Mining All Non-Derivable Frequent Itemsets

Toon Calders[1]* and Bart Goethals[2]

[1] University of Antwerp, Belgium
[2] University of Limburg, Belgium

Abstract. Recent studies on frequent itemset mining algorithms resulted in significant performance improvements. However, if the minimal support threshold is set too low, or the data is highly correlated, the number of frequent itemsets itself can be prohibitively large. To overcome this problem, recently several proposals have been made to construct a concise representation of the frequent itemsets, instead of mining all frequent itemsets. The main goal of this paper is to identify redundancies in the set of all frequent itemsets and to exploit these redundancies in order to reduce the result of a mining operation. We present deduction rules to derive tight bounds on the support of candidate itemsets. We show how the deduction rules allow for constructing a minimal representation for all frequent itemsets. We also present connections between our proposal and recent proposals for concise representations and we give the results of experiments on real-life datasets that show the effectiveness of the deduction rules. In fact, the experiments even show that in many cases, first mining the concise representation, and then creating the frequent itemsets from this representation outperforms existing frequent set mining algorithms.

1 Introduction

The frequent itemset mining problem [1] is by now well known. We are given a set of items $\mathcal{I}$ and a database $\mathcal{D}$ of subsets of $\mathcal{I}$, together with 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 $support(I, \mathcal{D})$, is defined as the number of transactions in $\mathcal{D}$ that contain all items of I; and an itemset is called *s-frequent* in $\mathcal{D}$ if its support in $\mathcal{D}$ exceeds s. $\mathcal{D}$ and s are omitted when they are clear from the context. The goal is now, given a minimal support threshold and a database, to find all frequent itemsets.

The search space of this 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 patterns. However, this directed search enforces several scans through the database, which brings up another great cost, because these databases tend to be very large, and hence they do not fit into main memory.

* Research Assistant of the Fund for Scientific Research - Flanders (FWO-Vlaanderen).

The standard Apriori algorithm [2] for solving this problem is based on the *monotonicity property*: all supersets of an infrequent itemset must be infrequent. Hence, if an itemset is infrequent, then all of its supersets can be *pruned* from the search-space. An itemset is thus considered potentially frequent, also called a *candidate* itemset, only if all its subsets are already known to be frequent. In every step of the algorithm, all candidate itemsets are generated and their supports are then counted by performing a complete scan of the transaction database. This is repeated until no new candidate itemsets can be generated.

Recent studies on frequent itemset mining algorithms resulted in significant performance improvements. In the early days, the size of the database and the generation of a reasonable amount of frequent itemsets were considered the most costly aspects of frequent itemset mining, and most energy went into minimizing the number of scans through the database. However, if the minimal support threshold is set too low, or the data is highly correlated, the number of frequent itemsets itself can be prohibitively large. To overcome this problem, recently several proposals have been made to construct a concise representation of the frequent itemsets, instead of mining all frequent itemsets [13, 3, 6, 5, 14, 15, 7, 11].

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

1. We present a complete set of *deduction rules* to derive *tight* intervals on the support of candidate itemsets.
2. We show how the deduction rules can be used to construct a *minimal representation* of all frequent itemsets, consisting of all frequent itemsets of which the exact support can not be derived, and present an algorithm that efficiently does so.
3. Also based on these deduction rules, we present an efficient method to find the exact support of all frequent itemsets, that are not in this concise representation, *without scanning the database*.
4. We present *connections* between our proposal and recent proposals for concise representations, such as *free sets* [6], *disjunction-free sets* [7], and *closed sets* [13]. We also show that known tricks to improve performance of frequent itemset mining algorithms, such as used in MAXMINER [4] and PASCAL [3], can be described in our framework.
5. We present several *experiments* on real-life datasets that show the effectiveness of the deduction rules.

The outline of the paper is as follows. In Section 2 we introduce the deduction rules. Section 3 describes how we can use the rules to reduce the set of frequent itemsets. In Section 4 we give an algorithm to efficiently find this reduced set, and in Section 5 we evaluate the algorithm empirically. Related work is discussed in depth in Section 6.

2 Deduction Rules

In all that follows, $\mathcal{I}$ is the set of all items and $\mathcal{D}$ is the transaction database.

We will now describe sound and complete rules for deducing tight bounds on the support of an itemset $I \subseteq \mathcal{I}$, if the supports of all its subsets are given. In order to do this, we will not consider itemsets that are no subset of I, and we can assume that all items in $\mathcal{D}$ are elements of I. Indeed, "projecting away" the other items in a transaction database does not change the supports of the subsets of I.

Definition 1. (I-Projection) *Let $I \subseteq \mathcal{I}$ be an itemset.*

- *The* I-projection of a transaction T, *denoted $\pi_I T$, is defined as $\pi_I T := \{i \mid i \in T \cap I\}$.*
- *The* I-projection of a transaction database $\mathcal{D}$, *denoted $\pi_I \mathcal{D}$, consist of all I-projected transactions from $\mathcal{D}$.*

Lemma 1. *Let I, J be itemsets, such that $I \subseteq J \subseteq \mathcal{I}$. For every transaction database $\mathcal{D}$, the following holds:*

$$support(I, \mathcal{D}) = support(I, \pi_J \mathcal{D}).$$

Before we introduce the deduction rules, we introduce fractions and covers.

Definition 2. (I-Fraction) *Let I, J be itemsets, such that $I \subseteq J \subseteq \mathcal{I}$, the* I-fraction *of $\pi_J \mathcal{D}$, denoted by $f_I^J(\mathcal{D})$ equals the number of transactions in $\pi_J \mathcal{D}$ that exactly consist of the set I.*

If $\mathcal{D}$ is clear from the context, we will write f_I^J, and if $J = \mathcal{I}$, we will write f_I. The support of an itemset I is then $\sum_{I \subseteq I' \subseteq \mathcal{I}} f_{I'}$.

Definition 3. (Cover) *Let $I \subseteq \mathcal{I}$ be an itemset. The* cover *of I in $\mathcal{D}$, denoted by $Cover(I, \mathcal{D})$, consists of all transactions in $\mathcal{D}$ that contain I.*

Again, we will write $Cover(I)$ if $\mathcal{D}$ is clear from the context.

Let $I, J \subseteq \mathcal{I}$ be itemsets, and $J = I \cup \{A_1, \ldots, A_n\}$. Notice that $Cover(J) = \bigcap_{i=1}^{n} Cover(I \cup \{A_i\})$, and that $|\bigcup_{i=1}^{n} Cover(I \cup \{A_i\})| = |Cover(I)| - f_I^J$. From the well-known *inclusion-exclusion principle* [10, p.181] we learn

$$|Cover(I)| - f_I^J = \sum_{1 \leq i \leq n} |Cover(I \cup \{A_i\})| - \sum_{1 \leq i < j \leq n} |Cover(I \cup \{A_i, A_j\})| + \cdots - (-1)^n |Cover(J)|,$$

and since $support(I \cup \{A_{i_1}, \ldots, A_{i_\ell}\}) = |Cover(I \cup \{A_{i_1}, \ldots, A_{i_\ell}\})|$, we obtain

$$(-1)^{|J-I|} support(J) - f_I^J = support(I) - \sum_{1 \leq i \leq n} support(I \cup \{A_i\}) + \sum_{1 \leq i < j \leq n} support(I \cup \{A_i, A_j\}) + \cdots + (-1)^{|J-I|-1} \sum_{1 \leq i \leq n} support(J - \{A_i\})$$

From now on, we will denote the sum on the right-hand side of this last equation by $\sigma(I, J)$.

Since f_I^J is always positive, we obtain the following theorem.

Theorem 1. *For all itemsets $I, J \subseteq \mathcal{I}$, $\sigma(I, J)$ is a lower (upper) bound on $support(J)$ if $|J - I|$ is even (odd). The difference $|support(J) - \sigma(I, J)|$ is given by f_I^J.*

We will refer to the rule involving $\sigma(I, J)$ as $\mathcal{R}_J(I)$ and omit J when clear from the context.

If for each subset $I \subset J$, the support $support(I, \mathcal{D}) = s_I$ is given, then the rules $\mathcal{R}_J(.)$ allow for calculating lower and upper bounds on the support of J. Let l denote the greatest lower bound we can derive with these rules, and u the smallest upper bound we can derive. Since the rules are sound, the support of J must be in the interval $[l, u]$. In [8], we show also that these bounds on the support of J are *tight*; i.e., for every smaller interval $[l', u'] \subset [l, u]$, we can find a database $\mathcal{D}'$ such that for each subset I of J, $support(I, \mathcal{D}') = s_I$, but the support of J is not within $[l', u']$.

Theorem 2. *For all itemsets $I, J \subseteq \mathcal{I}$, the rules $\{\mathcal{R}_J(I) \mid I \subseteq J\}$ are sound and complete for deducing bounds on the support of J based on the supports of all subsets of J.*

The proof of the completeness relies on the fact that for all $I \subseteq J$, we have $support(I, \mathcal{D}) = \sum_{I \subseteq I' \subseteq \mathcal{I}} f_{I'}$. We can consider the linear program consisting of all these equalities, together with the conditions $f_I \geq 0$ for all fractions f_I. The existence of a database $\mathcal{D}'$ that satisfies the given supports is equivalent to the existence of a solution to this linear program in the f_I's and $support(J, \mathcal{D}')$. From this equivalence, tightness of the bounds can be proved. For the details of the proof we refer to [8].

Example 1. Consider the following transaction database.

$$\mathcal{D} = \begin{array}{|l|}\hline A, B, C \\ A, C, D \\ A, B, D \\ C, D \\ B, C, D \\ A, D \\ B, D \\ B, C, D \\ B, C, D \\ A, B, C, D \\ \hline \end{array}$$

$$\begin{array}{llllll} s_A & = 5, & s_B & = 7, & s_C & = 7, \\ s_D & = 9, & s_{AB} & = 3, & s_{AC} & = 3, \\ s_{AD} & = 4, & s_{BC} & = 5, & s_{BD} & = 6, \\ s_{CD} & = 6, & s_{ABC} & = 2, & s_{ABD} & = 2, \\ s_{ACD} & = 2, & s_{BCD} & = 4. & & \end{array}$$

Figure 1 gives the rules to determine tight bounds on the support of $ABCD$. Using these deduction rules, we derive the following bounds on $support(ABCD)$ *without counting in the database.*

Lower bound:	$support(ABCD) \geq 1$	(Rule $\mathcal{R}(AC)$)
Upper bound:	$support(ABCD) \leq 1$	(Rule $\mathcal{R}(A)$)

$$
\begin{cases}
\mathit{support}(ABCD) \geq s_{ABC} + s_{ABD} + s_{ACD} + s_{BCD} - s_{AB} - s_{AC} - s_{AD} & \mathcal{R}_{\{\}} \\
\qquad\qquad -s_{BC} - s_{BD} - s_{CD} + s_A + s_B + s_C + s_D - s_{\{\}} & \\
\mathit{support}(ABCD) \leq s_A - s_{AB} - s_{AC} - s_{AD} + s_{ABC} + s_{ABD} + s_{ACD} & \mathcal{R}_A \\
\mathit{support}(ABCD) \leq s_B - s_{AB} - s_{BC} - s_{BD} + s_{ABC} + s_{ABD} + s_{BCD} & \mathcal{R}_B \\
\mathit{support}(ABCD) \leq s_C - s_{AC} - s_{BC} - s_{CD} + s_{ABC} + s_{ACD} + s_{BCD} & \mathcal{R}_C \\
\mathit{support}(ABCD) \leq s_D - s_{AD} - s_{BD} - s_{CD} + s_{ABD} + s_{ACD} + s_{BCD} & \mathcal{R}_D \\
\mathit{support}(ABCD) \geq s_{ABC} + s_{ABD} - s_{AB} & \mathcal{R}_{AB} \\
\mathit{support}(ABCD) \geq s_{ABC} + s_{ACD} - s_{AC} & \mathcal{R}_{AC} \\
\mathit{support}(ABCD) \geq s_{ABD} + s_{ACD} - s_{AD} & \mathcal{R}_{AD} \\
\mathit{support}(ABCD) \geq s_{ABC} + s_{BCD} - s_{BC} & \mathcal{R}_{BC} \\
\mathit{support}(ABCD) \geq s_{ABD} + s_{BCD} - s_{BD} & \mathcal{R}_{BD} \\
\mathit{support}(ABCD) \geq s_{ACD} + s_{BCD} - s_{CD} & \mathcal{R}_{CD} \\
\mathit{support}(ABCD) \leq s_{ABC} & \mathcal{R}_{ABC} \\
\mathit{support}(ABCD) \leq s_{ABD} & \mathcal{R}_{ABD} \\
\mathit{support}(ABCD) \leq s_{ACD} & \mathcal{R}_{ACD} \\
\mathit{support}(ABCD) \leq s_{BCD} & \mathcal{R}_{BCD} \\
\mathit{support}(ABCD) \geq 0 & \mathcal{R}_{ABCD}
\end{cases}
$$

Fig. 1. Tight bounds on $\mathit{support}(ABCD)$. s_I denotes $\mathit{support}(I)$

Therefore, we can conclude, without having to rescan the database, that the support of $ABCD$ in $\mathcal{D}$ is exactly 1, while a standard monotonicity check would yield an upper bound of 2.

3 Non-Derivable Itemsets as a Concise Representation

Based on the deduction rules, it is possible to generate a summary of the set of frequent itemsets. Indeed, suppose that the deduction rules allow for deducing the support of a frequent itemset I *exactly*, based on the supports of its subsets. Then there is no need to explicitly count the support of I requiring a complete database scan; if we need the support of I, we can always simply derive it using the deduction rules. Such a set I, of which we can perfectly derive the support, will be called a *Derivable Itemset* (DI), all other itemsets are called *Non-Derivable Itemsets* (NDIs). We will show in this section that the set of frequent NDIs allows for computing the supports of all other frequent itemsets, and as such, forms a *concise representation* [12] of the frequent itemsets. To prove this result, we first need to show that when a set I is non-derivable, then also all its subsets are non-derivable. For each set I, let l_I (u_I) denote the lower (upper) bound we can derive using the deduction rules.

Lemma 2. (Monotonicity) *Let $I \subseteq \mathcal{I}$ be an itemset, and $i \in \mathcal{I} - I$ an item. Then $2|u_{I\cup\{i\}} - l_{I\cup\{i\}}| \leq 2\min(|\mathit{support}(I) - l_I|, |\mathit{support}(I) - u_i|) \leq |u_I - l_I|$. In particular, if I is a DI, then also $I \cup \{i\}$ is a DI.*

Proof. The proof is based on the fact that $f_J^I = f_J^{I\cup\{i\}} + f_{J\cup\{I\}}^{I\cup\{i\}}$. From Theorem 1 we know that f_J^I is the difference between the bound calculated by $\mathcal{R}_I(J)$ and

the real support of I. Let now J be such that the rule $\mathcal{R}_I(J)$ calculates the bound that is closest to the support of I. Then, the width of the interval $[l_I, u_I]$ is at least $2f_J^I$. Furthermore, $\mathcal{R}_{I\cup\{i\}}(J)$ and $\mathcal{R}_{I\cup\{i\}}(J\cup\{i\})$ are a lower and an upper bound on the support of $I\cup\{i\}$ (if $|I\cup\{i\}-(J\cup\{i\})|$ is odd, then $|I\cup\{i\}-J|$ is even and vice versa), and these bounds on $I\cup\{i\}$ differ respectively $f_J^{I\cup\{i\}}$ and $f_{J\cup\{I\}}^{I\cup\{i\}}$ from the real support of $I\cup\{i\}$. When we combine all these observations, we get: $u_{I\cup\{i\}} - l_{I\cup\{i\}} \leq f_J^{I\cup\{i\}} + f_{J\cup\{I\}}^{I\cup\{i\}} = f_J^I \leq \frac{1}{2}(u_I - l_I)$. □

This lemma gives us the following valuable insights.

Corollary 1. *The width of the intervals exponentially shrinks with the size of the itemsets.*

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 2. *If I is a NDI, but it turns out that $\mathcal{R}_I(J)$ equals the support of I, then all supersets $I\cup\{i\}$ of I will be a DI, with rules $\mathcal{R}_{I\cup\{i\}}(J)$ and $\mathcal{R}_{I\cup\{i\}}(J\cup\{i\})$.*

We will use this observation to avoid checking all possible rules for $I\cup\{i\}$. This avoidance can be done in the following way: whenever we calculate bounds on the support of an itemset I, we remember the lower and upper bound l_I, u_I. If I is a NDI; i.e., $l_I \neq u_I$, then we will have to count its support. After we counted the support, the tests $\mathit{support}(I) = l_I$ and $\mathit{support}(I) = u_I$ are performed. If one of these two equalities obtains, we know that all supersets of I are derivable, without having to calculate the bounds.

Corollary 3. *If we know that I is a DI, and that rule $\mathcal{R}_I(J)$ gives the exact support of I, then $\mathcal{R}_{I\cup\{i\}}(J\cup\{i\})$ gives the exact support for $I\cup\{i\}$.*

Suppose that we want to build the entire set of frequent itemsets starting from the concise representation. We can then use this observation to improve the performance of deducing all supports. Suppose we need to deduce the support of a set I, and of a superset J of I; instead of trying all rules to find the exact support for J, we know in advance, because we already evaluated I, which rule to choose. Hence, for any itemset which is known to be a DI, we only have to compute a single deduction rule to know its exact support.

From Lemma 2, we easily obtain the following theorem, saying that the set of NDIs is a concise representation. We omit the proof due to space limitations.

Theorem 3. *For every database $\mathcal{D}$, and every support threshold s, let* $\mathrm{NDI}(\mathcal{D}, s)$ *be the following set:*

$$\mathrm{NDI}(\mathcal{D}, s) := \{(I, \mathit{support}(I, \mathcal{D})) \mid l_I \neq u_I\}.$$

$\mathrm{NDI}(\mathcal{D}, s)$ *is a concise representation for the frequent itemsets, and for each itemset J not in* $\mathrm{NDI}(\mathcal{D}, s)$*, we can decide whether J is frequent, and if J is frequent, we can exactly derive its support from the information in* $\mathrm{NDI}(\mathcal{D}, s)$.

4 The NDI-Algorithm

Based on the results in the previous section, we propose 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 below. The correctness of the algorithm follows from the results in Lemma 2.

NDI($\mathcal{D}$,s)
  $i := 1$; NDI := {}; $C_1 := \{\{i\} \mid i \in \mathcal{I}\}$;
  for all I in C_1 **do** $I.l := 0; I.u := |\mathcal{D}|$;
  while C_i not empty **do**
    Count the supports of all candidates in C_i in one pass over $\mathcal{D}$;
    $F_i := \{I \in C_i \mid \mathit{support}(I, \mathcal{D}) \geq s\}$;.
    NDI := NDI $\cup F_i$;
    $Gen := \{\}$;
    for all $I \in F_i$ **do**
      if $\mathit{support}(I) \neq I.l$ and $\mathit{support}(I) \neq I.u$ **then**
        $Gen := Gen \cup \{I\}$;
    $PreC_{i+1} := \mathit{AprioriGenerate}(Gen)$;
    $C_{i+1} := \{\}$;
    for all $J \in PreC_{i+1}$ **do**
      Compute bounds $[l, u]$ on support of J;
      if $l \neq u$ **then** $J.l := l; J.u := u; C_{i+1} := C_{i+1} \cup \{J\}$;
    $i := i + 1$
  end while
  return NDI

Since evaluating all rules can be very cumbersome, in the experiments we show what the effect is of only using a couple of rules. We will say that we use rules *up to depth* k if we only evaluate the rules $\mathcal{R}_J(I)$ for $|I - J| \leq k$. The experiments show that in most cases, the gain of evaluating rules up to depth k instead of up to depth $k - 1$ typically quickly decreases if k increases. Therefore, we can conclude that in practice most pruning is done by the rules of limited depth.

5 Experiments

For our experiments, we implemented an optimized version of the Apriori algorithm and the NDI algorithm described in the previous section. We performed our experiments on several real-life datasets with different characteristics, among which a dataset obtained from a Belgian retail market, which is a sparse dataset of 41 337 transaction over 13 103 items. The second dataset was the BMS-Webview-1 dataset donated by Z. Zheng et al. [16], containing 59 602 transactions over 497 items. The third dataset is the dense census-dataset as available in the UCI KDD repository [9], which we transformed into a transaction database by creating a different item for every attribute-value pair, resulting

in 32 562 transactions over 22 072 items. The results on all these datasets were very similar and we will therefore only describe the results for the latter dataset.

Figure 2 shows the average width of the intervals computed for all candidate itemsets of size k. Naturally, the interval-width of the singleton candidate itemsets is 32 562, and is not shown in the figure. In the second pass of the NDI-algorithm, all candidate itemsets of size 2 are generated and their intervals deduced. As can be seen, the average interval size of most candidate itemsets of size 2 is 377. From then on, the interval sizes decrease exponentially as was predicted by Corollary 1.

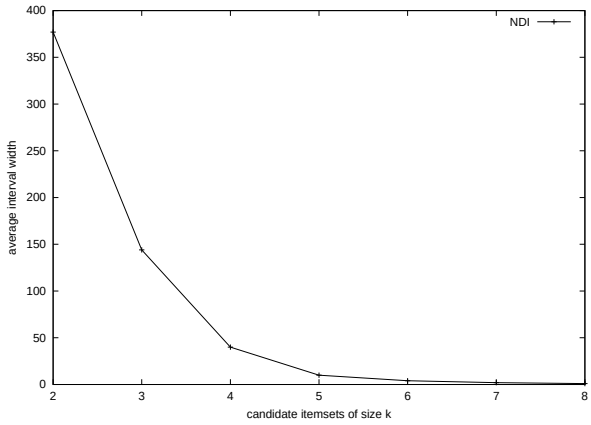

Fig. 2. Average interval-width of candidate itemsets.

Figure 3 shows the size of the concise representation of all NDIs compared to the total number of frequent patterns as generated by Apriori, for varying minimal support thresholds. If this threshold was set to 0.1%, there exist 990 097 frequent patterns of which only 162 821 are non-derivable. Again this shows the theoretical results obtained in the previous sections.

In the last experiment, we compared the strength of evaluating the deduction rules up to a certain depth, and the time needed to generate all NDIs w.r.t. the given depth. Figure 4 shows the results. On the x-axis, we show the depth up to which rules are evaluated. We denoted the standard Apriori monotonicity check by 0, although it is actually equivalent to the rules of depth 1. The reason for this is that we also used the other optimizations described in Section 3. More specifically, if the lower or upper bound of an itemset equals its actual support, we can prune its supersets, which is denoted as depth 1 in this figure. The left y-axis shows the number of NDIs w.r.t. the given depth and is represented by the line 'concise representation'. The line 'NDI' shows the time needed to generate these NDIs. The time is shown on the right y-axis. The 'NDI+DI' line shows the time needed to generate all NDIs plus the time needed to derive all DIs, resulting

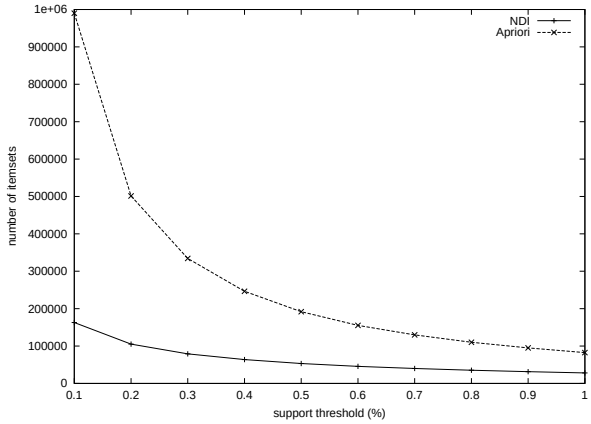

Fig. 3. Size of concise representation.

in all frequent patterns. As can be seen, the size of the concise representation drops quickly only using the rules of depth 1 and 2. From there on, higher depths result in a slight decrease of the number of NDIs. From depth 4 on, this size stays the same, which is not that remarkable since the number of NDIs of these sizes is also small. The time needed to generate these sets is best if the rules are only evaluated up to depth 2. Still, the running time is almost always better than the time needed to generate all frequent itemsets (depth 0), and is hardly higher for higher depths. For higher depths, the needed time increases, which is due to the number of rules that need to be evaluated. Also note that the total time required for generating all NDIs and deriving all DIs is also better than generating all frequent patterns at once, at depth 1,2,and 3. This is due to the fact that the NDI algorithm has to perform less scans through the transaction database. For larger databases this would also happen for the other depths, since the derivation of all DIs requires no scan through the database at all.

6 Related Work

6.1 Concise Representations

In the literature, there exist already a number of concise representations for frequent itemsets. The most important ones are *closed itemsets*, *free itemsets*, and *disjunction-free itemsets*. We compare the different concise representations with the NDI-representation.

Free sets [6] or Generators [11] An itemset I is called *free* if it has no subset with the same support. We will denote the set of all frequent free itemsets with *FreqFree*. In [6], the authors show that freeness is anti-monotone; the subset of a free set must also be free. *FreqFree* itself is not a concise representation for

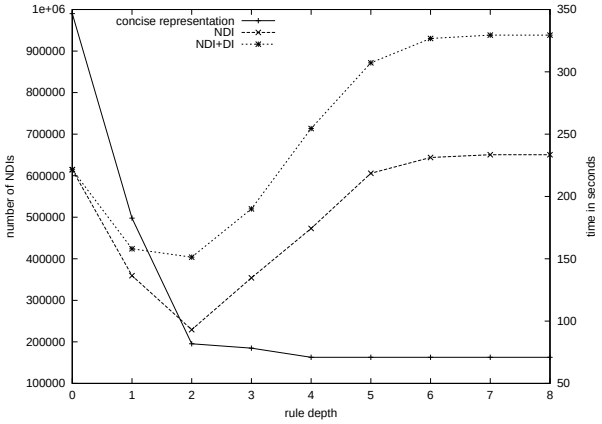

Fig. 4. Strength of deduction rules.

the frequent sets, unless if the set $Border(FreqFree) := \{I \subseteq \mathcal{I} \mid \forall J \subset I : J \in FreqFree \wedge I \notin FreqFree\}$ is added [6]. We call the concise representation consisting of these two sets *ConFreqFree*. Notice that free sets [6] and generators [13, 11] are the same.

Disjunction-free sets [7] or disjunction-free generators [11] *Disjunction-free* sets are essentially an extension of free sets. A set I is called disjunction-free if there does not exist two items i_1, i_2 in I such that $support(I) = support(I - \{i_1\}) + support(I - \{i_2\}) - support(I - \{i_1, i_2\})$. This rule is in fact our rule $\mathcal{R}_I(I - \{i_1, i_2\})$. Notice that free sets are a special case of this case, namely when $i_1 = i_2$. We will denote the set of frequent disjunction-free sets by *FreqDFree*. Again, disjunction-freeness is anti-monotone, and *FreqDFree* is not a concise representation of the set of frequent itemsets, unless we add the border of *FreqDFree*. We call the concise representation containing these two sets *ConFreqDFree*.

Closed itemsets [13] Another type of concise representation that received a lot of attention in the literature [5, 14, 15] are the closed itemsets. They 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 by $cl(I)$. An itemset is called *closed* if it equals its closure. We will denote the set of all frequent closed itemsets by *FreqClosed*. In [13], the authors show that *FreqClosed* is a concise representation for the frequent itemsets.

In the following proposition we give connections between the different concise representations.

Proposition 1. *For every dataset and support threshold, the following inequalities are valid.*

1. *The set of frequent closed itemsets is always smaller or equal in cardinality than the set of frequent free sets.*

2. The set of NDIs is always a subset of ConFreqDFree.

Proof. 1. We first show that $Closed = cl(Free)$.
- $\subseteq$ Let C be a closed set. Let I be a smallest subsets of C such that $cl(I) = C$. Suppose I is not a free set. Then there exist $J \subset I$ such that $support(J) = support(I)$. This rule however implies that $support(J) = support(C)$. This is in contradiction with the minimality of I.
- $\supseteq$ Trivial, since cl is idempotent.

This equality implies that cl is always a surjective function from *Free* to *Closed*, and therefore, $|Free| \geq |Closed|$.

2. Suppose I is not in *ConFreqDFree*. If I is not frequent, then the result is trivially satisfied. Otherwise, this means that I is not a frequent free set, and that there is at least one subset J of I that is also not a frequent free set (otherwise I would be in the border of *FreqDFree*.) Therefore, there exist $i_1, i_2 \in J$ such that $support(J) = support(J - \{i_1\}) + support(J - \{i_2\}) - support(J - \{i_1, i_2\}) = \sigma(J, J - \{i_1, i_2\})$. We now conclude, using Lemma 2, that I is a derivable itemset, and thus not in NDI.

□

Other possible inclusions between the described concise representations do not satisfy, i.e., for some datasets and support thresholds we have $|\text{NDI}| < |Closed|$, while other datasets and support thresholds have $|Closed| < |\text{NDI}|$. We omit the proof of this due to space limitations. We should however mention that even though *FreqDFree* is always a superset of NDI, in the experiments the gain of evaluating the extra rules is often small. In many cases the reduction of *ConFreqDFree*, which corresponds to evaluating rules up to depth 2 in our framework, is almost as big as the reduction using the whole set of rules. Since our rules are complete, this shows that additional gain is in many cases unlikely.

6.2 Counting Inference

MAXMINER [4] In MAXMINER, *Bayardo* uses the following rule to derive a lower bound on the support of an itemset:

$$support(I \cup \{i\}) \leq support(I) - \sum_{j \in T} drop(J, j)$$

with $T = I - J$, $J \subset I$, and $drop(J, j) = support(J) - support(J \cup \{j\})$. This derivation corresponds to repeated application of rules $\mathcal{R}_I(I - \{i_1, i_2\})$.

PASCAL [3] In their PASCAL-algorithm, *Bastide et al.* use counting inference to avoid counting the support of all candidates. The rule they are using to avoid counting is based on our rule $\mathcal{R}_I(I - \{i\})$. In fact the PASCAL-algorithm corresponds to our algorithm when we only check rules up to depth 1, and do not prune derivable sets. Instead of counting the derivable sets, we use the derived support. Here the same remark as with the *ConFreqDFree*-representation applies; although PASCAL does not use all rules, in many cases the performance comes very close to evaluating all rules, showing that for these databases PASCAL is nearly optimal.

References

1. R. Agrawal, T. Imilienski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. ACM SIGMOD Int. Conf. Management of Data*, pages 207–216, Washington, D.C., 1993.
2. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. VLDB Int. Conf. Very Large Data Bases*, pages 487–499, Santiago, Chile, 1994.
3. Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, and L. Lakhal. Mining frequent patterns with counting inference. *ACM SIGKDD Explorations*, 2(2):66–74, 2000.
4. R. J. Bayardo. Efficiently mining long patterns from databases. In *Proc. ACM SIGMOD Int. Conf. Management of Data*, pages 85–93, Seattle, Washington, 1998.
5. J.-F. Boulicaut and A. Bykowski. Frequent closures as a concise representation for binary data mining. In *Proc. PaKDD Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, pages 62–73, 2000.
6. J.-F. Boulicaut, A. Bykowski, and C. Rigotti. Approximation of frequency queries by means of free-sets. In *Proc. PKDD Int. Conf. Principles of Data Mining and Knowledge Discovery*, pages 75–85, 2000.
7. A. Bykowski and C. Rigotti. A condensed representation to find frequent patterns. In *Proc. PODS Int. Conf. Principles of Database Systems*, 2001.
8. T. Calders. Deducing bounds on the frequency of itemsets. In *EDBT Workshop DTDM Database Techniques in Data Mining*, 2002.
9. S. Hettich and S. D. Bay. *The UCI KDD Archive. [http://kdd.ics.uci.edu]*. Irvine, CA: University of California, Department of Information and Computer Science, 1999.
10. D.E. Knuth. *Fundamental Algorithms*. Addison-Wesley, Reading, Massachusetts, 1997.
11. M. Kryszkiewicz. Concise representation of frequent patterns based on disjunction-free generators. In *Proc. IEEE Int. Conf. on Data Mining*, pages 305–312, 2001.
12. H. Mannila and H. Toivonen. Multiple uses of frequent sets and condensed representations. In *Proc. KDD Int. Conf. Knowledge Discovery in Databases*, 1996.
13. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *Proc. ICDT Int. Conf. Database Theory*, pages 398–416, 1999.
14. J. Pei, J. Han, and R. Mao. Closet: An efficient algorithm for mining frequent closed itemsets. In W. Chen, J.F. Naughton, and P.A. Bernstein, editors, *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, Dallas, TX, 2000.
15. M.J. Zaki and C. Hsiao. ChARM: An efficient algorithm for closed association rule mining. In *Technical Report 99-10, Computer Science, Rensselaer Polytechnic Institute*, 1999.
16. Z. Zheng, R. Kohavi, and L. Mason. Real world performance of association rule algorithms. In *Proc. KDD Int. Conf. Knowledge Discovery in Databases*, pages 401–406. ACM Press, 2001.