }_{S \in \mathcal{B}d^-_{\mathcal{C}'_{am}}} |S|$.
Our new generation function is denoted $\texttt{generate}_m$.

```
function $\texttt{generate}_m(\mathcal{L}, k)$
    if $k = 0$ then return $\mathcal{B}d^-_{\mathcal{C}'_{am}} \cap \texttt{Items}_1$
    elsif $k < ms$ then return $\texttt{generate}_1(\mathcal{L}) \cup (\mathcal{B}d^-_{\mathcal{C}'_{am}} \cap \texttt{Items}_{k+1})$
    elsif $k = ms$ then return $\texttt{generate}_1(\mathcal{L})$
    elsif $k > ms$ then return $\texttt{generate}_2(\mathcal{L})$
    fi
```

Theorem 2. *This candidate generation function is complete and ensures that every candidate itemset verifies* $\neg\mathcal{C}'_{am}$.

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

2.4 Safe Pruning Function

Let us now introduce a correct and complete pruning algorithm.

```
function $\texttt{prune}_m(C, \mathcal{L})$
    $C' := C$
    for all $S \in C$ do for all $S' \subset S$ such that $|S'| = |S| - 1$
        do if $S' \notin \mathcal{L}_k$ and $\mathcal{C}_m(S') = true$
            then delete $S$ from $C'$ od
    od
    return $C'$
```

The next theorem states that this algorithm is correct, i.e., it does not prune any itemset that satisfy $\mathcal{C} = \mathcal{C}_{am} \wedge \mathcal{C}_{Free \wedge \mathcal{C}_m} \wedge \mathcal{C}_m$. The completeness of this algorithm means that it does not exists a more efficient pruning algorithm, i.e., it is not possible to prune more itemset without affecting the completeness of the generic algorithm.

Theorem 3. *The pruning algorithm* $\texttt{prune}_m$ *is correct and complete (when used in our generic algorithm).*

2.5 Related Work

Our generic algorithm is inspired by several algorithms [19, 10, 15] and can be considered as a generalization of them. Conjunctions of monotone and anti-monotone constraints cover every kind of constraints that have been "pushed" inside a levelwise algorithm (another kind of interesting constraint, the convertible constraints [17], can be pushed in depth-first exploration algorithms). The framework of succinct constraints introduced in [15] allows to find an effective generation function (i.e., an effective computation of the negative border $\mathcal{B}d^-_{\mathcal{C}'_{am}}$ of Theorem 1). Frequent δ-free sets are a condensed representation of frequent itemsets. Another related condensed representation is the collection of frequent closed sets [16, 4, 20].

3 Multiple Uses of Frequent δ-Free Sets

When an itemset S is not 0-free, it means that it is included in the $\mathtt{closure}_0$ of a 0-free set X. In this case their frequencies are the same. It is therefore straightforward to compute all frequent sets with the 0-free ones. Frequent 0-free sets act as the generators of frequent closed itemsets in an algorithm like Close [16]. For this special case, this has been independently formalized under the concept of key pattern in [3].

If the collection of 0-free sets is computed, it is possible to infer from it the exact knowledge of the frequent itemsets and their frequencies. By computing a smaller collection using less resources (see [7] for practical experimentations), every application that makes use of frequent itemsets can be considered. Among others, they can be used for deriving standard association rules [2], the computation of similarities between attributes [9] or the basis for boolean formula frequency approximation [13]. Notice also, that when we mine frequent association rules, most of the interestingness objective measures that have been proposed in order to alleviate confidence drawbacks (e.g., lift, conviction, J-measure) can be evaluated without looking back to the data, i.e., using only the frequencies of the frequent itemsets.

When $\delta \neq 0$, the frequencies of the frequent itemsets can be approximated (overestimated) but the error remains very low in practice [7]. Due to this approximation on the frequency, some of the generated itemsets can be infrequent: e.g., if the frequency threshold is 0.3 and the computed frequency of an itemset is 0.29 ± 0.02, this itemset may be infrequent. However, it is certain that no frequent itemset is missed.

When using other anti-monotone and/or monotone constraints on δ-free sets, the same generation algorithms can be used to generate constrained itemsets from constrained δ-free sets. In this case too, it is possible that some of the generated itemsets do not satisfy the anti-monotone constraints (further checking is needed to ensure this). However, all itemsets that satisfy the constraints are generated, therefore with a last testing step it is possible to generate exactly the itemsets that satisfy the constraint.

The interesting point here is that the extraction remains tractable for frequency thresholds that can not be considered with other methods. Again, it is possible to make use of the derived collection of frequent itemsets even though a small error is made on the frequencies.

In [5], useful constraints for mining association rules with negations have been experimented. An association rule with negation can contain either positive items or negative items: they are derived from generalized itemsets.

Example 5. the set $\{\mathtt{A}, \mathtt{C}, \neg\mathtt{D}\}$ is a generalized itemset that is 0.5-frequent in the data from Table 1.

It is well-known that mining frequent generalized itemsets is very difficult (see [6] for a general discussion). One application-driven possibility is to use constraints in order to drastically prune the search space although focusing on potentially interesting itemsets. For instance, considering frequent itemsets ($\mathcal{C}_{freq}$) that do not involve only negative items ($\mathcal{C}_{alpp}$) sounds reasonable. $\mathcal{C}_{alpp}(S)$ is true when S involves at least p positive items. Notice that it is a monotone constraint. However, first experiments have shown that it was interesting to relax such a constraint (i.e., accepting more sets) in order to give rise to more pruning (see [5] for a complete discussion and experimental results). Following that guideline, instead of $\mathcal{C}_{alpp}$, [5] consider 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). The extraction of frequent δ-free-sets under such constraints has given promising results.

4 Conclusion

We study the discovery of δ-free sets under constraints within a levelwise algorithm. Several interesting results have been already published the last three years, e.g., about the effective use of anti-monotone constraints or 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 and have discussed its potential of applications.

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 [10]. 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.

References

1. Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia, editors, *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'93)*, pages 207–216, Washington, D.C., USA, May 1993. ACM.

2. Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI Press, Menlo Park, CA, 1996.
3. Yves Bastide, Rafik Taouil, Nicolas Pasquier, Gerd Stumme, and Lotfi Lakhal. Mining frequent patterns with counting inference. *SIGKDD Explorations*, 2(2):66–75, December 2000.
4. Jean-François Boulicaut and Artur 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, April 2000. Springer-Verlag.
5. Jean-François Boulicaut, Artur Bykowski, and Baptiste Jeudy. Mining association rules with negations. Technical report, INSA Lyon, LISI, F-69621 Villeurbanne, November 2000.
6. Jean-François Boulicaut, Artur Bykowski, and Baptiste Jeudy. Towards the tractable discovery of association rules with negations. In *Proceedings of the Fourth International Conference on Flexible Query Answering Systems FQAS'00*, Advances in Soft Computing series, pages 425–434, Warsaw, PL, October 2000. Springer-Verlag.
7. Jean-François Boulicaut, Artur Bykowski, and Christophe Rigotti. Approximation of frequency queries by mean of free-sets. In J. Komorowski D. Zighed and J. M. Zytkow, editors, *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, September 2000. Springer-Verlag.
8. Jean-François Boulicaut and Baptiste Jeudy. Using constraint for itemset mining: should we prune or not? In *Proceedings Bases de Données Avançées" BDA'00*, pages 221–237, Blois, F, October 2000.
9. Gautam Das, Heikki Mannila, and Pirjo Ronkainen. Similarity of attributes by external probes. Technical Report C-1997-66, University of Helsinki, Department of Computer Science, P.O. Box 26, FIN-00014 University of Helsinki, Finland, October 1997.
10. Bart Goethals and Jan 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.
11. Gösta Grahne, Laks V. S. Lakshmanan, and Xiaohong Wang. Efficient mining of constrained correlated sets. In *Proc. ICDE 2000*, pages 512–521, San Diego, USA, 2000.
12. Laks V.S. Lakshmanan, Raymond Ng, Jiawei Han, and Alex 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.
13. Heikki Mannila and Hannu 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, August 1996. AAAI Press.
14. Heikki Mannila and Hannu Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.

15. Raymond Ng, Laks V.S. Lakshmanan, Jiawei Han, and Alex Pang. Exploratory mining and pruning optimizations of constrained associations rules. In Laura M. Haas and Ashutosh Tiwary, editors, *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'98)*, pages 13–24, Seattle, Washington, USA, 1998. ACM Press.
16. Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Efficient mining of association rules using closed itemset lattices. *Information Systems*, 24(1):25–46, January 1999.
17. Jian Pei, Jiawei Han, and Laks V. S. Lakshmanan. Mining frequent itemsets with convertible constraints. In *Proc. ICDE'01*, Heidelberg, Germany, 2001.
18. Jian Pei, Jiawei Han, and Runying 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.
19. Ramakrishnan Srikant, Quoc Vu, and Rakesh Agrawal. Mining association rules with item constraints. In David Heckerman, Heikki Mannila, Daryl Pregibon, and Ramasamy Uthurusamy, editors, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD'97)*, pages 67–73, Newport Beach, California, USA, 1997. AAAI Press.
20. Mohammed Javeed Zaki. Generating non-redundant association rules. In *Proc. KDD'00*, pages 34–43, Boston, USA, 2000.