Data Min Knowl Disc (2007) 14:171–206
DOI 10.1007/s10618-006-0054-6

Non-derivable itemset mining

Toon Calders · Bart Goethals

Received: 6 December 2005 / Accepted: 23 June 2006 /
Published online: 26 January 2007

Abstract All frequent itemset mining algorithms rely heavily on the monotonicity principle for pruning. This principle allows for excluding candidate itemsets from the expensive counting phase. In this paper, we present sound and complete deduction rules to derive bounds on the support of an itemset. Based on these deduction rules, we construct a condensed representation of all frequent itemsets, by removing those itemsets for which the support can be derived, resulting in the so called *Non-Derivable Itemsets* (NDI) representation. We also present connections between our proposal and recent other proposals for condensed representations of frequent itemsets. Experiments on real-life datasets show the effectiveness of the NDI representation, making the search for frequent non-derivable itemsets a useful and tractable alternative to mining all frequent itemsets.

Keywords Data mining · Itemsets · Condensed representation

Responsible editor: Geoffrey Webb

T. Calders (✉)
Eindhoven Technical University,
Eindhoven, The Netherlands
e-mail: t.calders@tue.nl

T. Calders · B. Goethals
University of Antwerp, Antwerp, Belgium
e-mail: bart.goethals@ua.ac.be

1 Introduction

1.1 Frequent set mining

The *frequent itemset mining problem* Agrawal et al. (1993) is by now well known. We are given a set of items $\mathcal{I}$ and a database $\mathcal{D}$ of pairs (tid, I) with I a subset of $\mathcal{I}$, and *tid* a unique identifier. The elements of $\mathcal{D}$ are called *transactions*. An *itemset* $I \subseteq \mathcal{I}$ is some set of items; its *support* in $\mathcal{D}$, denoted by $\text{supp}(I, \mathcal{D})$, is defined as the number of transactions in $\mathcal{D}$ that contain all items in I. An itemset is called σ*-frequent* in $\mathcal{D}$ if its support in $\mathcal{D}$ is greater than or equal to a given minimum support threshold σ. $\mathcal{D}$ and σ are omitted when they are clear from the context. The goal is now, given a minimal support threshold σ and a database $\mathcal{D}$, to find all σ-frequent itemsets in $\mathcal{D}$. The set of all σ-frequent itemsets in $\mathcal{D}$ is denoted $\mathcal{F}(\mathcal{D}, \sigma)$.

This problem originates from the field of market basket analysis. The items are the products in a supermarket, and every transaction can be seen as a record of the purchases of an individual customer. In this context, frequent itemsets are sets of products that are often purchased together. The scope of frequent itemset mining is broader than only market basket analysis, however. Frequent itemset mining forms the core of many data mining algorithms. Indeed, the identification of items, products, symptoms, characteristics, etc., that often occur together can be seen as one of the most basic tasks in data mining. It is therefore conceivable that theory and algorithms developed for this base case will carry over to other, more sophisticated data mining problems.

1.2 Condensed representations

The search space of the frequent itemset mining problem, all subsets of $\mathcal{I}$, is clearly huge. Instead of generating and counting the supports of all these itemsets at once, several solutions have been proposed to perform a more directed search through all itemsets (Agrawal and Srikant 1994; Zaki 2000; Zaki et al. 1997; Han et al. 2000). These recent studies on frequent itemset mining algorithms resulted in significant performance improvements. In these works, the size of the database and the generation of a reasonable number of frequent itemsets are considered the most costly aspects of frequent itemset mining, and most energy goes into minimizing the number of scans through the database. If the minimal support threshold is set too low, or the data is highly correlated, however, the number of frequent itemsets itself can be prohibitively large. In these situations, no matter how efficient the frequent set mining algorithm is, generating all frequent sets is impossible. To overcome this problem, recently several proposals have been made to construct only a condensed representation of the frequent itemsets (Pasquier et al. 1999; Bastide et al. 2000; Boulicaut et al. 2000; Boulicaut and Bykowski 2000; Pei et al. 2000; Zaki and Hsiao 1999; Bykowski and Rigotti 2003; Kryszkiewicz 2001; Kryszkiewicz and Gajek 2002b,a). A condensed representation only stores a non-redundant cover of all frequent itemsets. In many practical situations this cover is considerably smaller

than the complete collection of all frequent itemsets. Therefore, a condensed representation can be used in those situations where it is impossible to find all frequent itemsets.

1.3 Goal of the paper

The main goal of this paper is to present several methods to identify redundancies in the set of all frequent itemsets and to exploit these redundancies, resulting in a condensed representation of all frequent itemsets and significant performance improvements of a mining operation.

More specifically, we present a complete set of *deduction rules* to derive *tight* intervals on the support of a candidate itemset, given the supports of all its subsets. For example, from $\text{supp}(a) = 5$, $\text{supp}(ab) = 3$, and $\text{supp}(ac) = 4$, we can derive that the support of abc must be in the interval $[2,3]$. The upper bound 3 comes from the monotonicity principle that states that the support of abc is always smaller than the support of ab. The lower bound 2 can be seen as follows: there are exactly 5 transactions that contain a. From these 5 transactions, 3 contain b, and 4 contain c. Therefore, there must be an overlap of at least 2 transactions that contain a, b, and c; that is, the support of abc is at least 2. In this paper, we present a systematic approach to such deductions of bounds on the support of an itemset, resulting in a complete set of deduction rules, with which bounds on the support of an itemset are computed. When the lower bound equals the upper bound, we call an itemset *derivable* as its support can be uniquely derived from the supports of its subsets. We claim that in many applications, derivable itemsets are uninteresting as they represent no new information given their subsets; their supports are as they are for purely combinatorial reasons, and hence, do not represent any interesting regularity or pattern in the data, not already represented by its subsets. Of course, depending on the application, a user might still be interested in derivable itemsets because of other characteristics or interestingness measures. Nevertheless, when the number of frequent itemsets is too large, as is often the case, removing the derivable itemsets might be a reasonable solution.

Then, we show how to construct a *minimal representation* of all frequent itemsets, consisting of those frequent itemsets that are *non-derivable*, and we present an algorithm that efficiently does so. Additionally, we present an efficient method to find the exact support of all derivable frequent itemsets, *without scanning the database*.

Different algorithms for finding non-derivable itemsets are presented and compared empirically. We also compare the non-derivable itemsets representation to other representations, and to mining all frequent itemsets.

1.4 Contributions

We present many results readily reported upon in our previous works (Calders and Goethals 2002; Calders 2003b), and add several new results and properties

together with an in depth study, extensive explanations, many examples and thorough experiments. We also published several other techniques closely related to the material discussed in this paper, and we give clear references to them where necessary (Calders and Goethals 2003; Dexters and Calders 2004; Calders and Goethals 2005a,b; Goethals et al. 2005).

1.5 Outline

In the next section, we explain the deduction rules for deducing the support of itemsets. In Sect. 3, we present the notion of a Non-Derivable Itemset and its properties, after which we give an algorithm to efficiently generate them in Sect. 4. Sect. 5 presents several optimizations and variations of the NDI algorithm. In Sect. 6, a method to find all frequent itemsets from the NDI-representation is discussed. In Sect. 7, we present connections between our proposal and recent proposals for condensed representations. Sect. 8 contains several *experiments* on real-life and synthetic datasets showing the effectiveness of the presented techniques, and Sect. 9 concludes the paper in which we also give some ideas for future work.

2 Deduction rules

A transaction database over a finite set $\mathcal{I}$ is a finite set of pairs (tid, J) with tid a positive integer called the *identifier*, and J a subset of $\mathcal{I}$. The elements of $\mathcal{I}$ are called *items*, and the elements of $\mathcal{D}$, *transactions*. The identifiers must be unique in the database. That is, there can be no two transactions with the same identifier in a transaction database. In all that follows, we implicitly assume that we are working with a transaction database $\mathcal{D}$ over the set of items $\mathcal{I}$.

We will often denote an itemset by the sequence of its items. For example, $\{a, b, c\}$ is denoted abc. Similarly, the union of two itemsets X and Y is often denoted XY, and a set $I \cup \{a, b\}$ as Iab.

2.1 Generalized itemset

Let a be an item. We denote *the negation of* a as $\overline{a}$. A transaction is said to contain $\overline{a}$, if it does not contain a.

Let a *generalized itemset* be a set of items and negations of items. For example, $G = \{a, b, \overline{c}, d\}$ is a generalized itemset. We will often use the notation $X\overline{Y}$ to denote the generalized itemset $X \cup \{\overline{y} \mid y \in Y\}$.

A transaction $T(tid, J)$ *contains a generalized itemset* $G = X\overline{Y}$, denoted $G \subseteq T$, if $X \subseteq J$ and $J \cap Y = \emptyset$.

The *support of a generalized itemset* G *in a database* $\mathcal{D}$, denoted $\text{supp}(G, \mathcal{D})$, is the number of transactions of $\mathcal{D}$ that contain G. $\mathcal{D}$ is omitted when clear from the context.

We say that a generalized itemset $G = X\overline{Y}$ is *based on itemset* I if $I = X \cup Y$.

Example 2.1 Consider the following database:

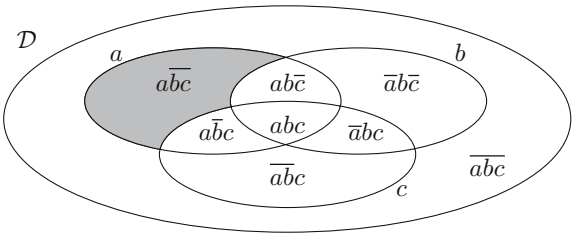
tid	*Items*
1	c
2	a,b
3	a,c
4	a,b,c

There are 8 generalized itemsets based on abc: abc, $ab\overline{c}$, $a\overline{b}c$, $a\overline{bc}$, $\overline{a}bc$, $\overline{a}b\overline{c}$, $\overline{ab}c$, and $\overline{abc}$. The support of $\overline{abc}$ is 0, since there are no transactions that contain a, and do contain neither b, nor c. The support of $\overline{b}c$ is 2, because there are two transactions (1 and 3) that contain c, and do not contain b.

2.2 Derivation of the rules

Before we discuss the derivation rules for support in general, we first illustrate the principle with an example.

Example 2.2 In the next drawing, a transaction database $\mathcal{D}$ over items a, b, c is depicted as a set. The elements of the set are the transactions. The subset marked with a (resp. b, c) consists of all transactions that contain the item a (resp. b, c). In this diagram, the region of the transactions that contain $\overline{abc}$ is filled.



As can be seen in the diagram,

$$\text{supp}(\overline{abc}) = \text{supp}(a) - \text{supp}(ab) - \text{supp}(ac) + \text{supp}(abc).$$

This equality is an instance of the inclusion–exclusion principle Galambos and Simonelli (1996). Since $\text{supp}(\overline{abc}) \geq 0$, the following holds:

$$\text{supp}(abc) \geq \text{supp}(ab) + \text{supp}(ac) - \text{supp}(a).$$

Notice that we can use this inequality to lower bound the support of abc if we know the supports of all its strict subsets.

This procedure can be repeated for every generalized itemset G based on abc; $\text{supp}(G)$ can be written in function of the supports of the subsets of abc. In this equality, $\text{supp}(abc)$ is isolated, and $\text{supp}(G)$ is lower bounded by 0. In

this way, we get 8 different derivation rules for the support of *abc*, one for each generalized itemset based on it.

We now discuss the derivation rules in general. From the inclusion–exclusion principle Galambos and Simonelli (1996), we know that for a given generalized itemset $X\overline{Y}$ based on I,

$$\text{supp}(X\overline{Y}) = \sum_{X \subseteq J \subseteq I} (-1)^{|J \setminus X|} \text{supp}(J). \tag{1}$$

Since the support of a generalized itemset is always larger than or equal to 0, we derive

$$\sum_{X \subseteq J \subseteq I} (-1)^{|J \setminus X|} \text{supp}(J) \geq 0. \tag{2}$$

We now isolate $\text{supp}(I)$ in inequality (2):

$$(-1)^{|I \setminus X|} \text{supp}(I) \geq - \sum_{X \subseteq J \subset I} (-1)^{|J \setminus X|} \text{supp}(J). \tag{3}$$

Since for all $X \subseteq J \subset I$ it holds that

$$- (-1)^{|I \setminus X|} (-1)^{|J \setminus X|} = (-1)^{|I \setminus X| + |J \setminus X| + 1} = (-1)^{|I \setminus J| + 1}, \tag{4}$$

we obtain the following theorem:

Theorem 2.3 *Let $X \subseteq I \subseteq \mathcal{I}$, and $Y = I \setminus X$, then,*

$$\begin{aligned} \text{supp}(I) &\leq \sum_{X \subseteq J \subset I} (-1)^{|I \setminus J| + 1} \text{supp}(J) \qquad \text{if } |I \setminus X| \text{ odd} \\ \text{supp}(I) &\geq \sum_{X \subseteq J \subset I} (-1)^{|I \setminus J| + 1} \text{supp}(J) \qquad \text{if } |I \setminus X| \text{ even} \end{aligned} \tag{5}$$

The rule (5) will be denoted $\mathcal{R}_X(I)$. The bound itself; that is, the sum on the right-hand side of the rule $\mathcal{R}_X(I)$, will be denoted $\delta_X(I)$. Hence,

$$\delta_X(I) =_{def} \sum_{X \subseteq J \subset I} (-1)^{|I \setminus J| + 1} \text{supp}(J).$$

Figure 1 shows all rules for the itemset *abcd*.

The different derivation rules can now be used to derive a bounding interval on the support of an itemset. The greatest lower bound on I will be denoted $\text{LB}(I, \mathcal{D})$, and the least upper bound by $\text{UB}(I, \mathcal{D})$. Thus, the derivation rules bound the support of itemset I within the interval $[\text{LB}(I), \text{UB}(I)]$.

$$
\begin{aligned}
\mathcal{R}_{\emptyset}: \ \mathrm{supp}(abcd) \ &\geq \ \mathrm{supp}(abc) + \mathrm{supp}(abd) + \mathrm{supp}(acd) + \mathrm{supp}(bcd) \\
&\quad - \mathrm{supp}(ab) - \mathrm{supp}(ac) - \mathrm{supp}(ad) \\
&\quad - \mathrm{supp}(bc) - \mathrm{supp}(bd) - \mathrm{supp}(cd) \\
&\quad + \mathrm{supp}(a) + \mathrm{supp}(b) + \mathrm{supp}(c) + \mathrm{supp}(d) \\
&\quad - \mathrm{supp}(\emptyset) \\
\mathcal{R}_{a}: \ \mathrm{supp}(abcd) \ &\leq \ \mathrm{supp}(a) - \mathrm{supp}(ab) - \mathrm{supp}(ac) - \mathrm{supp}(ad) \\
&\quad + \mathrm{supp}(abc) + \mathrm{supp}(abd) + \mathrm{supp}(acd) \\
\mathcal{R}_{b}: \ \mathrm{supp}(abcd) \ &\leq \ \mathrm{supp}(b) - \mathrm{supp}(ab) - \mathrm{supp}(bc) - \mathrm{supp}(bd) \\
&\quad + \mathrm{supp}(abc) + \mathrm{supp}(abd) + \mathrm{supp}(bcd) \\
\mathcal{R}_{c}: \ \mathrm{supp}(abcd) \ &\leq \ \mathrm{supp}(c) - \mathrm{supp}(ac) - \mathrm{supp}(bc) - \mathrm{supp}(cd) \\
&\quad + \mathrm{supp}(abc) + \mathrm{supp}(acd) + \mathrm{supp}(bcd) \\
\mathcal{R}_{d}: \ \mathrm{supp}(abcd) \ &\leq \ \mathrm{supp}(d) - \mathrm{supp}(ad) - \mathrm{supp}(bd) - \mathrm{supp}(cd) \\
&\quad + \mathrm{supp}(abd) + \mathrm{supp}(acd) + \mathrm{supp}(bcd) \\
\mathcal{R}_{ab}: \ \mathrm{supp}(abcd) \ &\geq \ \mathrm{supp}(abc) + \mathrm{supp}(abd) - \mathrm{supp}(ab) \\
\mathcal{R}_{ac}: \ \mathrm{supp}(abcd) \ &\geq \ \mathrm{supp}(abc) + \mathrm{supp}(acd) - \mathrm{supp}(ac) \\
\mathcal{R}_{ad}: \ \mathrm{supp}(abcd) \ &\geq \ \mathrm{supp}(abd) + \mathrm{supp}(acd) - \mathrm{supp}(ad) \\
\mathcal{R}_{bc}: \ \mathrm{supp}(abcd) \ &\geq \ \mathrm{supp}(abc) + \mathrm{supp}(bcd) - \mathrm{supp}(bc) \\
\mathcal{R}_{bd}: \ \mathrm{supp}(abcd) \ &\geq \ \mathrm{supp}(abd) + \mathrm{supp}(bcd) - \mathrm{supp}(bd) \\
\mathcal{R}_{cd}: \ \mathrm{supp}(abcd) \ &\geq \ \mathrm{supp}(acd) + \mathrm{supp}(bcd) - \mathrm{supp}(cd) \\
\mathcal{R}_{abc}: \ \mathrm{supp}(abcd) \ &\leq \ \mathrm{supp}(abc) \\
\mathcal{R}_{abd}: \ \mathrm{supp}(abcd) \ &\leq \ \mathrm{supp}(abd) \\
\mathcal{R}_{acd}: \ \mathrm{supp}(abcd) \ &\leq \ \mathrm{supp}(acd) \\
\mathcal{R}_{bcd}: \ \mathrm{supp}(abcd) \ &\leq \ \mathrm{supp}(bcd) \\
\mathcal{R}_{abcd}: \ \mathrm{supp}(abcd) \ &\geq \ 0
\end{aligned}
$$

Fig. 1 Tight bounds on supp(*abcd*)

Example 2.4 Consider the following database:

tid	*Items*
1	a
2	b
3	c
4	a,b
5	a,c
6	b,c
7	a,b,c

$$
\begin{aligned}
\mathrm{supp}(abc) &\geq 0 \\
&\leq \mathrm{supp}(ab) = 2 \\
&\leq \mathrm{supp}(ac) = 2 \\
&\leq \mathrm{supp}(bc) = 2 \\
&\geq \mathrm{supp}(ab) + \mathrm{supp}(ac) - \mathrm{supp}(a) = 0 \\
&\geq \mathrm{supp}(ab) + \mathrm{supp}(bc) - \mathrm{supp}(b) = 0 \\
&\geq \mathrm{supp}(ac) + \mathrm{supp}(bc) - \mathrm{supp}(c) = 0 \\
&\leq \mathrm{supp}(ab) + \mathrm{supp}(ac) + \mathrm{supp}(bc) \\
&\quad - \mathrm{supp}(a) - \mathrm{supp}(b) - \mathrm{supp}(c) \\
&\quad + \mathrm{supp}(\emptyset) = 1
\end{aligned}
$$

The rules above are the rules $\mathcal{R}_X(abc)$ for X, respectively, $abc, ab, ac, bc, a, b, c, \emptyset$. With the deduction rules we thus derive that the support of *abc* must be included in the interval $[0, 1]$.

Notice that, based on the derivation (1)→(5), it is easy to see that the difference between the bound $\delta_X(I)$ and the actual support of I is given by the following proposition:

Proposition 2.5 *Let $X \subseteq I$ be itemsets, and $Y = I \setminus X$.*

$$|\text{supp}(I) - \delta_X(I)| \;=\; \text{supp}(X\overline{Y}).$$

Example 2.6 We continue Example 2.4. The difference between the lower bound $\delta_a(abc) \;=\; \text{supp}(ab) + \text{supp}(ac) - \text{supp}(a)$, and the support of abc is 1, which is indeed exactly the support of the generalized itemset $a\overline{bc}$. Similarly, the bound $\delta_\emptyset(abc)$ equals the support of abc, and thus, $\text{supp}(\overline{abc}) = 0$.

An important question now is whether the bounds are non-redundant. The next corollary answers this question positively; for every derivation rule, there exists a database for which only this rule gives the best bound. Therefore, omission of the rule will result in a less tight interval.

Corollary 2.7 *None of the rules $\mathcal{R}_X(I)$ is redundant. For all itemsets $X \subseteq I$ there exists a database $\mathcal{D}$ such that $\mathcal{R}_X(I)$ gives the unique best approximation for the support of I.*

Proof According to Proposition 2.5 the difference $|\text{supp}(I) - \delta_X(I)|$ is given by $\text{supp}(X\overline{Y})$, with $Y = I \setminus X$. Consider now the database that contains one transaction $T(tid, X')$ for every subset $X' \neq X$ of I. Hence, for every subset $X' \neq X$ of I, $\text{supp}(X'\overline{Y'}) = 1$, with $Y' = I \setminus X'$, and $\text{supp}(X\overline{Y}) = 0$. Therefore, using Proposition 2.5, we get that $\mathcal{R}_X(I)$ gives the exact bound, while all other rules $\mathcal{R}_{X'}(I)$ are 1 off. □

2.3 Completeness of the rules

If for each subset $J \subset I$, the support is given, then the rules $\mathcal{R}_X(I)$ provide bounds on the support of I. But how good can we expect the bounds to be? Can they still be improved? The next theorem answers this question negatively; the bounds derived by the rules are the best ones possible. We show that the bounds $[\text{LB}(I), \text{UB}(I)]$ on the support of I are always *tight*.

Theorem 2.8 Calders (2003b) *For every itemset $I \subseteq \mathcal{I}$, the rules $\{\mathcal{R}_X(I) \mid X \subseteq I\}$ are sound and complete for deducing the tight lower and upper bound on the support of I in $\mathcal{D}$ based on the supports of all strict subsets of I in $\mathcal{D}$.*

Example 2.9 We continue Example 2.4. Based on

$$\begin{array}{llll} \text{supp}(\emptyset) = 7, & \text{supp}(a) = 4, & \text{supp}(b) = 4, & \text{supp}(c) = 4, \\ \text{supp}(ab) = 2, & \text{supp}(ac) = 2, & \text{supp}(bc) = 2, & \end{array}$$

the rules $\mathcal{R}_X(abc)$ allow to derive the bounds $[0, 1]$ on the support of abc. These bounds are tight, since this interval cannot be made smaller. Indeed, the database in Example 2.4 shows that $\text{supp}(abc) = 1$ is possible, and the following database shows that also the lower bound $\text{supp}(abc) = 0$ is possible. Thus, the interval $[0, 1]$ cannot be made smaller, as this would result in the exclusion of at least either 0 or 1.

tid	*Items*	*tid*	*Items*
1	a,b	5	b,c
2	a,b	6	b,c
3	a,c	7	
4	a,c		

Theorem 2.8 shows that the interval deduced by the rules cannot be made smaller. This statement, however, does not necessarily mean that every number in the derived interval is possible as support. The next theorem does establishes this fact.

Theorem 2.10 Calders (2003b) *Let I be an itemset, and $\mathcal{D}$ a transaction database. For every integer s in the interval* $[\text{LB}(I, \mathcal{D}), \text{UB}(I, \mathcal{D})]$*, there exists a database $\mathcal{D}'$ such that for all strict subsets J of I,* $\text{supp}(J, \mathcal{D}') = \text{supp}(J, \mathcal{D})$*, and* $\text{supp}(I, \mathcal{D}') = s$.

3 Non-derivable itemsets

Based on the deduction rules, it is possible to generate a summary of the set of frequent itemsets. Suppose that the deduction rules allow to deduce the support of an itemset I *exactly*, i.e. $\text{LB}(I, \mathcal{D}) = \text{UB}(I, \mathcal{D})$. Then, there is no need to explicitly count the support of I which requires a complete database scan over $\mathcal{D}$. Indeed, if we need the support of I, we can simply derive it using the deduction rules. We call all itemsets, of which we can perfectly derive the support, *Derivable Itemsets in $\mathcal{D}$* (DI), all other itemsets are called *Non-Derivable Itemsets in $\mathcal{D}$* (NDIs). The set of NDIs is denoted NDI. We show in this section that the set of frequent NDIs allows to compute the supports of all other frequent itemsets, and as such, forms a *condensed representation* Mannila and Toivonen (1996) of the frequent itemsets. To prove this result, we first present some important properties of the NDIs.

3.1 Properties of NDIs

First, we show that the width of the intervals computed by the deduction rules, halve every loop. That is, given an itemset I and an item a not in I, the width of the interval for Ia will be at most half of the width of the interval for I.

Theorem 3.1 *Let $I \subseteq \mathcal{I}$ be an itemset, and $a \in \mathcal{I} \setminus I$ an item. Then*

$$(\mathrm{UB}(Ia, \mathcal{D}) - \mathrm{LB}(Ia, \mathcal{D})) \leq \frac{1}{2}(\mathrm{UB}(I, \mathcal{D}) - \mathrm{LB}(I, \mathcal{D}))$$

Proof Let $X \subseteq I$, $Y = I \setminus X$, and let a be an item not in I. The proof is based on

$$\mathrm{supp}(X\overline{Y}) = \mathrm{supp}(Xa\overline{Y}) + \mathrm{supp}(X\overline{Ya}).$$

From Proposition 2.5 we know that $\mathrm{supp}(X\overline{Y})$, with $Y = I \setminus X$, is the difference between the bound $\delta_X(I)$ computed by $\mathcal{R}_X(I)$ and the actual support of I. Let now $X \subseteq I$ be such that $|\mathrm{supp}(I) - \delta_X(I)|$ is minimized. Then, the width of the interval $[\mathrm{LB}(I), \mathrm{UB}(I)]$ is at least $2 \cdot \mathrm{supp}(X\overline{Y})$. Furthermore, $\mathcal{R}_X(Ia)$ and $\mathcal{R}_{Xa}(Ia)$ are a lower and an upper bound on the support of Ia (if $|Ia \setminus Xa|$ is odd, then $|Ia \setminus X|$ is even and vice versa), and these bounds on Ia differ, respectively, $\mathrm{supp}(Xa\overline{Y})$ and $\mathrm{supp}(X\overline{Ya})$ from the support of Ia in $\mathcal{D}$. When we combine these observations, we get:

$$\begin{aligned}(\mathrm{UB}(Ia) - \mathrm{LB}(Ia)) &\leq \mathrm{supp}(Xa\overline{Y}) + \mathrm{supp}(X\overline{Ya}) \\ &= \mathrm{supp}(X\overline{Y}) \\ &\leq \frac{1}{2}(\mathrm{UB}(I) - \mathrm{LB}(I))\end{aligned}$$ □

Example 3.2 Consider the following database and bounds for ab.

tid	*Items*	*tid*	*Items*
1	a,b	5	b,c
2	a,b	6	b,c
3	a,c	7	
4	c		

$$\begin{aligned}\mathrm{supp}(ab) &\geq 0 \\ &\leq \mathrm{supp}(a) = 3 \\ &\leq \mathrm{supp}(b) = 4 \\ &\geq \mathrm{supp}(a) + \mathrm{supp}(b) \\ &\quad - \mathrm{supp}(\emptyset) = 0\end{aligned}$$

Therefore, for ab we have the interval $[0, 3]$. The upper bound $\delta_a(ab) = 3$ is the closest to the actual support of ab. The difference $|\mathrm{supp}(ab) - \delta_a(ab)|$ is 1. From this, we know that the interval for abc is at most of size 1, because this interval width will be reached when using the bounds $[\delta_a(abc), \delta_{ac}(abc)] = [0, 1]$. As predicted by Theorem 3.1, the width of the interval at least halves going from ab to abc.

Theorem 3.1 gives us the following valuable insights.

Corollary 3.3 *The width of the intervals exponentially shrinks with the size of the itemsets. Hence, every set I with $|I| > \log_2(|\mathcal{D}|) + 1$ must be derivable in $\mathcal{D}$.*

Proof For the singleton sets the bounds are $[0, |\mathcal{D}|]$. Let $I = \{i_1, \ldots, i_n\}$. Because of Theorem 3.1,

$$\begin{aligned} |\mathrm{UB}(I) - \mathrm{LB}(I)| &\leq \frac{1}{2}(\mathrm{UB}(I \setminus \{i_1\}) - \mathrm{LB}(I \setminus \{i_1\})) \\ &\leq \frac{1}{4}(\mathrm{UB}(I \setminus \{i_1, i_2\}) - \mathrm{LB}(I \setminus \{i_1, i_2\})) \\ &\leq \cdots \\ &\leq \frac{1}{2^{n-1}}(\mathrm{UB}(\{i_n\}) - \mathrm{LB}(\{i_n\})) \;=\; \frac{|\mathcal{D}|}{2^{n-1}}. \end{aligned}$$

Thus, if $|I| \geq \log_2(|\mathcal{D}|) + 2$, then

$$(\mathrm{UB}(I) - \mathrm{LB}(I)) \leq \frac{|\mathcal{D}|}{2^{\log_2(|\mathcal{D}|)+1}} \;=\; \frac{1}{2}.$$

Since the bounds are integers, $\mathrm{UB}(I)$ must equal $\mathrm{LB}(I)$. □

This remarkable fact is a strong indication that the number of large NDIs will be very small. This reasoning will be supported by the results of the experiments.

Corollary 3.4 *(Monotonicity) Let $J \subseteq I$ be itemsets. If J is a DI, then I is a DI as well.*

Proof Since J is a DI, by definition, $\mathrm{UB}(J) - \mathrm{LB}(J) = 0$. By Theorem 3.1, adding the elements of $I \setminus J$ to J, halves the interval $|I \setminus J|$ times. Hence, the width of the interval for I must be 0 as well. □

Corollary 3.5 *If $\delta_X(I)$ equals the support of I, then all supersets Ia of I will be DIs, with*

$$\mathrm{supp}(Ia) \;=\; \delta_X(Ia) \;=\; \delta_{Xa}(Ia).$$

Proof The reasoning is very similar to the one used in the proof of Theorem 3.1. If one of the rules $\mathcal{R}_X(I)$ gives the exact support for I, then $\mathrm{supp}(X\overline{Y}) = 0$, $Y = I \setminus X$. Since

$$0 \;=\; \mathrm{supp}(X\overline{Y}) \;=\; \mathrm{supp}(Xa\overline{Y}) + \mathrm{supp}(X\overline{Ya}),$$

this implies

$$\mathrm{supp}(Xa\overline{Y}) \;=\; \mathrm{supp}(X\overline{Ya}) \;=\; 0.$$

Hence, both $\mathcal{R}_X(Ia)$ and $\mathcal{R}_{Xa}(Ia)$ give the exact support of Ia. □

We will use this observation to avoid checking too many rules. More specifically, let I be an NDI, i.e. $\mathrm{LB}(I) \neq \mathrm{UB}(I)$, then, we store these bounds and we count the support of I. After that, if it turns out that $\mathrm{supp}(I) = \mathrm{LB}(I)$ or

$\text{supp}(I) = \text{UB}(I)$, we already know that all supersets of I are derivable, without computing their bounds.

Example 3.6 Consider the following database. Suppose that the set ab is generated as a candidate. The bounds on the support of ab are given on the right.

tid	*Items*	*tid*	*Items*
1	a	5	a,b,c
2	a	6	b
3	a	7	b
4	a,b		

$$\begin{aligned}\text{supp}(ab) &\geq 0\\ &\leq \text{supp}(a) = 5\\ &\leq \text{supp}(b) = 4\\ &\geq \text{supp}(a) + \text{supp}(b)\\ &\quad -\text{supp}(\emptyset) = 2\end{aligned}$$

The bounds for ab are thus $[2, 4]$, and hence ab is non-derivable, and needs to be counted. After we counted ab, however, we see that the support of ab equals the lower bound $\delta_\emptyset(ab) = 2$. Then, according to Corollary 3.5, the superset abc of ab must be derivable, with bounds $[\delta_c(abc), \delta_\emptyset(abc)]$. Indeed,

$$\begin{aligned}\delta_c(abc) &= \text{supp}(ac) + \text{supp}(bc) - \text{supp}(c) = 1\\ \delta_\emptyset(abc) &= \text{supp}(ab) + \text{supp}(ac) + \text{supp}(bc)\\ &\quad -\text{supp}(a) - \text{supp}(b) - \text{supp}(c) + \text{supp}(\emptyset) = 1\end{aligned}$$

The reason for this is that $\text{supp}(ab) = \delta_\emptyset(ab)$ implies that $\text{supp}(\overline{ab}) = 0$. Therefore, also the support of $\overline{abc}$ and $\overline{abc}$ must be zero, and hence $\delta_\emptyset(abc) = \delta_c(abc) = \text{supp}(abc)$.

3.2 Condensed representation based on NDIs

A *Condensed Representation of frequent sets* is, loosely speaking, a subset of $\mathcal{F}$, completed with the supports, that allows for reconstructing $\mathcal{F}$. For example, suppose that $\text{supp}(a) = \text{supp}(ab) = \text{supp}(abc) = 10$, and that the support threshold is 5. Then, a, ab, and abc are in the collection of frequent itemsets. However, it is easy to see that $\text{supp}(abc) = \text{supp}(a) = 10$, implies that ab must be frequent as well, and that its support must equal 10. Therefore, we can leave ab out of the collection, and still have the same information about the frequent itemsets. Hence, a condensed representation of the frequent itemsets is in fact a reduced collection of itemsets that still contains the same information. Notice that Mannila and Toivonen (1996) introduced the notion of a condensed representation in a slightly more general context. There already exist different proposals for condensed representations. The best-known representation is the *closed sets representation* Pasquier et al. (1999). In this representation, if two sets $I \subset J$ have the same support, I is not stored in the condensed representation. Later on in the paper we will give the intuition behind this representation.

Thus, based on a condensed representation, for each itemset I, we must be able to (a) decide whether I is frequent, and (b) if I is frequent, produce its

support. Clearly, from this point of view, a condensed representation needs to be defined with respect to a constructive procedure that performs extraction of supports from representations.

Theorem 3.7 *For every database $\mathcal{D}$, and every support threshold σ, let*

$$\text{NDIRep}(\mathcal{D},\sigma) =_{def} \left\{ (I, \text{supp}(I,\mathcal{D})) \;\middle|\; \begin{array}{l} \text{LB}(I,\mathcal{D}) \neq \text{UB}(I,\mathcal{D}), \\ \text{supp}(I,\mathcal{D}) \geq \sigma \end{array} \right\}.$$

NDIRep$(\mathcal{D},\sigma)$ *is a condensed representation for the frequent itemsets; that is, for each itemset I not in* NDIRep$(\mathcal{D},\sigma)$*, we can decide whether I is frequent, and if I is frequent, we can exactly derive its support from* NDIRep$(\mathcal{D},\sigma)$.

Proof `Base case` $I = \emptyset$ is always an NDI, and hence its support is in the representation NDIRep$(\mathcal{D},\sigma)$, or it is infrequent.

`General case` Suppose we know of each subset J of I whether it is frequent, and if J is frequent, we know supp(J) exact. If one of the subsets is infrequent, I must be infrequent as well. If all subsets are frequent, then we know all their supports. These supports allow us to apply the deduction rules and to derive bounds $[l,u]$ on the support of I. If $l = u$, we know the support of I exactly. If $l \neq u$, then I is an NDI, and thus either I is in NDIRep$(\mathcal{D},\sigma)$, together with its support, or I is infrequent. □

4 The NDI-algorithm

Based on the results in the previous section, Calders and Goethals (2002) have proposed a level-wise algorithm to find all frequent NDIs. Since derivability is monotone, we can prune an itemset if it is derivable. This gives the NDI-algorithm as shown in Algorithm 1. The correctness of the algorithm follows from the results in Theorem 3.1, Corollary 3.3, Corollary 3.4, and Corollary 3.5.

The NDI-algorithm is based on the Apriori-algorithm. We start with the singleton itemsets as the first candidates on line 1. On lines 2–4, we set the bounds for the singleton itemsets to $[0,|\mathcal{D}|]$. For an itemset I, once bounds on it are computed, $I.l$ will hold the lower bound, and $I.u$ the upper bound on its support. In the loop from line 5 to 24, ever larger sets are generated, until no new candidates are generated. C_ℓ holds the candidates for the ℓth loop iteration. Hence, in the ℓth loop iteration, the candidates of size ℓ are counted, in one scan over the database (line 6). In order to make the counting more efficient, index structures like a trie can be used to store the itemsets. After the candidates are counted, F_ℓ holds the frequent ones (line 7). These sets in F_ℓ are all frequent NDIs (line 8). In lines 9–22, the new candidates are generated, starting from F_ℓ. First of all, we will only use the itemsets I in F_ℓ such that the support of I does not equal the lower bound $I.l$ nor the upper bound $I.u$. We call these sets *generating sets*, and they are stored in *Gen* in lines 9–14. Because of Corollary 3.5, every superset of a set in $F_\ell \setminus Gen$ must be derivable. On line 15, we use the standard Apriori candidate generation function to generate new candidates. Since in the

NDI-algorithm we still have to evaluate the bounds, these sets generated by the standard Apriori generation procedure, will be called pre-candidates (line 15). On lines 16–22, the pre-candidates are further pruned using the deduction rules. For every pre-candidate I, the rules are evaluated (line 18). Calders and Goethals (2005b) provide more details on how this rule evaluation can be optimized. If the lower bound does not equal the upper bound, then I is an NDI. Furthermore, if $u < \sigma$, then I is certainly infrequent. Hence, I is a candidate for the next iteration only if $u \geq \sigma$ and $l \neq u$. For the candidates I, the lower and upper bounds are stored, and they are added to $C_{\ell+1}$, the set of candidates for the next iteration (line 20). The loop ends when no new candidates were generated. In that case, NDIRep contains all frequent NDIs.

Algorithm 1 The NDI algorithm

Require: Database $\mathcal{D}$, threshold σ.
Ensure: NDIRep$(\mathcal{D}, \sigma)$.

```
 1: ℓ := 1; NDIRep := {}; C_1 := {{i} | i ∈ I};
 2: for all I in C_1 do
 3:    I.l := 0; I.u := |D|;
 4: end for
 5: while C_ℓ not empty do
 6:    Count the supports of all candidates in C_ℓ in one pass;
 7:    F_ℓ := {I ∈ C_ℓ | supp(I, D) ≥ σ};
 8:    NDIRep := NDIRep ∪ F_ℓ;
 9:    Gen := {};
10:    for all I ∈ F_ℓ do
11:       if supp(I, D) ≠ I.l and supp(I, D) ≠ I.u then
12:          Gen := Gen ∪ {I};
13:       end if
14:    end for
15:    PreC_{ℓ+1} := AprioriGenerate(Gen);
16:    C_{ℓ+1} := {};
17:    for all I ∈ PreC_{ℓ+1} do
18:       Compute bounds [l, u] on the support of I;
19:       if l ≠ u and u ≥ σ then
20:          I.l := l; I.u := u; C_{ℓ+1} := C_{ℓ+1} ∪ {I};
21:       end if
22:    end for
23:    ℓ := ℓ + 1;
24: end while
```

AprioriGenerate is the standard procedure of the Apriori-algorithm to generate new candidates. Thus, the set *AprioriGenerate*(*Gen*) equals $\{I \in \mathcal{I} \mid |I| = i+1, \forall i \in I : I \setminus \{i\} \in Gen\}$.

5 Optimizations

5.1 Limiting the derivation depth

The main disadvantage of the algorithm proposed in the last section is that computing the results of all derivation rules can be very hard for large itemsets.

Fortunately, due to Corollary 3.3, we know that the non-derivable itemsets, for which these rules need to be evaluated, are small themselves.

Nevertheless, we could overcome this problem by restricting to the derivation with rules of limited depth. More specifically, to those rules $\mathcal{R}_X(I)$, where $|I \setminus X|$ is at most a predefined constant k.

We call an itemset I an *NDI for depth k* if its support can be deduced by using rules up to depth k only; i.e., $\mathrm{LB}_k(I) = \mathrm{UB}_k(I)$. The following lemma states that the monotonicity of derivability for depth k still holds.

Lemma 5.1 *Let $I \subseteq J$ be itemsets. If I is derivable at depth k, then J is derivable at depth k as well.*

Proof Since I is derivable for depth k, there must be subsets X and X' of I, with $|I \setminus X|$ and $|I \setminus X'|$ at most k, such that

$$\delta_X(I) \;=\; \delta_{X'}(I) \;=\; \mathrm{supp}(I),$$

one of $|I \setminus X|$ and $|I \setminus X'|$ is even, the other one odd. Without loss of generality we assume that $|I \setminus X| < |I \setminus X'|$. Hence, $|I \setminus X|$ is at most $k-1$. Let now $I \cup \{j\}$ be a superset of I. Because $\delta_X(I)$ equals $\mathrm{supp}(I)$, Corollary 3.5 gives

$$\mathrm{supp}(I \cup \{j\}) \;=\; \delta_X(I \cup \{j\}) \;= \delta_{X \cup \{j\}}(I \cup \{j\}).$$

It is now easy to see that $\delta_X(I \cup \{j\})$ and $\delta_{X \cup \{j\}}(I \cup \{j\})$ are a lower and an upper bound on $\mathrm{supp}(I \cup \{j\})$ of depth at most k that both equal the support of $I \cup \{j\}$. As such, $\mathrm{LB}_k(I \cup \{j\}) = \mathrm{UB}_k(I \cup \{j\})$. □

Unfortunately, one property of NDI's no longer holds for NDI's for depth k. For a non-derivable itemset I, with $\mathrm{supp}(I) = \mathrm{LB}_k(I)$ or $\mathrm{supp}(I) = \mathrm{UB}_k(I)$, it can no longer be derived that all supersets of I are derivable itemsets. The reason for this is simple: suppose I equals a bound $\delta_X(I)$. Then, Corollary 3.5 essentially says that both the bounds $\delta_X(I \cup \{j\})$ and $\delta_{X \cup \{j\}}(I \cup \{j\})$ must equal the support of $I \cup \{j\}$. The bound $\delta_X(I \cup \{j\})$, however, has a depth of one higher than the depth of the bound $\delta_X(I)$ for I. Thus, if the depth of $\delta_X(I)$ is k, the depth of $\delta_X(I \cup \{j\})$ is $k+1$. Because of this, $I \cup \{j\}$ is guaranteed to be derivable for depth $k+1$, but not for depth k. The next example shows that this case can indeed happen in practice.

Example 5.2 Consider the following database $\mathcal{D}$. The table on the right gives the lower and upper bounds for depth 2 and depth ∞, and the supports for the itemsets.

tid	*Items*
1	a
2	b
3	a,b
4	a,c
5	b,c
6	a,b,c

	LB_2	UB_2	LB	UB	*supp*
a	0	6	0	6	4
b	0	6	0	6	4
c	0	6	0	6	3
ab	**2**	4	2	4	**2**
ac	1	3	1	3	2
bc	1	3	1	3	2
abc	1	**2**	1	1	1

Notice that, even though $\text{supp}(ab) = \text{LB}_2(ab)$, *abc* is not derivable for depth 2. For depth 3, however, *abc* s derivable.

Notice that because this property of non-derivable itemsets no longer holds for derivability for depth k, some care is needed with the optimization in line 11 of the NDI-algorithm in Algorithm 1. If we are mining up to depth k, only for sets I with $|I| \leq k-1$, we are sure that the supersets are non-derivable for depth k if the test

$$\textbf{if } \text{supp}(I, \mathcal{D}) \neq I.l \text{ and } \text{supp}(I, \mathcal{D}) \neq I.u \textbf{ then}$$

fails (the maximal depth of a bound for a set I is $|I|$). Therefore, when we restrict the depth of the rules to k, line 11 of the algorithm becomes:

$$\textbf{if } (\text{supp}(I, \mathcal{D}) \neq I.l \text{ and } \text{supp}(I, \mathcal{D}) \neq I.u) \text{ or } (|I| \geq k) \textbf{ then}$$

5.2 Halving intervals at minimal cost

A disadvantage of limiting the derivation depth is that we lose the guarantee that the interval size halves in each step. That is, Corollary 3.3 no longer holds in this case, since not all rules are evaluated anymore, and thus, it is no longer guaranteed that the rule that gives the best bound is computed.

Another solution, however, can still maintain the halving of the interval sizes while only evaluating two rules per itemset. The procedure is based on Proposition 2.5.

First we introduce some notations. Let J be an itemset. We denote the interval on the support of J that can be computed by the deduction rules by $[J.l, J.u]$. Furthermore, we identify the sets $J.X_l$ and $J.X_u$ that index the rules that caused these bounds, that is, the subsets $J.X_l$ and $J.X_u$ of J, such that $\delta_{J.X_l}(J) = J.l$, and $\delta_{J.X_u}(J) = J.u$.

Given a pre-candidate I, we compute bounds on its support as follows: first, we select the subset $J = I \setminus \{i\}$ of I, and the set X, such that $X = J.X_l$, or $X = J.X_u$, and the difference $|\text{supp}(J) - \delta_X(J)|$ is minimal among all subsets J and such sets X. Thus, in fact, among all subsets $J = I \setminus \{i\}$ of I, we select the one with the best bound, together with the set that indexes this bound. Notice

that, according to Proposition 2.5, the difference $|\text{supp}(J) - \delta_X(J)|$ is exactly the support of $X\overline{Y}$, with $Y = J \setminus X$.

The bounds on I are computed with the rules $\mathcal{R}_X(I)$ and $\mathcal{R}_{Xi}(I)$. Notice that, if the first rule is an upper bound, the second one must be a lower bound and vice versa. We assume, without loss of generality, that $\mathcal{R}_{Xi}(I)$ provides an upper bound, and $\mathcal{R}_X(I)$ a lower. Thus, these two rules compute an interval on the support of I. Let $I.l$ be the lower bound, and $I.u$ the upper bound; that is, $I.l = \delta_X(I)$, and $I.u = \delta_{Xi}(I)$. We claim that the size of this interval $[I.l, I.u]$ is at most half the size of the interval of any of its subsets $I \setminus \{i\}$. Indeed, from Proposition 2.5, it follows that

$$|\text{supp}(J) - \delta_X(J)| = \text{supp}(X\overline{Y}).$$

Furthermore,

$$\begin{aligned}\text{supp}(X\overline{Y}) &= \text{supp}(X\overline{Yi}) + \text{supp}(Xi\overline{Y}) \\ &= (\text{supp}(I) - \delta_X(I)) + (\delta_{Xi}(I) - \text{supp}(I)) \\ &= I.u - I.l\end{aligned}$$

Because J and X where selected as to minimize $|\text{supp}(J) - \delta_X(J)|$, every interval of any subset of I has a width of at least $2 \cdot |\text{supp}(J) - \delta_X(J)|$, and thus, $I.u - I.l$ is at most half the size of the interval of any of the subsets $I \setminus \{i\}$

To summarize, the complete procedure is the following:

- Find $J = I \setminus \{i\}$, and $X = J.X_l$ or $X = J.X_u$, such that $|\text{supp}(J) - \delta_X(J)|$ is minimal.
- If $|J \setminus X|$ is even ($\delta_X(J)$ is an upper bound), $I.X_l = X \cup \{i\}$, and $I.X_u = X$, otherwise $I.X_l = X$, and $I.X_u = X \cup \{i\}$.
- $I.l = \delta_{I.X_l}(I)$, $I.u = \delta_{I.X_u}(I)$. Notice that only two rules are evaluated to find an interval for I.
- The support of I is computed in the next iteration (under the assumption that I ends up in $C_{\ell+1}$).
- After the support of I is computed, we do some bookkeeping: $I.X$ is set to $I.X_l$ if $\text{supp}(I) - I.l < I.u - \text{supp}(I)$, otherwise, $I.X$ is set to $I.X_u$. Put otherwise, $I.X$ is the set that provided the best bound for I.

In the algorithm this adaptation results in a modification of step 18. We replace step 18 with the following steps.

```
18a   % Compute bounds [l,u] on support of I;
18b   Let i := minarg_{i∈I}(|supp(I \ {i}) − δ_{(I\{i}).X}(I \ {i})|)
18c   J := I \ {i};
18d   Calculate l_I and u_I with the rules R_{J.X}(I)
      and R_{J.X∪{i}}(I);
18e   if |I \ J.X| is even then I.X_l := J.X else I.X_l = J.X ∪ {i};
18f   if |I \ J.X| is odd then I.X_u = J.X else I.X_u = J.X ∪ {i};
```

Furthermore, after we counted the support of a set I we have to do some bookkeeping to assign to $I.X$ the right set. This can, for example, be done in the loop 11–13: we add the following lines in the loop 11–13, right after step 12:

12b **if** $\text{supp}(I) - I.l < I.u - \text{supp}(I)$ **then** $I.X := I.X_l$ **else** $I.X := I.X_u$.

6 Generating $\mathcal{F}$ from NDIRep

Often, one is not only interested in the frequent NDIs, but in all frequent itemsets. The procedure *DeriveAll* generates all frequent itemsets together with their supports, starting from the representation NDIRep. In Fig. 2, a straightforward procedure to derive all itemsets has been given. The procedure is basically as follows: as long as there are itemsets I such that every strict subset of I is confirmed to be frequent, and I is not yet confirmed as infrequent, check I. These sets I are said to be *in the border* of the current collection of found frequent itemsets. The border $\mathcal{B}^-(\mathcal{S})$ of a set of itemsets $\mathcal{S}$ is formally defined as follows:

$$\mathcal{B}^-(\mathcal{S}) =_{def} \{I \subseteq \mathcal{I} \mid \forall J \subset I : J \in \mathcal{S} \wedge I \notin \mathcal{S}\}$$

In the procedure in Fig. 2, the set C contains all sets that are not checked yet, and that have every strict subset in $\mathcal{F}$. In (2.4), (2.5), the bounds are computed for every set in $\mathcal{C}$ *in isolation*. Indeed, there is no need to combine the computation of the rules, since no scan over the database is required. If $l \geq \sigma$ in (2.6), then I is certainly frequent. Since every frequent NDI is already in NDIRep, I must be a frequent DI in that case. Also the other direction obtains; if I is a frequent DI, then $l \geq \sigma$. The set *Checked* holds the sets that were generated, and that turned out to be infrequent. In the end, *Checked* will contain exactly the negative border of $\mathcal{F}$. The set *Checked* is maintained to avoid that sets that fail are regenerated and evaluated every loop, over and over again.

The simple procedure presented in Fig. 2 can be improved using Corollary 3.5. If we store, for every DI I, the rule $\mathcal{R}_X(I)$ such that $\delta_X(I) = l = u$, then,

Fig. 2 The procedure *DeriveAll* to generate all frequent sets with their supports from NDIRep

Input: NDIRep$(\mathcal{D}, \sigma)$, threshold σ.
Output: Set $\mathcal{F}^{\text{supp}}(\mathcal{D}, \sigma) = \{(I, \text{supp}(I, \mathcal{D})) \mid I \in \mathcal{F}(\mathcal{D}, \sigma)\}$.

(2.1) *DeriveAll*(NDIRep, σ)
(2.2) $\quad\mathcal{C} := \mathcal{B}^-(\text{NDIRep})$; *Checked* $:= \{\}$;
(2.3) $\quad$**while** $\mathcal{C}$ not empty **do**
(2.4) $\quad\quad$**for all** $I \in \mathcal{C}$ **do**
(2.5) $\quad\quad\quad$Compute bounds $[l, u]$ on the support of I;
(2.6) $\quad\quad\quad$**if** $l \geq \sigma$ **then** $\mathcal{F}^{\text{supp}} := \mathcal{F}^{\text{supp}} \cup \{(I, l)\}$;
(2.7) $\quad\quad$*Checked* $:=$ *Checked* $\cup\, \mathcal{C}$;
(2.8) $\quad\quad\mathcal{C} := \mathcal{B}^-(\text{NDIRep} \cup \mathcal{F}^{\text{supp}}) \setminus$ *Checked*;
(2.9) $\quad$**end while**
(2.10) $\quad$**return** $\mathcal{F}$;

for every superset $I \cup \{a\}$ of I, we only need to check the rule $\mathcal{R}_{X \cup \{a\}}(I \cup \{a\})$; because of Corollary 2, $I \cup \{a\}$ must equal the bound computed by that rule. This improvement requires only a little more bookkeeping, but will result in huge time savings for large frequent DIs. Let $I.X$ be the set such that $l = u = \delta_{I.X}(I)$. We assume that $I.X$ is *nil* if I is not a DI. In the procedure in Fig. 2 step (2.5) is replaced by the following steps:

(2.5a) % *Calculate bounds on support (I)*
(2.5b) *parent_di* := *false*;
(2.5c) **for all** $i \in I$ **do**
(2.5d) Let $J = I \setminus \{i\}$;
(2.5d) **if** $J.X \neq nil$ **then**
(2.5e) %
(2.5f) $I.X := \{J.X \cup \{i\}\}$
(2.5g) *parent_di* := *true*; **exit for**
(2.5h) **end for**
(2.5i) **if** *parent_di* = *false* **then**
(2.5j) Compute bounds $[l, u]$ on the support of I;
(2.5k) **else**
(2.5l) $l := \delta_{I.rule}(I)$;

Notice also that if our main objective is to mine all frequent itemsets, we can make the *DeriveAll*-procedure even less costly by doing some of the bookkeeping already in the *NDI*-algorithm. In steps 6, 7 of Algorithm 1, all infrequent NDIs in $\mathcal{B}^-$(NDIRep) can be identified, and after step 18, some of the DIs in $\mathcal{B}^-$(NDIRep) can be identified. If we remember the rules that gave the best lower and upper bound in step 20, we can also set $I.X$ for the DIs on line 18, and for the sets that are excluded from generation on line 11.

7 Related work

7.1 Condensed representations

In the literature, there exists already a number of condensed representations for frequent itemsets. The most important ones are the *closed* sets and the *free* sets (or *generators*) representations. Besides the free and the closed sets, there also are the *disjunction-free*, and *generalized disjunction-free* itemsets. Calders and Goethals (2003) give a complete empirical and theoretical comparison of all representations.

Free sets (Boulicaut et al. 2000) or Generators Pasquier et al. 1999; Kryszkiewicz 2001) are itemsets that don't have a subset with the same support as themselves. Boulicaut et al. (2000) have shown that freeness is anti-monotone; every subset of a free set must be free as well. A very useful property shown by Boulicaut et al. (2000) is the following: a rule $I \rightarrow a$ is said *to hold in database* $\mathcal{D}$ if every transaction that contains I, also contains a. A set I is free if and only if no rule $I \setminus \{i\} \rightarrow i$ with $i \in I$ holds.

Boulicaut et al. (2000) have shown that the collection of free sets together with the infrequent sets in its border forms a condensed representation of the frequent sets. Similarly, also the frequent free sets together with the frequent sets in the border form a condensed representation.

Disjunction-free sets (Bykowski and Rigotti 2001, 2003; Kryszkiewicz 2001) and Generalized Disjunction-free sets (Kryszkiewicz and Gajek 2002a,b) A set I is called disjunction-free if there do not exist two items i_1, i_2 in I such that

$$\text{supp}(I) = \text{supp}(I \setminus \{i_1\}) + \text{supp}(I \setminus \{i_2\}) - \text{supp}(I \setminus \{i_1, i_2\}) \tag{6}$$

Again, this equality can be restated as a rule. A *disjunctive rule* $I \setminus \{i_1, i_2\} \rightarrow i_1 \vee i_2$ is said to *hold in a database* $\mathcal{D}$ if every transaction that contains $I \setminus \{i_1, i_2\}$ also contains at least one of i_1 and i_2. The equality (6) now holds if and only if the *disjunctive rule* $I \setminus \{i_1, i_2\} \rightarrow i_1 \vee i_2$ holds. This rule corresponds to $\mathcal{R}_{I \setminus \{i_1, i_2\}}(I)$. Notice that free sets are a special case of the disjunctive-free sets, namely with $i_1 = i_2$. The disjunction-free sets representation consists of the collection of frequent disjunction-free sets, together with a part of its border.

The generalized disjunction-free sets are a generalization, where the number of disjuncts is unlimited. A set I is said to be *generalized disjunction-free* if there does not exist a subset Y of I such that the rule $I \setminus Y \rightarrow \bigvee Y$ holds. Kryszkiewicz and Gajek (2002b) have shown that the rule $I \setminus Y \rightarrow \bigvee Y$ holds if and only if the equality $\text{supp}(I) = \delta_{I \setminus Y}(I)$ does. Therefore, the generalized disjunctive rules and the NDIs are closely connected. The main difference is in the fact that the generalized disjunction-free sets representation is not based on lower and upper bounds, and that part of the border of the generalized disjunction-free sets needs to be stored as well. Calders and Goethals (2003) have shown that both the generalized disjunction-free representation and the NDI representation can be seen as different instantiations of a common unifying framework.

Closed itemsets (Pasquier et al. 1999). Probably the most well-known representation is the closed itemset representation (Boulicaut and Bykowski 2000; Pei et al. 2000; Zaki and Hsiao 1999). Closed itemsets can be introduced as follows: the *closure* of an itemset I is the largest superset of I such that its support equals the support of I. This superset is unique and is denoted $cl(I)$. An itemset is called *closed* if it equals its closure. Pasquier et al. (1999) have shown that the frequent closed sets form a condensed representation of the frequent itemsets. An alternative definition of the closed sets is as follows (Bastide et al. 2000): two itemsets I and J are called *equivalent* if they are in the exact same set of transactions. This equivalence divides the collection of itemsets into disjoint equivalence classes. Every of these equivalence classes can be characterized by its unique maximal element (w.r.t. set inclusion). These maximal elements are exactly the closed itemsets. Notice incidently that the *minimal* elements of these equivalence classes (which are not unique within the classes), are the free sets.

7.1.1 Relations between the representations

In the experimental section we will compare the different condensed representations on different real-life datasets. We compare the following illustrative set of condensed representations:

1. FreeRep: is based on the free sets. Because the frequent free sets alone do not form a representation, the frequent sets in their border are added to the representation.
2. DFreqRep (Bykowski and Rigotti 2003) and DFreeRep (Kryszkiewicz 2001): are based on the disjunction-free sets. Again, only the frequent disjunction-free sets do not form a representation. The representations DFreqRep and DFreeRep are the frequent disjunction free sets with, respectively, the frequent sets in the border and the free sets in the border.
3. GDFreqRep and GDFreeRep (Kryszkiewicz and Gajek 2002b): similarly as the representations based on the disjunction-free sets, GDFreqRep and GDFreeRep are the frequent generalized disjunction-free sets plus, respectively, the frequent sets in the border and the free sets in the border.
4. ClosedRep (Pasquier et al. 1999): is the closed itemsets representation.
5. NDIRep: is the non-derivable itemsets representation.

Calders (2003a) gives an exhaustive overview of all representations. Notice that the representations FreeRep and GDFreqRep that we propose here are improved variations of existing proposals (Boulicaut et al. 2003; Kryszkiewicz and Gajek 2002b).

Theoretically, the relations between the sizes of the different condensed representations are as follows:

$$\begin{array}{ccccccc} & & |\text{ClosedRep}| & \leq & |\text{FreeRep}| & & \\ |\text{NDIRep}| & \leq & |\text{GDFreqRep}| & \leq & |\text{DFreqRep}| & \leq & |\text{FreeRep}| \\ & & |\text{GDFreeRep}| & \leq & |\text{DFreeRep}| & & \end{array}$$

Calders and Goethals (2003) have given the complete proofs of these relations. Basically, the free sets, the disjunction-free sets and the generalized disjunction-free sets can be seen as variations of the NDI-representation, where the derivation depth is respectively 1, 2, and unbounded.

Example 7.1 Consider the database $\mathcal{D}$ that is given in Fig. 3. In the lattice in Fig. 3, different sets of itemsets are indicated for easy reference. The free sets, disjunction-free sets, and generalized disjunction free sets are indicated with grey boxes. The free sets are a superset of the disjunction-free sets, which are on their turn a superset of the generalized disjunction-free sets. The generalized disjunction free sets are indicated in the darkest shade of grey. Then, the disjunction-free sets that are not generalized disjunction-free in the middle shade, and finally the free sets that are not disjunction-free in the lightest shade. The sets below the horizontal line are non-derivable, the ones above the line are derivable. The closed itemsets are in bold. The curve separates the frequent (below) from the infrequent (top) sets. The minimal support in this example is 3.

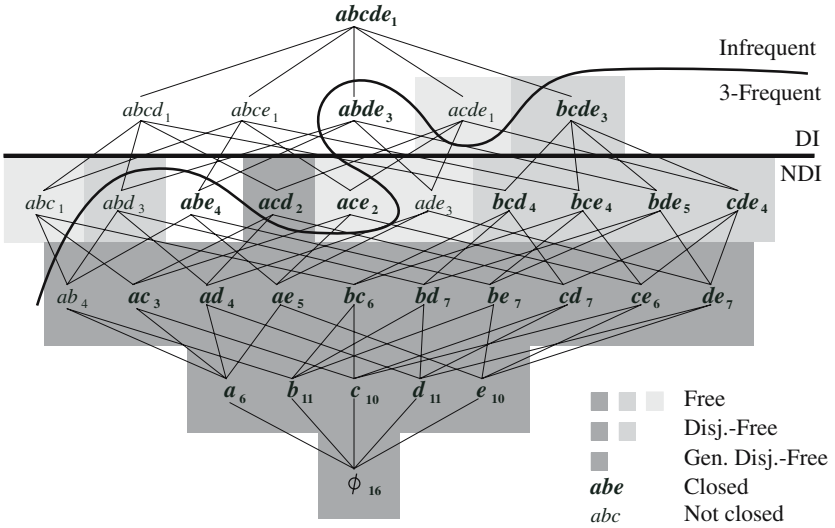


Fig. 3 Free, disjunction-free, generalized disjunction-free, closed, and non-derivable sets

This example clearly illustrates the aforementioned properties of the representations. First of all, the free sets are a superset of the disjunction free sets, which, on their turn are a superset of the generalized disjunction-free sets. Secondly, the frequent free (disjunction-free, generalized disjunction-free) sets are not sufficient to have a condensed representation. For example, both *abe* and *ace* are not frequent generalized-disjunction free. Based on a condensed representation, one should be able to see whether a set is frequent or not. Based on the frequent generalized disjunction-free sets alone, however, it is not possible to differ between these two sets. Therefore, a part of the border has to be added in order to get a representation. In the case of the generalized disjunction free sets, we can choose between the frequent sets in the border (*abd*, *abe*, *ade*, *bcd*, *bce*, *bde*, and *cde*) to get GDFreqRep, or the free sets in the border (*abc*, *abd*, *acd*, *ace*, *ade*, *bcd*, *bce*, *bde*, *cde*) to get GDFreeRep.

Notice that the generalized disjunction-free sets are a subset of the non-derivable itemsets. This is true in general. The representations based on the generalized disjunction-free sets, however, are often larger than the NDI-representation because of the need to add a part of the border. For the GDFreqRep representation, for example, it can even be shown that it is always at least as large as the NDI-representation.

A third important observation is the relation between the free sets and the closed sets. One equivalence class of itemsets appearing in the same set of transactions is the class {*abd*, *ade*, *abde*}. The unique maximal sets in the equivalence classes are the closed sets. For the given class this is *abde*. This class also shows that for the minimal sets in a class, which are the free sets, are not unique (*abd* and *ade*). Therefore, for every closed itemset there are one or more free sets. Furthermore, in the free sets representation, additionally, a border has to be

stored. Therefore, the closed itemsets representation is always smaller than the free sets representation.

7.1.2 Empirical evaluation of the sizes of the condensed representations

In Figure 4, the sizes of the different condensed representations have been given for different support thresholds and datasets. For more information on the datasets, we refer to the experiments section 8.

Except for BMS-Webview-1, the NDI-representation is always among the best representations, together with our new variant of the generalized

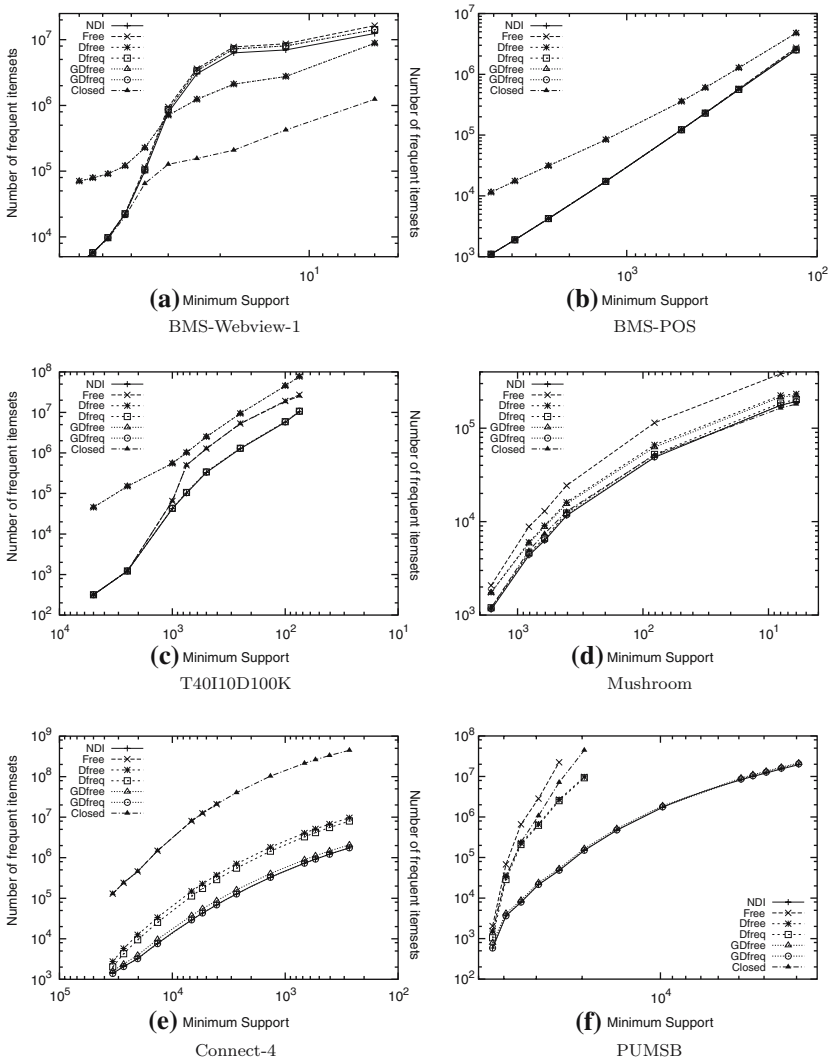


Fig. 4 Size of condensed representations

disjunction free representation, GDFreqRep. Only for the very sparse dataset BMS-Webview-1, the closed itemsets are significantly in smaller numbers than all the other representations. Also GDFreeRep shows some better results here. Nevertheless, GDFreeRep performs significantly worse for the, also sparse, synthetic dataset T40I10D100K than the others. The closed itemsets perform significantly worse on the dense datasets Connect-4 and PUMSB. As shown theoretically, it can be seen that FreeRep is always worse than ClosedRep and DFreqRep. DFreqRep on its turn is worse than GDFreqRep, which is always larger than NDIRep, although these differences are not always very large. Also, DFreeRep is worse than GDFreeRep. The differences between DFreqRep and DFreeRep on the one hand, and GDFreqRep and GDFreeRep on the other hand are small most of the time. Only in the BMS-Webview-1, DFreeRep and GDFreeRep are significantly better for low thresholds, and in the T40I10D100K, DFreqRep and GDFreqRep are better. These differences can be explained by the fact that they store a different part of the border. Calders and Goethals (2003) have given a more in-depth study of these representations.

7.2 Combinatorics

7.2.1 Approximate inclusion–exclusion

Probabilists and statisticians frequently use the inclusion–exclusion bounds to approximate the probability of a union of finitely many events (Jordan 1927; Bonferroni 1936). The inclusion–exclusion principle allows to compute the number of elements in the union of sets $S_1, \ldots, S_n$ given the numbers of elements in all possible intersections $S_{i_1} \cap \ldots \cap S_{i_k}$, $1 \leq i_1, \ldots, i_k \leq n$, $k \leq n$. If for some of these intersections, the number of elements is missing, we can only compute an approximate bound on the size of the union $S_1 \cup \ldots \cup S_n$. It is exactly this type of problems that is studied in *approximate inclusion–exclusion* (Galambos and Simonelli 1996; Kahn et al. 1996; Melkman and Shimony 1997). Melkman and Shimony (1997) have studied the case in which only the count of the number of items in $S_1 \cap \ldots \cap S_n$ is missing. In this case, the bounds on the union $S_1 \cup \ldots \cup S_n$ provide us with bounds on the intersection $S_1 \cap \ldots \cap S_n$. Both problems are alike, and hence many of the results of Melkman and Shimony also apply to our framework. Actually, the completeness and non-redundancy of the inclusion–exclusion rules $\mathcal{R}_X(I)$ for the support of the itemset I are also implicitly proven by Melkman and Shimony.

Bonferroni inequalities are a specific family of combinatorial inequalities for approximate inclusion–exclusion when all intersections up to a fixed constant are known (Galambos and Simonelli 1996). An interesting application of Bonferroni inequalities to data mining is described by Jaroszewicz and Simivici (2002), and Jaroszewicz et al. (2002). Based on the supports of some itemsets, bounds on the supports of arbitrary boolean expressions are computed using these Bonferroni inequalities. The bounds obtained by them are however not tight.

7.2.2 Fréchet bounds

Fréchet bounds (Fréchet 1951) are often used in stochastic processes to estimate an upper and/or a lower bound on the queue length in a queuing system with two different but known marginal inter-arrivals times distributions of two types of customers. The simplest form of the bounds is the following.

$$\max(0, P(a) + P(b) - 1) \leq P(ab) \leq \min(P(a), P(b))$$

The lower bound corresponds to the rule $\mathcal{R}_{\emptyset}(ab)$. The upper bounds are the monotonicity rules $\mathcal{R}_a(ab)$ and $\mathcal{R}_b(ab)$.

7.2.3 Statistical data protection

In statistical databases the privacy of data is studied (Fienberg 1998; Dobra and Fienberg 2000, 2001). In many situations it is common to only provide aggregated data instead of giving the individual data records. An example of this is census data, in which the individual data records are protected, but at the same time aggregated values are published. Fienberg (1998), and Daobra and Fienberg (2000,2001) study the problem of computing sharp upper and lower bounds on the cells of a multi-way contingency table, given a set of marginal tables.

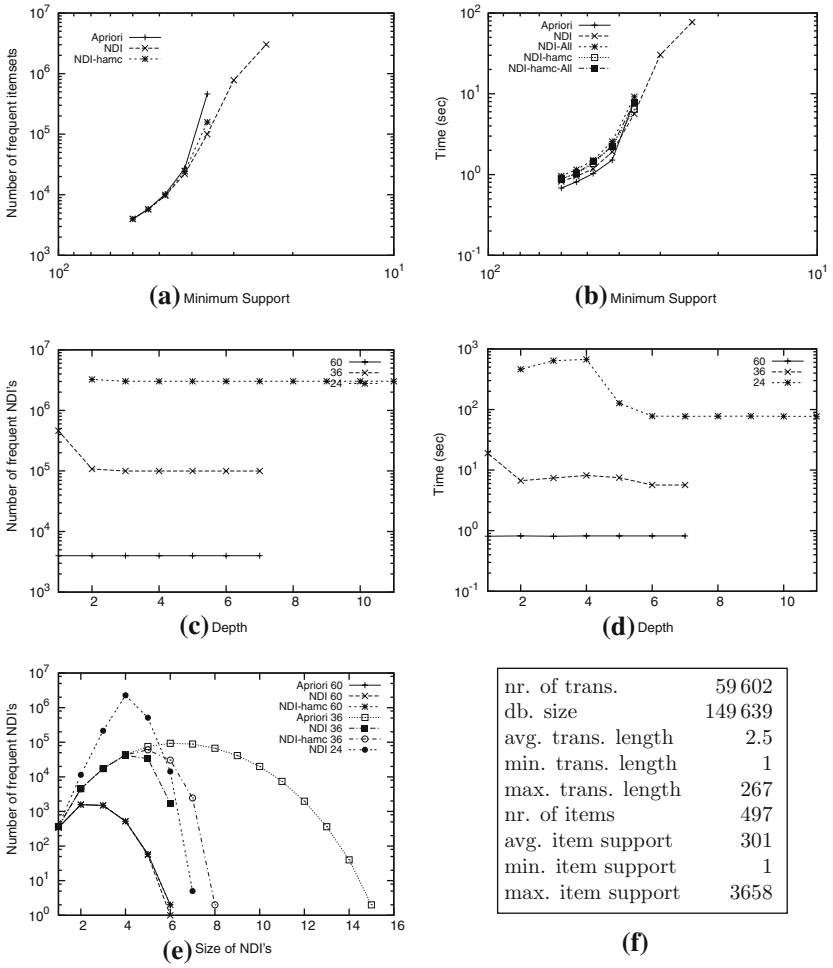
Most relevant for our work are the bounds for entries in dichotomous k-dimensional tables, given all $(k-1)$-dimensional marginals (Dobra 2002: 54). That is, the table has k binary-valued attributes, and the counts of all $(k-1)$-dimensional sub-tables are given. Actually, this case is similar to finding the bounds on the frequency of an itemset of size k, given the frequencies of all itemsets of size $k-1$. For this case, Dobra (2002) derives very similar bounds using a similar proof technique as we do for the non-derivable itemsets.

Most techniques developed in statistical data protection, however, are incomplete, or rely on computationally intensive iterative procedures. Furthermore, in statistical data protection the goal is only to derive the bounds for *one* specific table, where in our NDI-setting, we want to find *all* sets that can be derived completely. Thus, the relation between our work and the work on statistical data protection is restricted to the computation of tight bounds on the support of one itemset.

7.3 Support bounding

In MAXMINER, Bayardo (1998) uses the following rule to derive a lower bound on the support of an itemset:

$$\text{supp}(I \cup J) \geq \text{supp}(I) - \sum_{j \in J} drop(K, j)$$



nr. of trans.	59 602
db. size	149 639
avg. trans. length	2.5
min. trans. length	1
max. trans. length	267
nr. of items	497
avg. item support	301
min. item support	1
max. item support	3658

(f)

Fig. 5 BMS-Webview-1

with $K \subset I$, and $drop(K, j) = \text{supp}(K) - \text{supp}(K \cup \{j\})$. The intuition behind this rule is the following: $drop(K, j)$ expresses how many transactions contain K, but do not contain j. Hence, if we add the item j to the itemset K, the support will decrease. How much the support drops from K to $K \cup \{j\}$, is expressed by $drop(K, j)$. $\sum_{j \in J} drop(K, j)$ is used as an estimate of the drop from I to $I \cup J$. Indeed, for every transaction T that does contain I, but does not contain $I \cup J$, there is at least one $j \in J$ such that T does not contain $K \cup \{j\}$. Bayardo (1998) uses this rule when searching for the frequent itemsets of maximal cardinality. The lower bound is used to jump from I to $I \cup J$ in the search space whenever the lower bound on $I \cup J$ is at least as high as the support threshold.

For $K = I$ and $J = \{i_1, i_2\}$, this rule correspond to $\mathcal{R}_I(I \cup \{i1, i2\})$. In general, the rule used in MAXMINER can be derived from the deduction rules $\mathcal{R}_X(I)$.

For example, let $I = abcd, J = bcd$, and $K = \emptyset$. The MAXMINER rule can be derived from the rules $\mathcal{R}_{\cdot}(abcd)$ as follows:

$$\begin{aligned} &3 \cdot \mathcal{R}_{\emptyset}(abcd) + 2 \cdot \mathcal{R}_{a}(abcd) + 2 \cdot \mathcal{R}_{b}(abcd) + 2 \cdot \mathcal{R}_{c}(abcd) + 2 \cdot \mathcal{R}_{d}(abcd) \\ &\quad + \mathcal{R}_{ab}(abcd) + \mathcal{R}_{ac}(abcd) + \mathcal{R}_{ad}(abcd) + \mathcal{R}_{bc}(abcd) + \mathcal{R}_{bd}(abcd) \\ &\quad + \mathcal{R}_{cd}(abcd) + \mathcal{R}_{abc}(abcd) + \mathcal{R}_{abd}(abcd) + \mathcal{R}_{acd}(abcd) + \mathcal{R}_{bcd}(abcd) \end{aligned}$$

gives the MAXMINER rule

$$\begin{aligned} &\text{supp}(abcd) \geq \\ &\quad \text{supp}(a) + \text{supp}(b) + \text{supp}(c) + \text{supp}(d) - 3 \cdot \text{supp}(\emptyset). \end{aligned}$$

In their PASCAL-algorithm, Bastide et al. (2000) use counting inference to avoid counting the support of all candidates. The rule they are using to avoid counting is based on our rule $\mathcal{R}_{I \setminus \{i\}}(I)$. In fact, the PASCAL-algorithm is the Apriori-algorithm in which the counting of sets derivable with $\mathcal{R}_{I \setminus \{i\}}(I)$ are not counted in the database. Notice that PASCAL can straightforwardly be extended using all rules $\mathcal{R}_X(I)$ for a candidate set I.

Another application of deduction rules, closely related to condensed representations, is developed by Groth and Robertson (2001). Based on the observation that highly frequent items tend to blow up the output of a data mining query by an exponential factor, the authors develop a technique to leave out these highly frequent items, and to reintroduce them after the mining phase by using a deduction rule, called the *multiplicative* rule. The multiplicative rule can be stated as follows: let I, J be itemsets, then

$$\text{supp}(I \cup J, \mathcal{D}) \geq \text{supp}(I, \mathcal{D}) + \text{supp}(J, \mathcal{D}) - \text{supp}(\emptyset, \mathcal{D}).$$

This rule can be derived from the rules in our framework. For $J = \{a, b\}$ for example, the multiplicative rule corresponds to $\mathcal{R}_I(I \cup \{a, b\})$.

Furthermore, Goethals et al. (2005) develops, based on the NDI support bounding technique, a mechanism to prune derivable association rules.

8 Experiments

All experiments were performed on a 3 GHz Pentium IV with 1 Gb of main memory. To empirically evaluate the proposed algorithms and deduction rules, we performed several tests on five real-life datasets and one synthetic dataset. These datasets are all well-known benchmarks for frequent itemset mining. The *BMS-Webview* and *BMS-POS* datasets are click-stream data from a small dot-com company that no longer exists. These two datasets were donated to the research community by Blue Martini Software. The *Pumsb*-dataset is based on census data, the *Mushroom* dataset contains characteristics from different species of mushrooms. The *Connect-4* dataset contains different game positions.

All these datasets are available from the FIMI repository[1]. The *T40I10D100K* dataset was generated using the IBM synthetic data generator.

For each of these datasets, we performed five sets of experiments in which we compared the following algorithms:

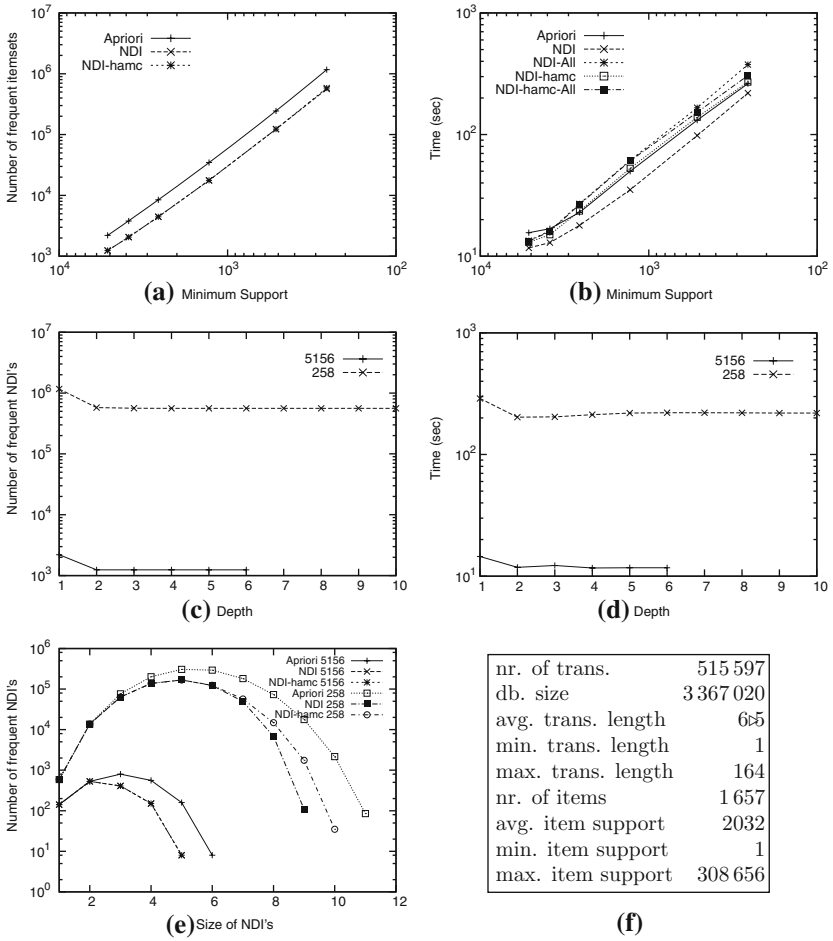
- Apriori: the standard algorithm.
- NDI: as described in Algorithm 1. All derivation rules are evaluated, i.e., the depth is set to ∞.
- NDI-All: the NDI algorithm followed by DeriveAll as described in Fig. 2.
- NDI-hamc: the NDI algorithm using the halving at minimal cost optimization.
- NDI-hamc-All: the NDI-hamc algorithm followed by DeriveAll.

The NDI algorithm is essentially breadth-first, and as such it is an extension of the Apriori algorithm, in which the monotonicity check is replaced with the derivation rules. Therefore, we performed several experiments to illustrate the effect of this replacement. Note, however, there already exist numerous algorithms that outperform Apriori (Goethals and Zaki 2004). Nevertheless, comparing NDI to Apriori nicely shows the improvements solely caused by the deduction rules. For fairness of comparison, all NDI implementations are based on the Apriori implementation.

Every experiment was allowed to run for at most 15 min, and was terminated otherwise. For each dataset, we present the following plots and statistics.

(a) plots the number of frequent itemsets produced by NDI, Apriori and NDI-hamc.
(b) plots the time needed to produce these itemsets. Here, figures are also shown for finding all frequent itemsets with NDI-All and NDI-hamc-All.
(c) shows the number of frequent non-derivable itemsets when only derivation rules of limited depth were allowed to be used for computing the bounds on the itemset supports. This experiment is repeated for three different thresholds, i.e. the highest threshold we used in the experiments for that respective dataset, the lowest threshold for which Apriori was able to finish within time, and the lowest threshold for which NDI was able to finish within time. (Note that the latter two can be the same, as for BMS-POS.) Recall that using depth equal to 1 is equivalent with the number of itemsets found by apriori.
(d) shows the time needed to produce these itemsets for each depth used.
(e) shows the distribution of the number of frequent itemsets w.r.t. the size of the itemsets for apriori and for NDI, again for three different thresholds as before.
(f) shows some simple statistics about the considered database in that Figure. That is, the number of transactions, the size of the database (in number of items), the average, minimum and maximum transaction length, the number of different items, the average, minimum and maximum support of all items.

[1] `http://fimi.cs.helsinki.fi/`



nr. of trans.	515 597
db. size	3 367 020
avg. trans. length	6.5
min. trans. length	1
max. trans. length	164
nr. of items	1 657
avg. item support	2032
min. item support	1
max. item support	308 656

(f)

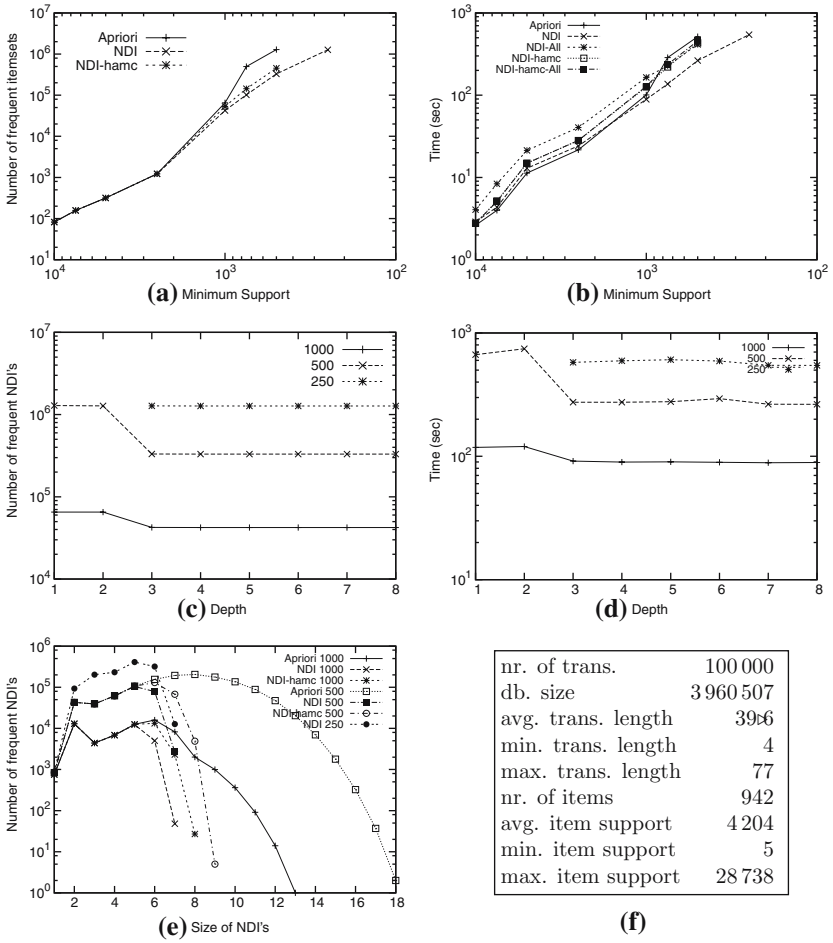
Fig. 6 BMS-POS

Note that all figures have a logarithmically scaled *y*-axis.

8.1 The number of NDI's

For high minimum support values, the number of frequent non-derivable itemsets is not much different from the total number of frequent itemsets for the BMS-Webview-1, BMS-POS and T40I10D100K datasets (Figs. 6(a), 7(a) and 8(a)). This is mainly due to the small sizes of the discovered itemsets. Indeed, small itemsets are not likely to be derivable, as is also illustrated in (e).

When the threshold becomes small enough, however, the number of frequent itemsets rises very quickly, while the number of frequent non-derivable itemsets remains manageable. Again, this is mainly due to the itemset sizes. As explained before, larger itemsets are very likely to be derivable (cfr. Corollary 3.3).



nr. of trans.	100 000
db. size	3 960 507
avg. trans. length	39.6
min. trans. length	4
max. trans. length	77
nr. of items	942
avg. item support	4 204
min. item support	5
max. item support	28 738

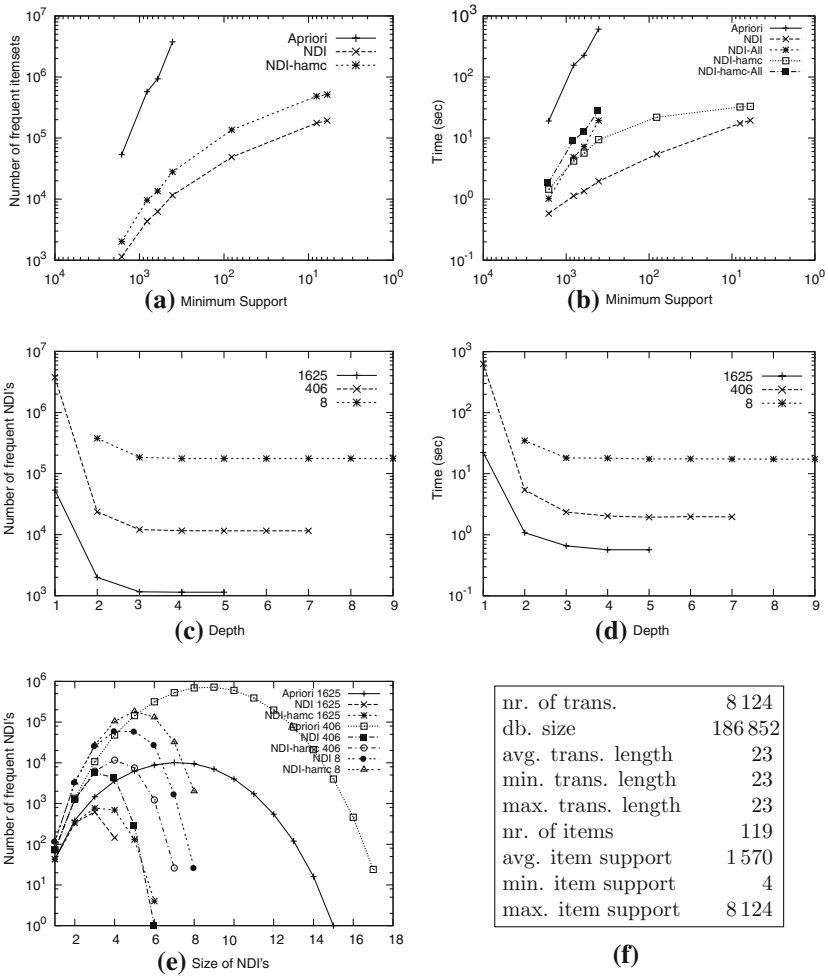
(f)

Fig. 7 T40I10D100K

For the Mushroom, Connect-4, and PUMSB datasets, however, there is a significant difference in the number of NDI's already for high support values. These dense datasets produce very large frequent itemsets, which is known to be a major problem for frequent itemset mining (Goethals and Zaki 2004), while this is exactly what NDI improves upon, as can be seen in Figs. 9(e), 10(e), 5(e).

8.2 The performance of NDI

When comparing NDI and Apriori, one main factor determines the difference in performance. Namely, NDI needs to count the support of significantly less candidate itemsets. On the other hand, the evaluation of the deduction rules causes some overhead. Most of the time, however, this overhead is more than



nr. of trans.	8 124
db. size	186 852
avg. trans. length	23
min. trans. length	23
max. trans. length	23
nr. of items	119
avg. item support	1 570
min. item support	4
max. item support	8 124

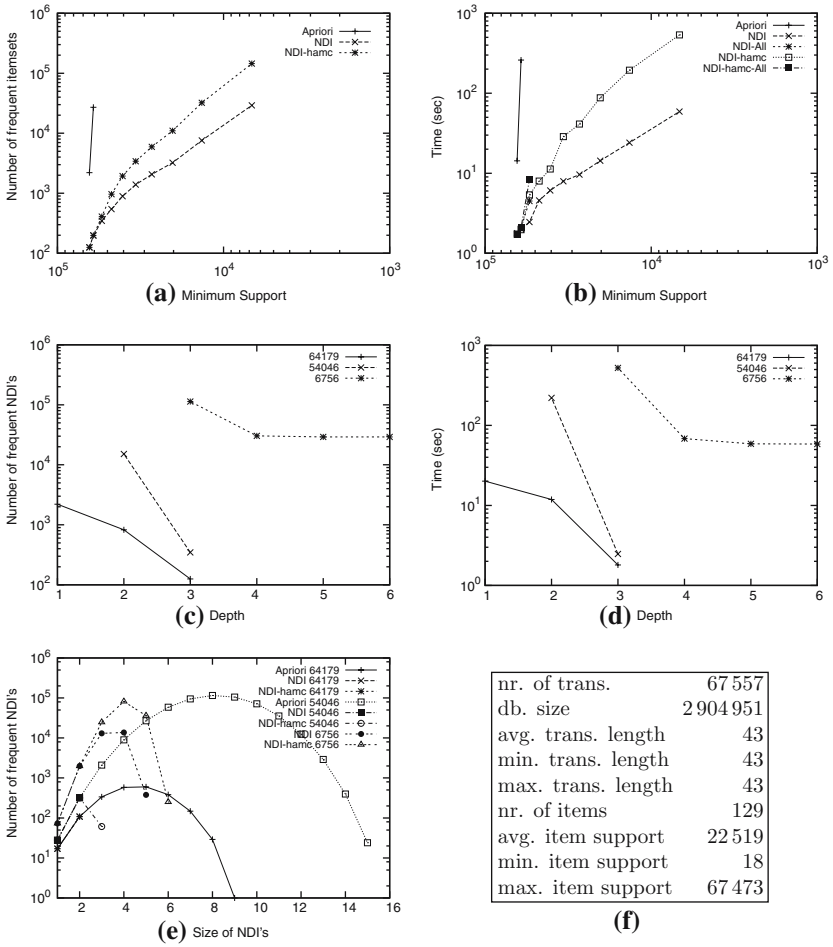
(f)

Fig. 8 Mushroom

compensated by the gain obtained from the reduction in the number of candidate itemsets. Of course, when the number of frequent NDI's is very close to the number of frequent itemsets, NDI will turn out to perform a little slower than Apriori. This can be seen for both the BMS-Webview-1 and T40I10D100K datasets and high minimum support thresholds. For the other datasets or low support thresholds, the performance improvements of NDI over Apriori are impressive.

8.3 The performance of DeriveAll

When all frequent itemsets are derived after generating and counting all non-derivable itemsets, the experimental results show that significant improvements



nr. of trans.	67 557
db. size	2 904 951
avg. trans. length	43
min. trans. length	43
max. trans. length	43
nr. of items	129
avg. item support	22 519
min. item support	18
max. item support	67 473

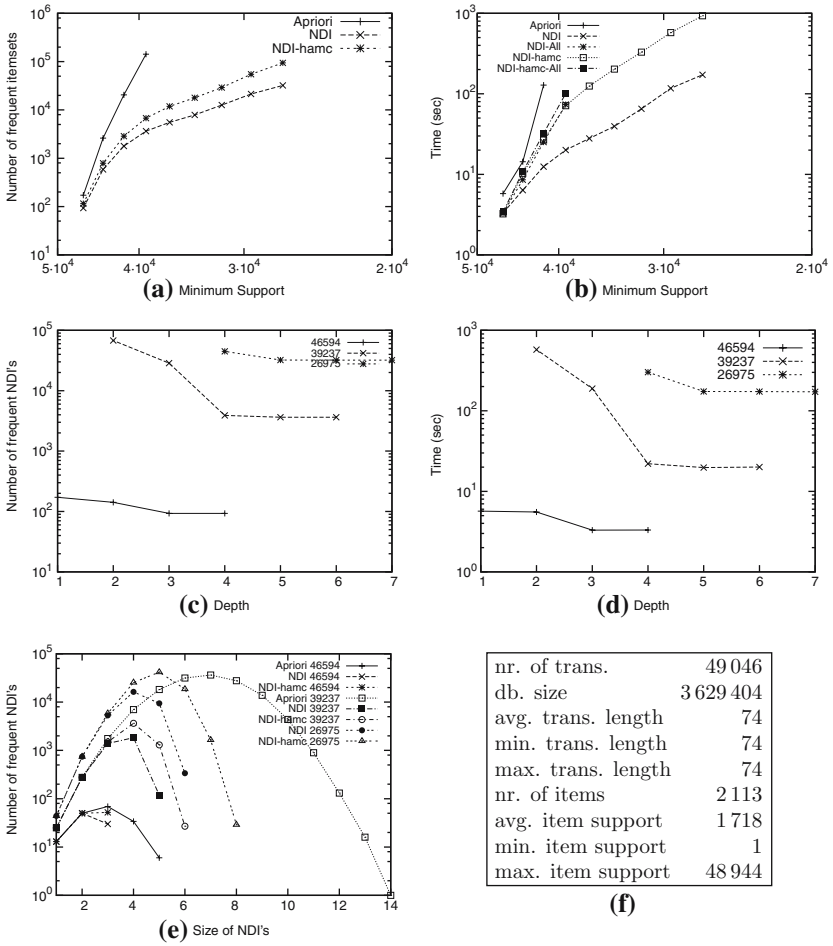
(f)

Fig. 9 Connect-4

can be accomplished w.r.t. Apriori, especially when the difference in the number of itemsets grows. This is clearly due to the benefit of the support derivation techniques over brute force counting in the database. This effect is nicely illustrated on the T40I10D100K dataset where NDI-All starts outperforming Apriori as soon as the number of frequent itemsets grows faster than the number of NDI's (cfr. Figs. 8(a) and 8(b)).

8.4 Limiting the derivation depth

In each Figure, plots (c) and (d) illustrate the effect of limiting the depth of the derivation rules (cfr. Sect. 5.1). Note that points are omitted for the experiments which did not execute within the allowed 15 min or when the derivation depth is larger than the size of the largest candidate itemset.



nr. of trans.	49 046
db. size	3 629 404
avg. trans. length	74
min. trans. length	74
max. trans. length	74
nr. of items	2 113
avg. item support	1 718
min. item support	1
max. item support	48 944

(f)

Fig. 10 PUMSB

Obviously, the overhead created by increasing the depth of the derivation rules is also here most of the time negligible. This is mainly due to the fact that most non-derivable itemsets are small. Indeed, derivation rules of higher depths can only be evaluated for itemsets with a size at least that depth, of which there are only a few (see also plot (e)).

The main cause of the performance improvements is due to the decrease in the number of frequent NDI's of that depth. This effect can be seen for depths up to three or four, for all datasets, for almost all minimum support thresholds.

Figure 6d, however, shows an interesting exception for the smallest threshold. There, the performance improvement can not be attributed to a decrease in the number of frequent NDI's. The reason for this sudden performance increase is due to a significant decrease in the number of candidate itemsets for which the derivation rules need to be evaluated. More specifically, the supersets of a non-derivable itemset can already be pruned when its support equals its lower

or upper bound and *its cardinality is smaller than the derivation depth* (cfr. Sect. 5.1). As Fig. 6e shows, there is a significant peak in the number of non-derivable itemsets of size four, for which this optimization can only be applied starting from derivation depth five.

8.5 Halving intervals at minimal cost

Surprisingly, NDI-hamc is almost always outperformed by NDI. Again, the difference in the number of non-derivable itemsets is the main cause of this. As already explained in the comparison between Apriori and NDI, also here, the overhead created by evaluating all derivation rules is more that compensated by the decrease in the number of non-derivable itemsets.

Even though the total number of itemsets generated by NDI-hamc is larger than that number for NDI, most of the itemsets are still relatively small as compared to the itemsets generated by Apriori. After all, the desired property that the sizes of the derived intervals decrease exponentially for increasing itemset size, still holds.

9 Conclusion

In this paper, we presented non-derivable itemsets (NDI's) as an alternative to mining all itemsets. Starting from a system of sound and complete deduction rules, based on the inclusion–exclusion principle, bounds on the support of an itemset can be derived. The collection of all frequent non-derivable itemsets form a highly condensed representation of the frequent itemsets.

A nice property of the NDI's is that their size is bounded by the logarithm of the database size. This property is a strong indication that the non-derivable itemsets, in general, will not be very large. This indication was indeed also observed in the experiments. Using NDI's allows the use of smaller minimum support thresholds, even for mining dense datasets.

Also the connections between the NDI-representation and different other representations was discussed. We showed how well-known optimization techniques in frequent set mining can be expressed with the deduction rules.

An algorithm to mine all frequent non-derivable itemsets, called NDI, was developed, and thoroughly tested. Although the condensed representation is the main focus of the paper, we also presented an efficient method to derive all frequent itemsets from the NDI-representation. In the experiments, different parameters, such as the depth of the rules to be evaluated, and different optimizations were tried. One of the most important conclusions from the experiments was that the rules of limited depth already provide good bounds. This observation is very important, since it indicates that with the simplest rules, we already obtain significant pruning abilities, making the search for frequent non-derivable itemsets a useful and tractable alternative to mining all frequent itemsets.

There are still some important directions for future work. First of all, for many of the condensed representations, approximate versions have been developed.

E.g., the δ-free sets (Boulicaut et al. 2003) for the free sets, and the condensed frequent-pattern bases (Pei et al. 2004) for the closed sets. It would be interesting to consider approximate representations in the context of non-derivable itemsets as well.

Another direction of future research is the integration of non-derivable itemset mining with depth-first approaches. Obviously, a major problem with this integration is that for the non-derivable itemsets, when considering a candidate, the supports of all subsets are needed, while this is exactly what depth-first approaches try to avoid. Calders and Goethals (2005a) already present some preliminary results on how to solve this.

Acknowledgements Part of this work has been done while Toon Calders was funded by the Fund for Scientific Research — Flanders (FWO-Vlaanderen) as a post-doctoral fellow. This work has been partially funded by the EU contract IQ FP6-516169.

References

Agrawal R, Imilienski T, Swami A (1993) Mining association rules between sets of items in large databases. In: Proc. ACM SIGMOD Int. Conf. Management of Data, Washington, DC, pp 207–216

Agrawal R, Srikant R (1994) Fast algorithms for mining association rules. In: Proc. VLDB Int. Conf. Very Large Data Bases, Santiago, Chile, pp 487–499

Bastide Y, Taouil R, Pasquier N, Stumme G, Lakhal L (2000) Mining frequent patterns with counting inference. SIGKDD Explor 2(2):66–75

Bayardo RJ (1998) Efficiently mining long patterns from databases. In: Proc. ACM SIGMOD Int. Conf. Management of Data, Seattle, Washington, pp 85–93

Bonferroni C (1936) Teoria statistica della classi e calcolo della probabilitá. Publicazioni del R. Instituto Superiore di Scienze Economiche e Commerciali di Firenze 8:1–62

Boulicaut J, Bykowski A, Rigotti C (2003) Free-sets: a condensed representation of boolean data for the approximation of frequency queries. Data Mining Knowledge Discovery 4:5–22

Boulicaut J-F, Bykowski A (2000) Frequent closures as a concise representation for binary data mining. In: Proc. PaKDD Pacific-Asia Conf. on Knowledge Discovery and Data Mining, pp 62–73

Boulicaut, J.-F., A. Bykowski, and C. Rigotti (2000). Approximation of frequency queries by means of free-sets. In *Proc. PKDD Int. Conf. Principles of Data Mining and Knowledge Discovery*, pp. 75–85.

Bykowski A, Rigotti C (2001) A condensed representation to find frequent patterns. In: Proc. PODS Int. Conf. Principles of Database Systems, pp 267–273

Bykowski A, Rigotti C (2003) DBC: a condensed representation of frequent patterns for efficient mining. J Inform Syst 28(8):949–977

Calders T (2003a) Axiomatization and deduction rules for the frequency of itemsets. Ph. D. thesis, University of Antwerp, Belgium

Calders T (2003b) Deducing bounds on the support of itemsets. In: Database technologies for data mining, vol 2682 of LNCS, pp 214–233, Springer

Calders T, Goethals B (2002) Mining all non-derivable frequent itemsets. In: Proc. PKDD Int. Conf. Principles of Data Mining and Knowledge Discovery, pp 74–85. Springer

Calders T, Goethals B (2003) Minimal k-free representations of frequent sets. In: Lavrac N, Gamberger D, Blockeel H, Todorovski L (eds) Proc. PKDD Int. Conf. Principles of Data Mining and Knowledge Discovery, vol 2838 of Lecture Notes in Computer Science, pp 71–82. Springer-Verlag.

Calders T, Goethals B (2005a) Depth-first non-derivable itemset mining. In: Proc. SIAM Int. Conf. on Data Mining

Calders T, Goethals B (2005b) Quick inclusion–exclusion. In: Proceedings ECML-PKDD 2005 Workshop Knowledge Discovery in Inductive Databases, vol 3933 of LNCS, pp 86–103. Springer

Dexters N, Calders T (2004) Theoretical bounds on the size of condensed representations. In: Proceedings ECML-PKDD 2004 Workshop Knowledge Discovery in Inductive Databases, pp 25–36

Dobra A (2002) Statistical tools for disclosure limitation in multi-way contingency tables. Ph. D. thesis, Department of Statistics, Carnegie Mellon University

Dobra A, Fienberg S (2000) Bounds for cell entries in contingency tables given marginal totals and decomposable graphs. Proc Nat Acad Sci 97(22):11885–11892

Dobra A, Fienberg SE (2001) Bounds for cell entries in contingency tables induced by fixed marginal totals. UNECE Stat J 18:363–371

Fienberg SE (1998) Fréchet and bonferroni bounds for multi-way tables of counts with applications to disclosure limitation. In: Statistical data protection (SDP-98), pp 115–129. Eurostat

Fréchet M (1951) Sur les tableaux de correlation dont les marges sont donnés. Ann Univ Lyon Sect A, Series 3 14:53–77

Galambos J, Simonelli I (1996) *Bonferroni-type inequalities with applications*. Springer

Goethals B, Muhonen J, Toivonen H (2005) Nonderivable association rules. In: Proc. SIAM Int. Conf. on Data Mining

Goethals B, Zaki M (2004) Advances in frequent itemset mining implementations: report on fimi'03. SIGKDD Explor Newslett 6(1):109–117

Groth D, Robertson E (2001) Discovering frequent itemsets in the presence of highly frequent items. In: In Proceedings Workshop on Rule Based Data Mining, in Conjunction with the 14th International Conference On Applications of Prolog

Han J, Pei J, Yin Y (2000) Mining frequent patterns without candidate generation. In: Proc. ACM SIGMOD Int. Conf. Management of Data, Dallas, TX, pp 1–12

Jaroszewicz S, Simivici DA (2002) Support approximations using bonferroni-type inequalities. In: Proc. PKDD Int. Conf. Principles of Data Mining and Knowledge Discovery, pp 212–224

Jaroszewicz S, Simivici DA, Rosenberg I (2002) An inclusion-exclusion result for boolean polynomials and its applications in data mining. In: Proc. of the Discrete Mathematics in Data Mining Workshop, SIAM Datamining Conference

Jordan C, (1927) The foundations of the theory of probability. Mat Phys Lapok 34:109–136

Kahn J, Linial N, Samorodnitsky A (1996) Inclusion-exclusion: Exact and approximate. Combinatorica 16:465–477

Kryszkiewicz M (2001) Concise representation of frequent patterns based on disjunction-free generators. In: Proc. IEEE Int. Conf. on Data Mining, pp 305–312

Kryszkiewicz M, Gajek M (2002a) Concise representation of frequent patterns based on generalized disjunction-free generators In: Proc. PaKDD Pacific-Asia Conf. on Knowledge Discovery and Data Mining, pp 159–171

Kryszkiewicz M, Gajek M (2002b) Why to apply generalized disjunction-free generators representation of frequent patterns? In: Proc. International Syposium on Methodologies for Intelligent Systems, pp 382–392

Mannila H, Toivonen H (1996) Multiple uses of frequent sets and condensed representations. In: Proc. KDD Int. Conf. Knowledge Discovery in Databases

Melkman AA, Shimony SE (1997) A note on approximate inclusion-exclusion. Discrete Appl Math 73:23–26

Pasquier N, Bastide Y, Taouil R, Lakhal L (1999) Discovering frequent closed itemsets for association rules. In: Proc. ICDT Int. Conf. Database Theory, pp 398–416

Pei J, Dong G, Zou W, Han J (2004) Mining condensed frequent-pattern bases. Knowl Inf Syst 6(5):570–594

Pei J, Han J, Mao R (2000) Closet: an efficient algorithm for mining frequent closed itemsets. In: ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, Dallas, TX

Zaki M, (2000, May/June). Scalable algorithms for association mining. IEEE Trans Knowledge Data Eng 12(3):372–390

Zaki M, Hsiao C (1999) ChARM: an efficient algorithm for closed association rule mining. In: Technical Report 99-10, Computer Science, Rensselaer Polytechnic Institute

Zaki M, Parthasarathy S, Ogihara M, Li W (1997) New algorithms for fast discovery of association rules. In: Heckerman D, Mannila H, Pregibon D (eds), Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, pp 283–286. AAAI Press