What You Store Is What You Get (extended abstract)

Floris Geerts, Bart Goethals, and Taneli Mielikäinen

HIIT Basic Research Unit
Department of Computer Science
University of Helsinki, Finland

Abstract. Recent studies emerged the need for representations of frequent itemsets that allow to estimate supports. Several methods have been proposed that achieve this goal by generating only a subset of all frequent itemsets. In this paper, we propose another approach, that given a minimum support threshold, stores only a small portion of the original database from which the supports of frequent itemsets can be estimated. This representation is especially valuable in case one is interested in frequent itemset of size at least some threshold value. However, we also present methods for estimating the support of smaller frequent itemsets.

1 Introduction

Due to the large amount of frequent itemsets that can be generated from transactional databases, recent studies emerged the need for concise representations of all frequent itemsets. In the context of inductive databases, one wants to answer support queries as efficiently as possible. Not only the effort to obtain the supports is important, but also the storage capacity needed plays an important role. This resulted in several successful algorithms that generate only a small portion of all frequent itemsets that allow to derive the support of all other frequent itemsets [2,3,4,5,6,7]. The drawback of some of these algorithms is that they compute representations which are difficult to interpret and to understand.

Two extreme examples of such algorithms are the following: When the primary concern is the data storage, the database itself can serve as representation for the frequent sets. Indeed, in many cases the number of unique transactions in the database is smaller than the number of all frequent itemsets. To obtain the support of itemsets, one simply scans the database and count. When time is of importance, the set of frequent itemsets itself provides an efficient representation since the support of an itemset is obtained by a simple lookup.

It is therefore only natural to ask for intermediate representations that store parts of *both* the transaction database and the set of frequent itemsets. In this paper, we define such a representation and report some initial investigations. To the best of our knowledge, such intermediate representations have not been considered before.

The transaction database part of this representation is obtained by *trimming*. The idea is to modify the transaction database optimally for finding all frequent

sets of size larger than a certain size k. This is done by throwing away transactions from the database that do not contribute to the support of any frequent k-itemset and by throwing away items from all transactions that no longer occur in any frequent k-itemset supported by that transaction. A similar technique is used to speed up Apriori-like algorithms, such as DHP [11] and DCI [10].

Although the frequencies of frequent itemsets of size larger or equal to k can be retrieved easily from such a trimmed database, this is less obvious for smaller itemsets.

We provide three approaches to obtain information about the support of itemsets of size smaller than k. The first approach consists of simply adding these frequent sets to the representation; in this way, the representation becomes intermediate. The second approach estimates the support of an itemset by computing it relative the the trimmed database. The last approach uses affine transformations to obtain a more accurate estimates for the supports of these small itemsets.

The paper is organized as follows. In Section 2 we provide the necessary definitions. Section 3 explains the process of database trimming and Section 4 shows three ways of dealing with small itemsets. In Section 5, we end the paper with some final thoughts and ideas for future work.

2 Preliminaries

Let $\mathcal{I}$ be a set of items. A subset of $\mathcal{I}$ is called an itemset, or a k-itemset if it consists of k items. A transaction over $\mathcal{I}$ is a pair (tid, I) where tid is the transaction identifier and I is an itemset. A transaction (tid, I) supports an itemset $J \subseteq \mathcal{I}$ if $J \subseteq I$. A transaction database $\mathcal{D}$ over $\mathcal{I}$ is a finite set of transaction over $\mathcal{I}$.

The cover of an itemset I in $\mathcal{D}$ is the set of transactions in $\mathcal{D}$ which support I:

$$\text{cover}\,(I, \mathcal{D}) := \{(tid, J) \in \mathcal{D} \mid I \subseteq J\}\,.$$

The support of an itemset I in $\mathcal{D}$, denoted by $\text{s}(I, \mathcal{D})$, is the size of its cover in $\mathcal{D}$. The cover of a set of itemsets $I_1, \ldots, I_k$ in $\mathcal{D}$, denoted by $\text{cover}(\{I_1, \ldots, I_k\}, \mathcal{D})$, is the union of the covers of the itemsets.

Let σ be a natural number, then an itemset I is σ-frequent in $\mathcal{D}$ if $\text{s}(I, \mathcal{D}) \geq \sigma$. We will denote the set of σ-frequent k-itemsets in $\mathcal{D}$ by $\mathcal{F}_{k,\sigma}(\mathcal{D})$, or $\mathcal{F}_{k,\sigma}$ when $\mathcal{D}$ is clear from the context. Also, $\mathcal{F}_\sigma := \cup_{i=0}^{\infty}\mathcal{F}_{i,\sigma}$, $\mathcal{F}_{\geq k,\sigma} := \cup_{i=k}^{\infty}\mathcal{F}_{i,\sigma}$ and $\mathcal{F}_{<k,\sigma} := \cup_{i=0}^{k-1}\mathcal{F}_{i,\sigma}$.

For any $(tid, I) \in \mathcal{D}$ and a set of itemsets $\mathcal{S}$, let

$$I\,[\mathcal{S}] := \{i \mid i \in J \wedge J \in \mathcal{S} \wedge J \subseteq I\}\,.$$

3 Trimming the database

In this section we propose a representation for all frequent itemsets of size at least a certain k. The representation consists of a transaction database which is

smaller than the original transaction database, and is obtained by trimming the original database.

3.1 Horizontal Trimming

The horizontal trimming consists of throwing away transactions from the database that do not contribute to the support of any frequent k-itemset. Define for $k = 1, 2, \ldots, |\mathcal{I}|$,

$$\mathcal{H}_{k,\sigma} := \text{cover}(\mathcal{F}_{k,\sigma}, \mathcal{D}).$$

Since the number of transactions supporting frequent k-itemsets decreases as k increases. i.e.,

$$\mathcal{D} \supseteq \mathcal{H}_{1,\sigma} \supseteq \cdots \supseteq \mathcal{H}_{|\mathcal{I}|,\sigma},$$

we have that $\text{cover}(\mathcal{F}_{\geq k,\sigma}, \mathcal{D}) = \text{cover}(\mathcal{F}_{k,\sigma}, \mathcal{D})$. We obtain the following:

Lemma 1. *For any $k = 1, 2, \ldots, |\mathcal{I}|$, we have that for any $I \in \mathcal{F}_{\geq k,\sigma}$,*

$$\text{s}(I, \mathcal{D}) = \text{s}(I, \mathcal{H}_{k,\sigma}). \tag{1}$$

Proof. Since $\mathcal{H}_{k,\sigma} \subseteq \mathcal{D}$, we immediately have the inequality $\text{s}(I, \mathcal{D}) \geq \text{s}(I, \mathcal{H}_{k,\sigma})$. We now prove that for $I \in \mathcal{F}_{\geq k,\sigma}$ also $\text{s}(I, \mathcal{D}) \leq \text{s}(I, \mathcal{H}_{k,\sigma})$ holds. Let $(tid, J) \in \text{cover}(I, \mathcal{D})$. Now, $I \in \mathcal{F}_{\geq k,\sigma}$ implies that $\text{cover}(I, \mathcal{D}) \subseteq \text{cover}(\mathcal{F}_{\geq k,\sigma}, \mathcal{D}) = \mathcal{H}_{k,\sigma}$, and hence also $(tid, J) \in \text{cover}(I, \mathcal{H}_{k,\sigma})$. □

This lemma implies that itemsets of size at least k are σ-frequent in $\mathcal{D}$ if and only if they are frequent in $\mathcal{H}_{k,\sigma}$. Hence, the following theorem holds.

Theorem 1. *For any k and σ,*

$$\mathcal{F}_{\geq k,\sigma}(\mathcal{H}_{k,\sigma}) = \mathcal{F}_{\geq k,\sigma}(\mathcal{D}).$$

3.2 Vertical Trimming

The vertical trimming consists of throwing away items from all transactions that no longer occur in any frequent k-itemset supported by that transaction.

Define for $k = 1, 2, \ldots, |\mathcal{I}|$,

$$\mathcal{V}_{k,\sigma} := \{(tid, I[\mathcal{F}_{k,\sigma}]) \mid (tid, I) \in \mathcal{D}\}.$$

Similar to Theorem 1 we obtain:

Theorem 2. *For any k and σ,*

$$\mathcal{F}_{\geq k,\sigma}(\mathcal{V}_{k,\sigma}) = \mathcal{F}_{\geq k,\sigma}(\mathcal{D}).$$

3.3 Combined Trimming

The combined trimming consists of performing both horizontal and vertical trimming. Define for $k = 1, 2, \ldots, |\mathcal{I}|$,

$$\mathcal{D}_{k,\sigma} := \text{cover}(\mathcal{F}_{k,\sigma}, \mathcal{V}_{k,\sigma})$$

or equivalently,

$$\mathcal{D}_{k,\sigma} := \{(tid, I[\mathcal{F}_{k,\sigma}]) \mid (tid, I) \in \mathcal{H}_{k,\sigma}\}.$$

Again, we obtain:

Theorem 3. *For any k and σ,*

$$\mathcal{F}_{\geq k,\sigma}(\mathcal{D}_{k,\sigma}) = \mathcal{F}_{\geq k,\sigma}(\mathcal{D}).$$

Note that $\mathcal{D}_{k,\sigma}$ actually consists of all non-empty transactions in $\mathcal{V}_{k,\sigma}$.

3.4 An example

We illustrate the effect of trimming on the BMS-Webview-1 database [15]. The support is fixed to $\sigma = 36 = 0.06 \times 59602$. A lower support is not feasible due to the combinatorial explosion of the number of frequent itemsets.

The size of the trimmed database $\mathcal{D}_{k,\sigma}$ in function of k is shown in Figure 1. As can be seen, the size of the trimmed transaction database decreases very quickly when k increases.

Of course, in general this is not always the case. For example, the regularly used mushroom dataset from the UCI repository of machine learning databases [1] contains approximately 8000 transactions of which almost all of them contain frequent itemsets of all sizes (for most widely used minimum support thresholds).

4 Taking care of the smaller itemsets

The database trimming we proposed in the previous section provides a concise representation for the frequent sets of size at least k. However, sometimes one is also interested in the small frequent sets. In this section, we propose three methods for estimating the support of itemsets of size smaller than k.

4.1 A naive lossless approach

A naive approach to obtain the exact support of any itemset of size smaller than k is to augment $\mathcal{D}_{k,\sigma}$ to a representation for all frequent sets by adding the frequent sets of size smaller than k.

Theorem 4. *For any k and σ,*

$$\mathcal{F}_{<k,\sigma}(\mathcal{D}) \cup \mathcal{F}_{\geq k,\sigma}(\mathcal{D}_{k,\sigma}) = \mathcal{F}_{\sigma}.$$

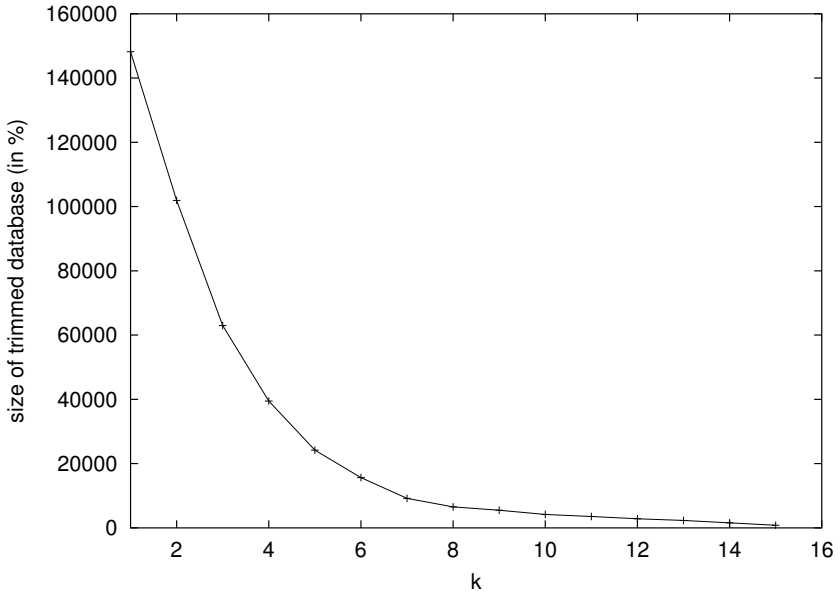

Fig. 1. Effect of trimming on BMS-Webview-1 database with $\sigma = 36$.

Note that for $k = 1$, the proposed representation actually consists of the original database from which all infrequent items and uncovered transactions are removed. The only frequent set stored in this case is the empty set whose support corresponds to the number of transactions in the original database. For other values of k, this representation consists of a part of the database and a part of all frequent itemsets.

In Figure 2 the representation for all frequent sets for the BMS-Webview-1 dataset and $\sigma = 36$ is shown. The figure shows for each k the size of this lossless representation. Additionally, the total size what is size? of the original database is shown, as well as the total size what is size? of all frequent itemsets. As can be seen, for small k, the size of the proposed representation becomes half of the original database. For larger k, the trimmed database becomes much smaller (as could be seen on Figure 1), and hence, a lot of information contained in those transactions get lost, such that much more frequent itemsets need to be stored. Eventually, the other extreme is reached in which all frequent itemsets are stored and the trimmed database is empty. Note that for this dataset with the used minimum support threshold, the original database itself is already orders of magnitude smaller than the total size of all frequent itemsets. Hence, the space-time tradeoff of storing all frequent itemsets, or only storing the database becomes eminent in this case. Fortunately, intermediate solutions are near.

The complexity of retrieving the support of an arbitrary frequent itemset I then amounts to a simple lookup in $\mathcal{F}_{<k,\sigma}(\mathcal{D})$ if $|I| < k$. In the other case, if $|I| \geq k$, we have to compute the support of I in the sometimes significantly

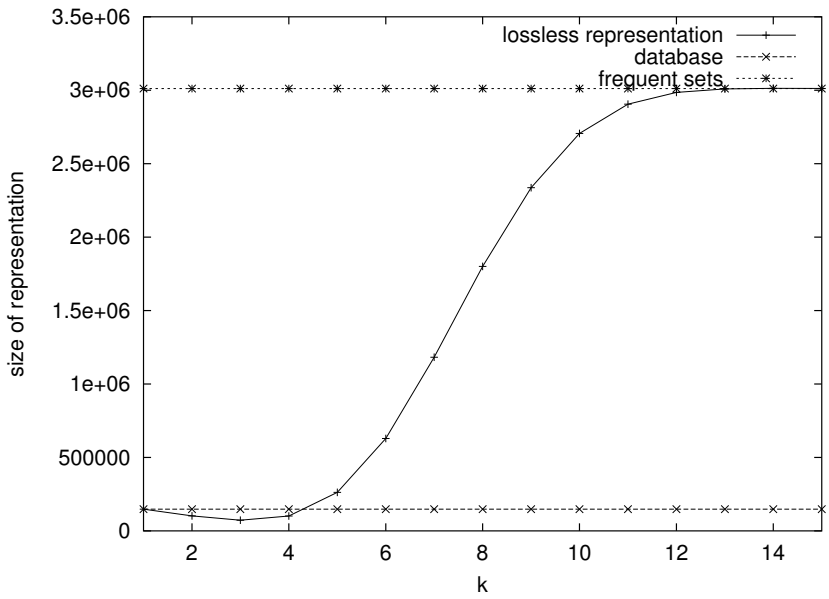

Fig. 2. The representation for $\mathcal{F}_\sigma$ for the BMS-Webview-1 dataset and $\sigma = 36$.

smaller database $\mathcal{D}_{k,\sigma}$. Additionally, we can use any other concise representation on $\mathcal{F}_{<k,\sigma}(\mathcal{D})$ to reduce the total size even more [2,8,9,12,13].

4.2 A naive lossy approach

If the exact support of the itemsets of size smaller than k is not important, but an approximation suffices, then in many cases, $\mathcal{D}_{k,\sigma}$ itself can already provide a useful estimate for the support of these smaller itemsets.

In Figure 3 we show the squared error

$$\text{sqrerr}\left(\mathcal{F}_\sigma(\mathcal{D}), \mathcal{F}_\sigma(\mathcal{D}_{k,\sigma})\right) = \sum_{I \in \mathcal{F}_\sigma(\mathcal{D})} \left(\text{s}(I, \mathcal{D}) - \text{s}(I, \mathcal{D}_{k,\sigma})\right)^2$$

in function of k. Here, we used again the BMS-Webview-1 database with $\sigma = 36$.

4.3 A less lossy less naive approach

Another approach is to partition the frequent sets by grouping the frequent sets of size at most $k-1$ and to determine how each group should be corrected using one correction operator within one group. The grouping and correction operators for each group should have simple descriptions.

As a concrete example, we consider the case where the grouping is determined by the size of the sets. That is, our groups that need to be corrected are $\mathcal{F}_{0,\sigma}, \ldots, \mathcal{F}_{k-1,\sigma}$. As a correction we compute an affine transformation

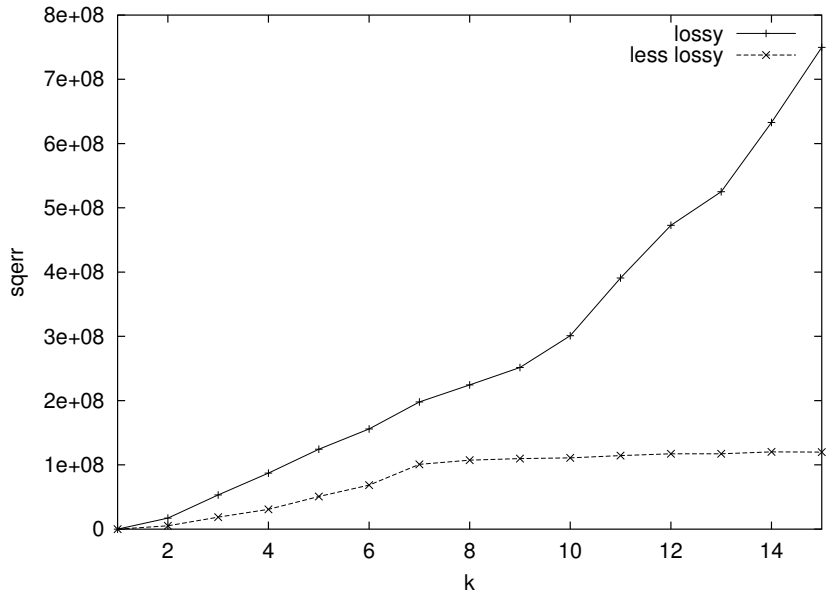

Fig. 3. The squared error for $a = 1$ and $b = 0$ compared with the squared error for optimal values for a and b for the BMS-Webview-1 database with $\sigma = 36$.

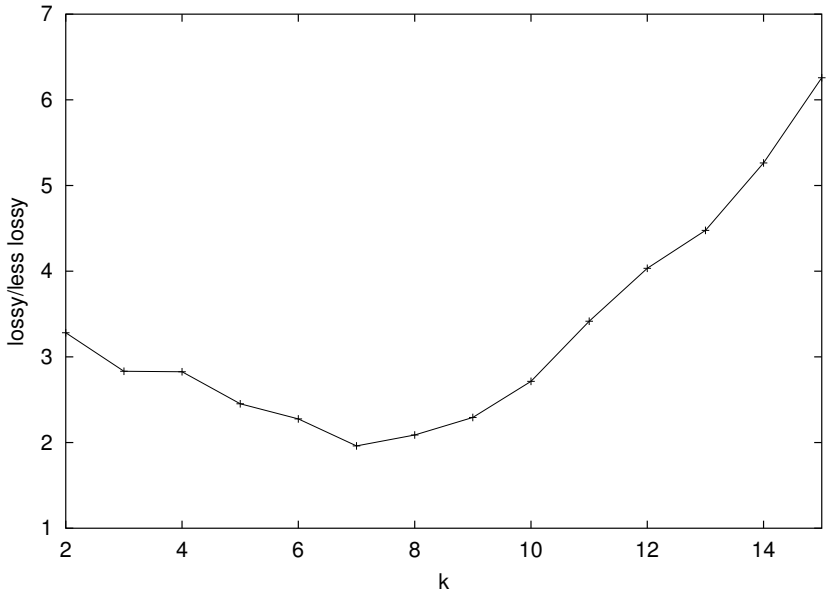

Fig. 4. The ratio of the squared error for $a = 1$ and $b = 0$ and the squared error for optimal values for a and b for the BMS-Webview-1 database with $\sigma = 36$.

$(x, a_i, b_i) \mapsto a_i x + b_i$ for each group $\mathcal{F}_{i,\sigma}, 0 \leq i \leq k-1$ that minimized the sum of squared differences between the correct support and the support estimated from $\mathcal{D}_k$:

$$\begin{aligned}\text{sqrerr}\left(\mathcal{F}_\sigma(\mathcal{D}), \mathcal{F}_\sigma(\mathcal{D}_k)\right) &= \sum_{I \in \mathcal{F}_\sigma} \left(\text{s}(I, \mathcal{D}) - (a_{|I|}\text{s}(I, \mathcal{D}_{k,\sigma}) + b_{|I|})\right)^2 \\ &= \sum_{I \in \mathcal{F}_{<k},\sigma} \left(\text{s}(I, \mathcal{D}) - (a_{|I|}\text{s}(I, \mathcal{D}_{k,\sigma}) + b_{|I|})\right)^2.\end{aligned}$$

The error sqrerr is minimized by choosing $(a_i, b_i), 0 \leq i \leq k-1$ to be (see e.g. [14]):

$$a_i = \frac{|\mathcal{F}_{i,\sigma}| \sum_{I \in \mathcal{F}_{i,\sigma}} \text{s}(I, \mathcal{D})\text{s}(I, \mathcal{D}_{k,\sigma}) - \left(\sum_{I \in \mathcal{F}_{i,\sigma}} \text{s}(I, \mathcal{D})\right)\left(\sum_{I \in \mathcal{F}_{i,\sigma}} \text{s}(I, \mathcal{D}_{k,\sigma})\right)}{|\mathcal{F}_{i,\sigma}| \sum_{I \in \mathcal{F}_{i,\sigma}} \text{s}(I, \mathcal{D}_{k,\sigma})^2 - \left(\sum_{I \in \mathcal{F}_{i,\sigma}} \text{s}(I, \mathcal{D}_{k,\sigma})\right)^2}$$

$$b_i = \frac{\sum_{I \in \mathcal{F}_{i,\sigma}} \text{s}(I, \mathcal{D}) - a_i \sum_{I \in \mathcal{F}_{i,\sigma}} \text{s}(I, \mathcal{D}_{k,\sigma})}{|\mathcal{F}_{i,\sigma}|}$$

For example, the correction (a_0, b_0) for the support of the empty set (i.e., all frequent sets of size 0 as $\mathcal{F}_{0,\sigma} = \{\emptyset\}$) is equal to $(|\mathcal{D}|/|\mathcal{D}_k|, 0)$. In general (a_i, b_i) attempts to tell how much the databases $\mathcal{D}$ and $\mathcal{D}_k$ differ when only considering the frequent i-itemsets: a_i scales the supports and b_i shifts them.

This method is illustrated in Figure 3 for the BMS-Webview-1 database with $\sigma = 36$. To adjust the estimates even more, we also use the trivial information that supports are non-negative and cannot be larger than the number of transactions in the database. In Figure 4 we compare the lossy approach with the the less lossy approach.

5 Conclusions and future work

In the context of inductive databases for frequent itemsets, it is important to find good representations of the dataset in order to efficiently answer support queries. Instead of storing only a subset of all frequent itemsets, we propose to store parts of both the set of frequent itemsets and the database. In this way, the support of small frequent itemsets can be computed using a simple lookup, and the support of large frequent itemsets can be computed using a scan through a sometimes significantly reduced database.

When the representation is allowed to be lossy, two techniques are proposed which already show promising results with respect to the error on the support of an approximated itemset.

The proposed methodology of storing only a part of the database and a part of the frequent itemsets sheds a new light on representations, which we will investigate further. An interesting question is whether there exists an optimal separation of transactions and frequent itemsets, such that the resulting representation is small but still offers an efficient method to compute or approximate the support of all frequent itemsets.

Acknowledgements

We thank Blue Martini Software for contributing the KDD Cup 2000 data which we used in our experiments.

References

1. C.L. Blake and C.J. Merz. *UCI Repository of machine learning databases*. University of California, Irvine, Dept. of Information and Computer Sciences, `http://www.ics.uci.edu/~mlearn/MLRepository.html`, 1998.
2. J.-F. Bouliçaut and A. Bykowski. Frequent closures as a concise representation for binary data mining. In T. Terano, H. Liu, and A. L. P. Chen, editors, *Knowledge Discovery and Data Mining*, volume 1805 of *Lecture Notes in Artificial Intelligence*, pages 62–73. Springer-Verlag, 2000.
3. J.-F. Bouliçaut, A. Bykowski, and C. Rigotti. Free-sets: a condensed representation of Boolean data for the approximation of frequency queries. *Data Mining and Knowledge Discovery*, 7(1):5–22, 2003.
4. T. Calders and B. Goethals. Mining all non-derivable frequent itemsets. In T. Elomaa, H. Mannila, and H. Toivonen, editors, *Principles of Data Mining and Knowledge Discovery*, volume 2431 of *Lecture Notes in Artificial Intelligence*, pages 74–865. Springer-Verlag, 2002.
5. T. Calders and B. Goethals. Minimal representations of frequent sets. In Nada Lavrac, Dragan Gamberger, Ljupco Todorovski, and Hendrik Blockeel, editors, *Proceedings of the 7th European Conference on Principles of Data Mining and Knowledge Discovery*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 2003. To appear.
6. M. Kryszkiewicz. Concise representation of frequent patterns based on disjunction-free generators. In N. Cercone, T. Y. Lin, and X. Wu, editors, *Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 305–312. IEEE Computer Society, 2001.
7. H. Mannila and H. Toivonen. Multiple uses of frequent sets and condensed representations. In E. Simoudis, J. Han, and U. M. Fayyad, editors, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 189–194. AAAI Press, 1996.
8. T. Mielikäinen. Frequency-based views to pattern collections. In *IFIP/SIAM Workshop on Discrete Mathematics and Data Mining*, 2003.
9. T. Mielikäinen and H. Mannila. The pattern ordering problem. In Nada Lavrac, Dragan Gamberger, Ljupco Todorovski, and Hendrik Blockeel, editors, *Proceedings of the 7th European Conference on Principles of Data Mining and Knowledge Discovery*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 2003. To appear.
10. S. Orlando, P. Palmerini, R. Perego, and F. Silvestri. Adaptive and resource-aware mining of frequent sets. In V. Kumar, S. Tsumoto, P.S. Yu, and N.Zhong, editors, *Proceedings of the 2002 IEEE International Conference on Data Mining*, pages 338–345. IEEE Computer Society, 2002.
11. J.S. Park, M.-S. Chen, and P.S. Yu. An effective hash based algorithm for mining association rules. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, volume 24(2) of *SIGMOD Record*, pages 175–186. ACM Press, 1995.

12. D. Pavlov, H. Mannila, and P. Smyth. Beyond independence: probabilistic methods for query approximation on binary transaction data. *IEEE Transactions on Data and Knowledge Engineering*, To appear.
13. J. Pei, G. Dong, W. Zou, and J. Han. On computing condensed pattern bases. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), 9-12 December 2002, Maebashi City, Japan*, pages 378–385. IEEE Computer Society, 2002.
14. W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C++: The Art of Scientific Computing*. Cambridge University Press, 2002.
15. Z. Zheng, R. Kohavi, and L. Mason. Real world performance of association rule algorithms. In F. Provost and R. Srikant, editors, *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 401–406. ACM Press, 2001.