A Condensed Representation to Find Frequent Patterns

Artur Bykowski Christophe Rigotti
Laboratoire d'Ingénierie des Systèmes d'Information
INSA Lyon, Bâtiment 501
F-69621 Villeurbanne Cedex, France
Tel. +33 4 72 43 60 49, Fax. +33 4 72 43 87 13
Artur.Bykowski,Christophe.Rigotti@insa-lyon.fr

ABSTRACT

Given a large set of data, a common data mining problem is to extract the frequent patterns occurring in this set. The idea presented in this paper is to extract a condensed representation of the frequent patterns called disjunction-free sets, instead of extracting the whole frequent pattern collection. We show that this condensed representation can be used to regenerate all frequent patterns and their exact frequencies. Moreover, this regeneration can be performed without any access to the original data. Practical experiments show that this representation can be extracted very efficiently even in difficult cases. We compared it with another representation of frequent patterns previously investigated in the literature called frequent closed sets. In nearly all experiments we have run, the disjunction-free sets have been extracted much more efficiently than frequent closed sets.

1. INTRODUCTION

An important data mining problem is to extract efficiently the frequent patterns occurring in a large data set. In this paper, we consider the extraction of frequent patterns called *frequent itemsets*. This problem can be shortly stated as follows, in the context of a common data mining task : *basket analysis*. A collection of purchases of customers is encoded in a table, where each row represents a customer basket (i.e., a set of items purchased together). A toy example, with four items A,B,C and D is depicted in Table 1, using 'x' to denote the occurrences of the items in the baskets. Let r be such a table and X be a set of items. The *support* of X in r denoted $Sup(r, X)$ is the number of baskets in r containing all items in X. Then the frequent itemset mining problem is to find all pairs $\langle X, Sup(r, X)\rangle$ such that $Sup(r, X)$ exceeds a given threshold.

Example 1. If we consider the baskets represented in Table 1 and a support threshold value of 2, then the problem is to find the pairs $\langle \emptyset, 8\rangle$, $\langle \{A\}, 4\rangle$, $\langle \{A, B\}, 2\rangle$, $\langle \{A, B, C\}, 2\rangle$, $\langle \{A, C\}, 2\rangle$, $\langle \{A, D\}, 2\rangle$, $\langle \{B\}, 4\rangle$, $\langle \{B, C\}, 4\rangle$, $\langle \{C\}, 5\rangle$ and $\langle \{D\}, 4\rangle$.

Table 1: Baskets of customers

A	B	C	D
x	x	x	
	x	x	x
x			x
			x
		x	
x			x
	x	x	
x	x	x	

In practice the number of rows and the number of items are rather large and naïve solutions can not be used. Several different efficient techniques have been proposed to perform this task on large data sets (e.g., [2, 11, 6]). An alternative promising approach has been developed in [9, 5], based on the following idea: instead of mining all frequent patterns, it is sufficient to extract a particular subset of the frequent pattern collection, such that we can regenerate from this subset the whole collection. In this paper, this subset will be called a *condensed representation*[1] of the frequent pattern collection.

Ideally, a condensed representation is much smaller than the original collection and can be extracted more efficiently, while allowing a quick regeneration of all frequent patterns without costly scan of the original data and new support counting.

To our knowledge, two condensed representations have been proposed in the literature for frequent itemsets: *closed sets* [9] and *δ-free sets* [5]. The first framework allows regenerating exactly the frequencies of the patterns, while the second leads to an approximation of these frequencies.

In this paper, we propose a new condensed representation called disjunction-free sets. Informally, this representation is based on the following idea. Let A, B, C and D represent items in a table r. If the rule $A \wedge B \Rightarrow C \vee D$ holds in the data (i.e., if A and B occur in a row then C or D also occur in this row) then we can determine the support of item-

[1]We borrow this term from [7].

set $\{A, B, C, D\}$ using the supports of $\{A, B\}$, $\{A, B, C\}$ and $\{A, B, D\}$. This can be done according to the following observation: the sum of the supports of $\{A, B, C\}$ and $\{A, B, D\}$ is equal to the sum of the supports of $\{A, B\}$ and $\{A, B, C, D\}$. So if we know the supports of $\{A, B\}$, $\{A, B, C\}$ and $\{A, B, D\}$, then we can avoid the extraction and the support counting of $\{A, B, C, D\}$. In our terminology, $\{A, B, C, D\}$ will be a non-disjunction-free set. In this paper, we formalize a representation based on the disjunction-free sets and show that it can be used to regenerate all frequent itemsets and their exact supports.

Next, we show that the frequent disjunction-free sets are an interesting condensed representation of the frequent closed sets, which have already been shown to be an interesting condensed representation of the frequent itemsets [9]. We compare the extraction of the frequent disjunction-free sets using breadth-first and depth-first strategies to the corresponding state-of-the-art algorithms proposed for closed sets. In nearly all experiments, the extraction of the frequent disjunction-free sets is significantly more efficient.

We also consider experimentally the generation of all frequent closed sets from the collection of frequent disjunction-free sets. Since the latter collection contains the information about the supports of all frequent itemsets the generation of all frequent closed sets may be performed without accessing the original data. In this case, the experiments show that it is in general much more efficient to begin with an extraction of the disjunction-free sets and then to generate the frequent closed sets, instead of extracting directly the closed sets. Moreover, we always measure a gain of efficiency in the most difficult cases (i.e. at the lowest support thresholds).

Organization of the paper. In the next section we provide preliminary definitions used in this paper. In Section 3, we define the notion of disjunction-free set, and show that these sets can be used as a condensed representation for frequent itemsets. In Section 4, we present both breadth-first and depth-first algorithms to extract the frequent disjunction-free sets. In Section 5, we present practical experiments showing that frequent disjunction-free sets can be extracted efficiently and in most cases more efficiently than frequent closed sets. Finally, we conclude with a summary.

2. PRELIMINARY DEFINITIONS

When possible, we follow the notational conventions and definitions of [7, 8]. In particular, we use multisets to represent collections of rows and given such a multiset r, we write $t \in r$ to denote that a particular row t belongs to r.

Definition 1. **(binary database)** Let R be a set of symbols called items. A *row* is a subset of R. A *binary database* r *over* R is a multiset of rows.

Definition 2. **(support and frequent itemsets)** We note $\mathcal{M}(r, X) = \{t \in r | X \subseteq t\}$ the multiset of rows matched by the itemset X and $Sup(r, X) = |\mathcal{M}(r, X)|$ the *support* of X in r, i.e., the number of rows matched by X. Let σ be a support threshold (σ is an absolute number of rows), $Freq(r, \sigma) = \{X | X \subseteq R$ and $Sup(r, X) \geq \sigma\}$ is the set of all σ-frequent itemsets in r.

In this paper, to keep the presentation concise, we consider the following notational conventions to handle a *generalized* form of itemsets.

Definition 3. **(generalized itemsets)** Let R be a set of symbols. The symbols in R will be called *positive* items, and for each positive item $A \in R$ we consider a *negative* item noted $\overline{A}$. $Gen(R) = R \cup \{\overline{A} | A \in R\}$ is the set of generalized items based on R and a subset X of $Gen(R)$ is called a *generalized itemset* based on R. We denote the items appearing positively and negatively as follows, $Pos(X) = \{A \in R | A \in X\}$ and $Neg(X) = \{A \in R | \overline{A} \in X\}$.

We generalize in this context $\mathcal{M}$ and Sup.

Definition 4. **(generalized support)** Let r be a binary database over R and X be a generalized itemset based on R. Then $Gen\mathcal{M}(r, X) = \{t \in r | Pos(X) \subseteq t \wedge Neg(X) \cap t = \emptyset\}$ is the multiset of rows matched by X and the support of X in r is $GenSup(r, X) = |Gen\mathcal{M}(r, X)|$.

Intuitively, the rows matched by a generalized itemset X are the rows that contain all positive items in X but none of the items appearing under a negative form in X. For example, in Table 1 the generalized support of $\{\overline{A}, B, C\}$ is 2.

3. DISJUNCTION-FREE SETS

The notion of disjunction-free set will be based on a kind of dependency between the items called *disjunctive rules* and defined as follows.

Definition 5. **(simple disjunctive rule)** Let X be a set of (positive) items, a *simple disjunctive rule* based on X is an expression of the form $Y \Rightarrow A \vee B$, where $Y \subset X$ and $A, B \in X \setminus Y$. Notice that A and B are single items and also that a rule of the form $Y \Rightarrow A \vee A$ is a particular case of simple disjunctive rule. Let r be a binary database over R where $X \subseteq R$. The simple disjunctive rule $Y \Rightarrow A \vee B$ is *valid* in r if and only if $\mathcal{M}(r, Y) = \{t \in r | t \in \mathcal{M}(r, Y \cup \{A\}) \vee t \in \mathcal{M}(r, Y \cup \{B\})\}$.

Now, we state three lemmas. The first is a key implementation property for the mining of disjunction-free sets (see Section 4.2.2), and the two others are fundamentals needed in the rest of the paper.

The proof of the first lemma is immediate using the definitions 4 and 5.

LEMMA 1. *Let r be a binary database over R, and Y be a set of items $Y \subseteq R$. Let A and B be single items in R, and $\overline{A}, \overline{B}$ be their corresponding negative items. Then $Y \Rightarrow A \vee B$ is a valid simple disjunctive rule in r if and only if $GenSup(r, Y \cup \{\overline{A}, \overline{B}\}) = 0$.*

The proofs of the two following lemmas are also immediate and thus omitted.

LEMMA 2. *Let r be a binary database over R, and X, Y be itemsets, $X, Y \subseteq R$, such that $Y \subseteq X$. Let $A, B \in Y$. If $Y \setminus \{A, B\} \Rightarrow A \vee B$ is valid in r then $X \setminus \{A, B\} \Rightarrow A \vee B$ is also valid in r.*

LEMMA 3. *Let r be a binary database over R, $X \subseteq R$ be an itemset, and A, B be items in X. Then there exists $Y \subset X$ such that $Y \Rightarrow A \vee B$ is a valid simple disjunctive rule based on X, if and only if $Sup(r, X) = Sup(r, X \setminus \{A\}) + Sup(r, X \setminus \{B\}) - Sup(r, X \setminus \{A, B\})$.*

We define now the notion of disjunction-free sets, and show that the collection of frequent disjunction-free sets together with a collection of itemsets called the negative border (defined below), are sufficient to determine the support of any frequent itemset.

Definition 6. **(disjunction-free set)** Let r be a binary database over R, $X \subseteq R$ is a *disjunction-free set* w.r.t. r if and only if there is no valid simple disjunctive rule based on X in r. The set of all disjunction-free sets w.r.t. r is noted $DFree(r)$.

Definition 7. **(frequent disjunction-free set)** Let r be a binary database over a set of items R, $FreqDFree(r, \sigma) = Freq(r, \sigma) \cap DFree(r)$ denotes the set of all σ-frequent disjunction-free sets w.r.t. r.

We define the concept of negative border [8] for σ-frequent disjunction-free sets. Informally, the negative border consists of the smallest itemsets (w.r.t. set inclusion) that are not σ-frequent disjunction-free sets.

Definition 8. **(negative border)** Let r be a binary database over a set of items R, the negative border of $FreqDFree(r, \sigma)$ is noted $\mathcal{B}d^-(r, \sigma)$ and is defined as follows: $\mathcal{B}d^-(r, \sigma) = \{X | X \subseteq R, X \notin FreqDFree(r, \sigma) \wedge (\forall Y \subset X, Y \in FreqDFree(r, \sigma))\}$.

We can now state the *correctness* of the representation of the frequent itemsets by means of the frequent disjunction-free sets and the negative border.

THEOREM 1. *Let r be a binary database over a set of items R, $X \subseteq R$, and σ be an absolute support threshold. Using the sets in $FreqDFree(r, \sigma)$ and in $\mathcal{B}d^-(r, \sigma)$, together with their supports, we can determine if X is σ-frequent, and when X is σ-frequent we can also determine $Sup(r, X)$.*

PROOF. Let X be any itemset. The proof is made by induction on $|X|$.

Hypothesis. Suppose that for every itemset $W \subset X$, we can determine if W is σ-frequent, and when W is σ-frequent we can also determine $Sup(r, W)$.

If $X \in FreqDFree(r, \sigma)$ then we have trivially the claim.

If $X \notin FreqDFree(r, \sigma)$. By Definition 8, $\exists Y \subseteq X, Y \in \mathcal{B}d^-(r, \sigma)$. Let Y be such an itemset.

If $Y = X$, then $Sup(r, Y) = Sup(r, X)$ and we also have the claim.

We consider now the case where $Y \subset X$.

If $Sup(r, Y) < \sigma$ then X is not σ-frequent since $Y \subseteq X$.

If $Sup(r, Y) \geq \sigma$ then Y is not a disjunction-free set since $Y \in \mathcal{B}d^-(r, \sigma)$. In this case, we now construct a valid simple disjunctive rule based on Y, and then we use this rule to decide if X is σ-frequent and to compute its support.

By Definition 6, Y is not a disjunction-free set implies that there exists $Z \subset Y$ and $A, B \in Y \setminus Z$ such that $Z \Rightarrow A \vee B$ is a valid simple disjunctive rule in r. By Lemma 3, we have $Sup(r, Z \cup \{A, B\}) = Sup(r, Z \cup \{B\}) + Sup(r, Z \cup \{A\}) - Sup(r, Z)$.

Since Y is σ-frequent and $|Y| < |X|$, by the induction hypothesis we can determine the support of all its subsets. So we can find among all subsets of Y four sets $Z, Z \cup \{A, B\}, Z \cup \{B\}, Z \cup \{A\}$ satisfying the relation above.

Moreover, since $Y \subseteq X$, by Lemma 2, we know that $X \setminus \{A, B\} \Rightarrow A \vee B$ is valid.

By the induction hypothesis, we can determine for the sets $X \setminus \{A, B\}, X \setminus \{A\}$ and $X \setminus \{B\}$ if they all are σ-frequent and in this case their supports. If at least one of these sets is not σ-frequent so neither is X and we have the claim. If the three sets are σ-frequent, since $X \setminus \{A, B\} \Rightarrow A \vee B$ is valid, using their supports and Lemma 3 we can determine the support of X. □

It should be noticed that the proof of Theorem 1 is constructive and that it can be used as a naïve recursive algorithm to determine $Sup(r, X)$.

4. DISCOVERING ALL FREQUENT DISJUNCTION-FREE SETS

In this section we describe two algorithms to mine all frequent disjunction-free sets. Two main strategies have been proposed the explore the search space during frequent itemsets mining: breadth-first (e.g., [2]) and depth-first [6, 1]. Each of them has its pros and cons, and even if we consider only the extraction time criterion there is no always-winning strategy as we will show in Section 5. So, in this section we consider the two strategies and we describe the corresponding algorithms. For each, we give an abstract version to find $FreqDFree(r, \sigma)$ and $\mathcal{B}d^-(r, \sigma)$, and then we describe the key implementation issues.

In both cases the *anti-monotonicity* of disjunction-freeness w.r.t. itemset inclusion is important for an efficient mining. This anti-monotonicity property follows directly from the definition of disjunction-free sets and is stated by the following lemma.

LEMMA 4. *Let r be a binary database over R, and X be an itemset, $X \subseteq R$. For all $Y \subseteq X$ if $X \in DFree(r)$ then $Y \in DFree(r)$.*

During the extraction of disjunction-free sets, according to this property, when a set is not disjunction-free then there is no need to consider any of its supersets. This pruning criterion will be applied in breadth-first and depth-first strategies.

4.1 Breadth-first extraction

4.1.1 Abstract algorithm

The algorithm, presented below, is formulated as an instance of the generic levelwise-search algorithm[2] presented in [8]. It explores iteratively the itemset lattice (w.r.t. set inclusion) levelwise, starting from the empty set and stopping at the level of the largest set from $FreqDFree(r,\sigma) \cup \mathcal{B}d^-(r,\sigma)$. At each iteration, it scans the database to find which sets of the current level are frequent disjunction-free sets. Then, it generates candidates for the next iteration considering only the sets of the next level for which all proper subsets are frequent disjunction-free sets.

ALGORITHM 1 (HLINEX).

Input: r a binary database over a set of items R, and σ an absolute support threshold.

Output: $FreqDFree(r,\sigma)$ and the search space explored.

1. $\mathcal{C} := \{\emptyset\}$;
2. $\mathcal{FDF} := \emptyset$; $\mathcal{C}used := \emptyset$
3. **while** $\mathcal{C} \neq \emptyset$ **do**
4. $\quad \mathcal{FDF} := \mathcal{FDF} \cup \{X | X \in \mathcal{C}$ *and* X *is* σ*-frequent disjunction-free in* $r\}$;
5. $\quad \mathcal{C}used := \mathcal{C}used \cup \mathcal{C}$;
6. $\quad \mathcal{C} := \{X | X \subseteq R$ *and* $\forall\, Y \subset X, Y \in \mathcal{FDF}\} \setminus \mathcal{C}used$;
7. **od**;
8. **output** $\langle \mathcal{FDF}, \mathcal{C}used \rangle$;

Using the anti-monotonicity of disjunction-freeness stated by Lemma 4 and the correctness result of the levelwise search algorithm of [8], the following theorem is straightforward.

THEOREM 2 (CORRECTNESS OF HLINEX). *Given r a binary database over a set of items R, and σ an absolute support threshold, Algorithm 1 computes $\langle \mathcal{FDF}, \mathcal{C}used \rangle$, and we have $FreqDFree(r,\sigma) = \mathcal{FDF}$ and $\mathcal{B}d^-(r,\sigma) = \mathcal{C}used \setminus \mathcal{FDF}$.*

4.1.2 Implementation issues

Techniques similar to the ones presented in [3] for levelwise mining of frequent itemsets are used. The candidate generation is made using a join-based function, and the itemset support counters are updated w.r.t. a row of the database using a *prefix-tree* data structure. A specific aspect is the implementation of the disjunction-freeness test. When we need to test this property for an itemset X, we already know the support of all its subsets. So we check if there exist Y and Z, subsets of X, such that $|Y| = |Z| = |X| - 1$ and $Sup(r,X) = Sup(r,Y) + Sup(r,Z) - Sup(r, Y \cap Z)$. By Lemma 3 and Definition 6, X is disjunction-free if and only if no such pair of subsets exists.

[2]Efficient frequent set mining algorithms like APRIORI [2] can also be seen as an instance of this generic algorithm.

4.2 Depth-first extraction

4.2.1 Abstract algorithm

The algorithm given below is described by means of a recursive function *Find* which explores the itemset lattice in a depth-first way. A call $Find(X,r,\sigma,\mathcal{C}old)$ finds the σ-frequent disjunction-free supersets of the itemset X in r, but discards during the exploration all sets that are already in $\mathcal{C}old$. The algorithm uses the efficient divide-and-conquer strategy of [6, 1] as follows. A call to the function will consider only the subset of the search space corresponding to supersets of X. Moreover, this subspace is further reduced to sets that have not been examined earlier using $\mathcal{C}old$. Considering only supersets of X enables an efficient support determination since, in this case, only a restricted part of the database needs to be used (i.e., the rows matched by X).

ALGORITHM 2 (VLINEX).

Function $Find(X,r,\sigma,\mathcal{C}old)$

Input: X the itemset considered as the current starting point in the search space, r a binary database over a set of items R, σ an absolute support threshold, and $\mathcal{C}old$ the itemsets in the search space that has already been explored.

Output: $FreqDFree(r,\sigma)$ and the search space explored.

1. $\mathcal{C} := \{X \cup \{A\} | A \in R\} \setminus \mathcal{C}old$;
2. $\mathcal{C}used := \mathcal{C}old$;
3. $\mathcal{FDF} := \emptyset$;
4. **for all** $Y \in \mathcal{C}$ **do**
5. $\quad \mathcal{C}used := \mathcal{C}used \cup \{Y\}$;
6. $\quad$ **if** Y *is a* σ*-frequent disjunction-free set in* r **then**
7. $\quad\quad \langle \mathcal{FDF}'', \mathcal{C}used \rangle := Find(Y, \mathcal{M}(r,Y), \sigma, \mathcal{C}used)$;
8. $\quad\quad \mathcal{FDF} := \mathcal{FDF} \cup \mathcal{FDF}'' \cup \{Y\}$;
9. $\quad$ **fi**
10. **od**;
11. **output** $\langle \mathcal{FDF}, \mathcal{C}used \rangle$;

It should be noticed that the algorithm does not consider the empty itemset, and also that it does not provide an immediate characterization of the negative border. However, the empty itemset can be handled trivially and the computation of the exact negative border can be performed in a straightforward post-processing step using $\mathcal{C}used$.

The correctness of Algorithm 2 is stated by the following theorem.

THEOREM 3 (CORRECTNESS OF VLINEX). *Given r a binary database over a set of items R, and σ an absolute support threshold, the call $Find(\emptyset,r,\sigma,\emptyset)$ returns $\langle \mathcal{FDF}, \mathcal{C}used \rangle$,*

and we have $FreqDFree(r,\sigma)\setminus\{\emptyset\} = \mathcal{FDF}$ *and* $\mathcal{B}d^-(r,\sigma)\setminus\{\emptyset\} \subseteq \mathcal{C}used\setminus\mathcal{FDF}$.

PROOF. *(sketch)*

We consider $\langle\mathcal{FDF},\mathcal{C}used\rangle$ returned by $Find(\emptyset,r,\sigma,\emptyset)$.

First, we show that $\forall X, X\in FreqDFree(r,\sigma)\setminus\{\emptyset\} \implies X\in\mathcal{FDF}$ using an induction on $|X|$.

Let $X\in FreqDFree(r,\sigma)$ and $|X|=1$ (i.e. X is a singleton itemset). Note that every singleton itemset is considered directly by $Find(\emptyset,r,\sigma,\emptyset)$ (line 1) and therefore if X is σ-frequent disjunction-free (line 6) we have $X\in\mathcal{FDF}$ (line 8).

Induction hypothesis. Suppose that the property holds for every itemset Y such that $1\leq|Y|\leq n$.

Given $X\in FreqDFree(r,\sigma)$, such that $|X|=n+1$, let Y be any subset of X satisfying $|Y|=n$.

By Lemma 4, $Y\in FreqDFree(r,\sigma)$, and then by the induction hypothesis $Y\in\mathcal{FDF}$. It follows that $Find$ has been called at least once using $Find(Y,\mathcal{M}(r,Y),\sigma,\mathcal{C}used_Y)$ where $\mathcal{C}used_Y$ was the search space already explored. This call has started by the following candidate generation $\mathcal{C}:=\{Y\cup\{A\}|A\in R\}\setminus\mathcal{C}used_Y$. So, either X was in $\mathcal{C}$ and then was tested for σ-frequent disjunction-freeness, or X was in $\mathcal{C}used_Y$ and thus has already been considered. In both cases X is collected in $\mathcal{FDF}$.

The soundness of the algorithm (i.e., $\forall X, X\in\mathcal{FDF} \implies X\in FreqDFree(r,\sigma)$) is immediate (lines 6 and 8).

Finally, we consider again the call $Find(\emptyset,r,\sigma,\emptyset)$, which returns $\langle\mathcal{FDF},\mathcal{C}used\rangle$. Let X be an element of $\mathcal{B}d^-(r,\sigma)\setminus\{\emptyset\}$, and Y be any subset of X such that $|Y|=|X|-1$. By Definition 8, $Y\in FreqDFree(r,\sigma)$, and since $\forall Z, Z\in FreqDFree(r,\sigma)\setminus\{\emptyset\} \implies Z\in\mathcal{FDF}$, we have $Y\in\mathcal{FDF}$ unless $Y=\emptyset$. Using the same reasoning as above we know that $X\in\mathcal{C}used$, and thus $\mathcal{B}d^-(r,\sigma)\setminus\{\emptyset\}\subseteq\mathcal{C}used\setminus\mathcal{FDF}$. □

4.2.2 Implementation issues

We combined best features from state-of-the-art algorithms implementing a depth-first strategy for frequent itemset mining. In particular, we use a support counting technique similar to the one presented in [1], and a compact storage of the rows matched by an itemsets in a prefix-tree structure as described in [6]. Additionally, in order to avoid the multiple generation of the same candidate we used an enumeration process inspired by [4].

The specificity of our implementation lies in the disjunction-freeness test. For a set X, this test is performed as follows. We consider all pairs of items A and B in X, and compute $GenSup(\mathcal{M}(r,X\setminus\{A,B\}),X\cup\{\overline{A},\overline{B}\}\setminus\{A,B\})$. If this support is equal to zero for one of the pairs, then by Lemma 1, X is not disjunction-free.

5. EXPERIMENTS

We compare the extraction of the disjunction-free sets to the extraction of the closed sets [9], the other condensed representation for frequent itemsets investigated in the literature and allowing an exact regeneration of their supports.

In this section HLINEX (resp. VLINEX) refers to the implementation of the breadth-first algorithm (resp. depth-first) proposed in this paper to extract the disjunction-free sets. We also used a program called *regen* to generate all frequent closed sets from the disjunction-free sets. This program uses a simple level-wise generation algorithm, which computes the collection of frequent closed sets using the collection of frequent disjunction-free sets and the support-inference relation given in Lemma 3.

To mine efficiently the closed sets we use two algorithms proposed recently: a breadth-first algorithm called CLOSE [9] and a depth-first algorithm called CLOSET [10]. We have implemented these algorithms as described by their authors. We notice however that the implementations of all algorithms (including HLINEX and VLINEX) use the same low-level data structures and techniques, in order to ensure a fair comparison. All prototypes have been implemented in C++, and a similar effort has been spent on specific fine-tuning of each of them.

We have run all experiments on a PC with 128 MB of memory and a 350 MHz Pentium II processor under Linux operating system.

We report experiments on three different commonly used data sets: *Mushroom* (characteristics of some mushroom species), *Connect-4* (collection of game-related state information), and *Pumsb* (a PUMS census data). All these data sets have been preprocessed by researchers from IBM Almaden Research Center[3]. The particularity of the selected data sets is that they are very dense and the combinatorial explosion of the number of frequent itemsets makes the mining of all frequent itemsets together with their supports intractable for low support thresholds [4].

The running times for several support thresholds are given in Figure 1 (note that the axes are logarithmically scaled). It should be noticed that the support thresholds are given as relative frequency thresholds (i.e., 100 × absolute support threshold / total number of rows in the data set). The results are given for HLINEX, VLINEX, the implementations of CLOSE and CLOSET, and also for HLINEX and VLINEX followed by *regen*.

When the curve corresponding to HLINEX+*regen* (resp. VLINEX+*regen*) completely overlaps with the curve corresponding to HLINEX (resp. VLINEX) alone (i.e., the regeneration time is very short) we only give one of the two curves.

The *Mushroom* data set is based on 119 items only and contains 8124 rows - a relatively small number in data mining. Nonetheless, the correlation level between itemsets is high. On this data set, mining our condensed representation is

[3]http://www.almaden.ibm.com/cs/quest/data/long_patterns.bin.tar

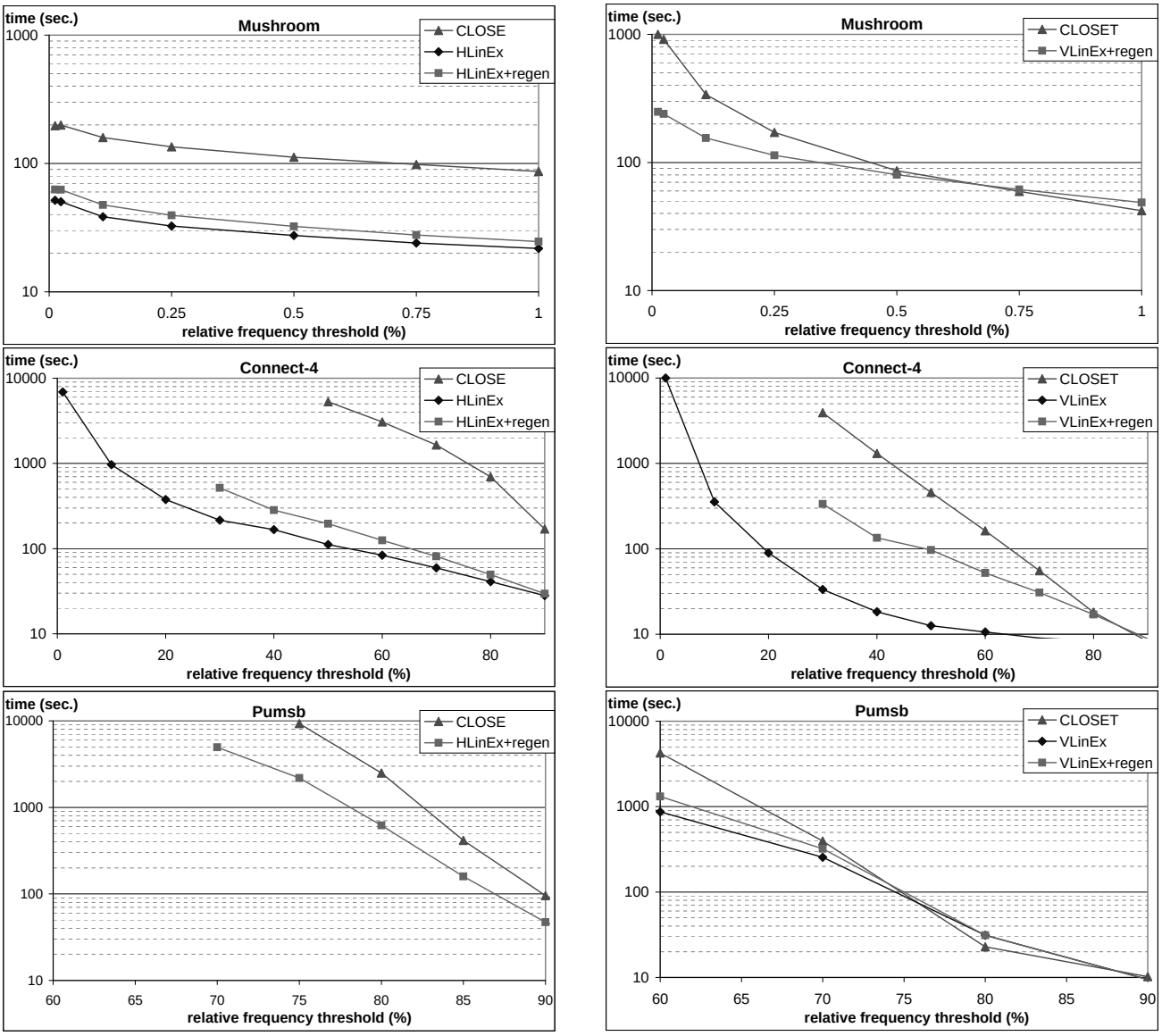

Figure 1: Experiments on the three data sets with breadth-first (left) and depth-first (right) algorithms.

.

advantageous especially using HLinEx (see Figure 1, uppermost graphics).

The *Connect-4* data set is much more difficult. It contains 67557 rows, but a relatively small number of items (129). Its difficulty lies in a very high correlation. Frequent closed sets could not be computed for lower frequency thresholds than 30% - we stopped the corresponding executions after 2 hours (see Figure 1, middle graphics). We compared the direct extraction of closed sets (using CLOSE and CLOSET) to the mining of disjunction-free sets followed by the regeneration of closed sets. We found that the latter is up to 25 times faster, clearly demonstrating the usefulness of our representation.

The last data set, *Pumsb*, contains 49046 rows and is very challenging because of the high number of items (7117). In this case, mining disjunction-free sets was very difficult for HLinEx and VLinEx as for CLOSE and CLOSET. However, HLinEx and VLinEx offer an evident benefit at lower frequency thresholds (see Figure 1, lowermost graphics).

In nearly all experiments, the extraction of the frequent disjunction-free sets is significantly more efficient than the extraction of the frequent closed sets. Moreover, if we add the generation step using *regen* to produce the frequent closed sets from the frequent disjunction-free sets, the overall process remains much more efficient than the direct extraction of the frequent closed sets.

6. CONCLUSION

Knowledge discovery tasks based on frequent set extraction are generally difficult at interesting support thresholds due to the large number of frequent sets. In this paper,

we proposed a condensed representation, called disjunction-free sets, that can be extracted very efficiently in practice and that can be used to regenerate the exact supports of all frequent sets.

We proposed two algorithms to extract this representation, based respectively on a depth-first and a breadth-first strategy.

We also showed that in general disjunction-free sets can be extracted much more efficiently than closed sets [9], the other condensed representation investigated in the literature that allows an exact regeneration of the supports of the frequent sets.

As a future work, we plan to extend this framework towards disjunctive rules having more than two disjuncts in their right hand side (e.g. $Y \Rightarrow A \vee B \vee C$) and to consider the corresponding generalized disjunction-free sets.

7. REFERENCES

[1] R. C. Agarwal, C. C. Aggarwal, and V. V. Prasad. A tree projection algorithm for finding frequent itemsets. *Journal of Parallel and Distributed Computing*, to appear.

[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, 1996.

[3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proc. of the 20th International Conference on Very Large Data Bases (VLDB'94)*, pages 487 – 499, Santiago de Chile, Chile, Sept. 1994.

[4] R. J. Bayardo. Efficiently mining long patterns from databases. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 85–93, Seattle, Washington, USA, June 1998. ACM Press.

[5] J. Boulicaut, A. Bykowski, and C. Rigotti. Approximation of frequency queries by mean of free-sets. In *Proc. of the 4th European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD'00)*, pages 75–85, Lyon, France, Sept. 2000.

[6] 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*, pages 1–12, Dallas, Texas, May 2000.

[7] H. Mannila and H. Toivonen. Multiple uses of frequent sets and condensed representations. In *Proceedings KDD'96*, pages 189–194, Portland, USA, Aug. 1996.

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

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

[10] J. Pei, J. Han, and R. Mao. Closet: An efficient algorithm for mining frequent closed itemsets. In *Proceedings of the 2000 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 21–30, Dallas, Texas, May 2000.

[11] H. Toivonen. Sampling large databases for association rules. In *Proc. of the 22th International Conference on Very Large Data Bases (VLDB'96)*, pages 134 – 145, Bombay, India, Sept. 1996.