Under consideration for publication in Knowledge and Information
Systems

Approximating the Number of Frequent Sets in Dense Data [1]

Mario Boley and Henrik Grosskreutz

Fraunhofer IAIS, Schloss Birlinghoven, 53754 Sankt Augustin, Germany

Abstract. We investigate the problem of counting the number of frequent (item)sets—a problem known to be intractable in terms of an exact polynomial time computation. In this paper, we show that it is in general also hard to approximate. Subsequently, a randomized counting algorithm is developed using the Markov chain Monte Carlo method. While for general inputs an exponential running time is needed in order to guarantee a certain approximation bound, we show that the algorithm still has the desired accuracy on several real-world datasets when its running time is capped polynomially.

Keywords: Data mining; Frequent Itemsets; Approximate Counting; Markov chain Monte Carlo

1. Introduction

Frequent pattern mining is considered one of the most influential methods in data mining as compiled by Wu et al. (2008), and recently it has even made the step into commercial database systems (Yoshizawa, Pramudiono and Kitsuregawa, 2000; Li and Mozes, 2004; Utley, 2005). It is used within several combinations of local pattern types and application domains. A few examples for such combinations are association rules for market basket data, frequent subgraphs for molecule prediction, and sequential patterns for time series data (see Han and Kamber, 2000 for an overview). Their unifying property is that patterns are only considered interesting if they satisfy a (minimum) frequency constraint, i.e., a certain number of records of the input dataset have to "contain" the pattern.

---

[1] A short version of this paper has appeared in the Proceedings of the 2008 Eighth IEEE International Conference on Data Mining (ICDM 2008).

Received Jan 14, 2009
Revised Mar 09, 2009
Accepted Apr 03, 2009

In contrast to most other work devoted to frequent pattern mining which deals with the development and evaluation of algorithms that *list* all frequent patterns, in this study, we are interested in *counting* them quickly. Knowing the relationship between frequency threshold and the resulting number of frequent patterns can for instance be used for computing a *frequency-plot*, i.e., a plot showing all possible thresholds (x-axis) against the corresponding number of frequent sets (y-axis). Having such a plot prior to the actual mining process can for instance be used within intelligent discovery assistants (IDAs) (Bernstein, Provost and Hill, 2005) in order to either provide user guidance for setting the frequency threshold or to tune it automatically.

For this purpose it is an essential requirement that the involved computation is performed quickly. "Indeed, the problem is precisely to predict a combinatorial explosion without suffering from it [. . .]", as Geerts, Goethals and Bussche (2005) have put it in their related study of bounding the number of candidate patterns that have to be processed within a BFS-listing of all frequent patterns. In particular for dense datasets this requirement prohibits the use of any of the known exhaustive data mining algorithms that list all frequent patterns. Even though these algorithms have been optimized to an impressive level in recent years, they have the number of frequent sets as an inherent lower bound of their time complexity. While this does not pose a problem for sparse datasets, in dense ones this number behaves in essence exponentially (for low frequency thresholds). For that reason we are aiming for an algorithm counting the number of frequent sets in a time that does not depend on that number. Since it is well known that no exact deterministic algorithm with this property can exist (unless $\mathbf{P} = \mathbf{NP}$), we are aiming for a randomized approximation algorithm using the Markov chain Monte Carlo method. Moreover, we restrict ourselves to the case of plain frequent (item)sets, which are used, e.g., for generating association rules. In summary, we are aiming for an input polynomial algorithm solving the following computational problem:

Problem 1 (#-FREQUENT SETS). *Given* a transactional dataset and a frequency threshold, *compute* the cardinality of the corresponding frequent set family.

We discuss the question of why all this is a worthwhile venture in some more detail in Section 2. After some formal definitions (Section 3) that are needed for the subsequent technical content we analyze theoretical limits in Section 4. As the main result of this section we show that the number of frequent sets is hard to approximate. In addition we interpret some known complexity results in the context of our problem. We then develop a randomized approximation scheme in Section 5 that makes use of a sampling procedure presented in section 6. Details and speed-ups are presented in Section 7 resulting in a hybrid algorithm with an exhaustive and an approximative counting phase. As indicated by the hardness result, we show that for general inputs the algorithm's correctness cannot be guaranteed or its time complexity is not bounded polynomially. However, experiments we present in Section 8 constitute its applicability on several real-world and synthetic datasets. A concluding discussion is given in Section 9.

2. Motivation

In order to raise business value of data mining or generally to make it more accessible to non-expert users, approaches like Intelligent Discovery Assistants (IDAs) (Bernstein et al., 2005) or Mining Mart (Morik and Scholz, 2002) have been proposed. Both assist a user in selecting a valid data mining process for their data. Still, once a user has decided on a valid process the next problem is to find good parameter settings, and in case a frequent pattern mining step is part of the process this usually involves a minimum frequency threshold. Knowledge about the relation between frequency threshold and the corresponding number of frequent sets is very helpful in this context, as it allows to control the output size and thus indirectly the output time, because frequent set mining algorithms usually exhibit a time complexity that is roughly linear in the output. Consequently, quickly counting the frequent sets helps to make optimal use of the available time budget. Moreover, as we will argue below, a computed frequency plot can also help semantically to setup and interpret the whole process. It should be noted that the following discussion is illustrative and motivating. A thorough investigation of how to choose a minimum frequency threshold justifies a complete study in its own right.

2.1. Frequency Plots

In order to explain the possible use of a frequency plot let us assume we have a dataset generated by the following illustrative and absolutely idealized underlying process.

Example 2 (Beginner's Guide Process). There is a hobby shop carrying items for k different hobbies. In particular there is a "beginner's guide" a_i for every hobby and a corresponding "starter kit" consisting of items $c_{i,1}, \ldots, c_{i,l(i)}$ $(1 \leq i \leq k)$. A usual (senior) customer purchases every item independent from one another with probability p_c except for the beginner's guides, which he will never purchase. However, with probability p_a (per hobby independent from one another) a customer will pick up a new hobby and buy the corresponding beginner's guide a_i. In this case he will always also purchase items $c_{i,1}$ to $c_{i,l(i)}$ and behaves like a usual customer otherwise.

Clearly, the most (if not the only) interesting association rules for this underlying process are the rules

$$a_i \to c_{i,1} \ldots c_{i,l(i)}$$

for $i \in \{1, \ldots, k\}$. All of these rules will have the maximum confidence[2] of 1, which measures the actual semantical value of a rule. Whereas their expected support is equal to the probability p_a. We generated three datasets using the above process for $p_a = 0.1, 0.13, 0.16$ and $p_c = 0.6$. Each of the datasets was generated with $k = 5$ implicant items, 10 different consequence items for each implicant, and 5000 transactions. Figure 1 shows their resulting frequency plots. Depending on initial assumption on the generative process there can be different

---

[2] The confidence of a rule is the fraction of transactions containing a rule's consequence among those containing its antecedence.

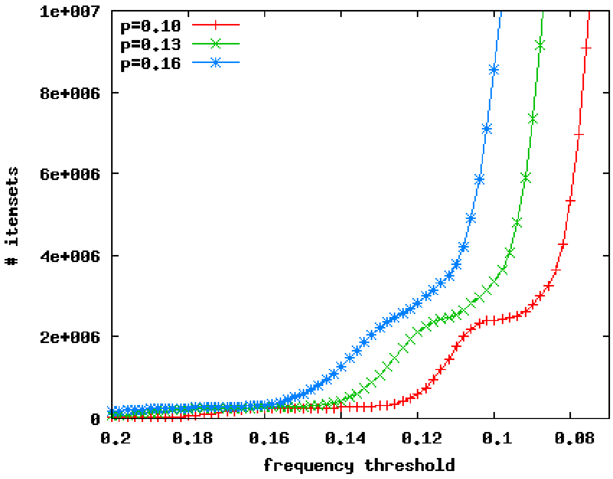


Fig. 1. frequency plots

strategies of inferring a good threshold from this plot. First, however, it is necessary to clarify the motivation of introducing a minimum frequency threshold at all.

2.2. Frequency Thresholds

The reasons for introducing a frequency constraint can roughly be subsumed under three purposes:

(1) suppress statistically insignificant results that are a mere random fluctuation of the underlying data generating process (for instance all antecedences occurring in only one transaction will induce rules with maximum confidence),
(2) raise output pattern value in the sense that they are applicable to more data instances, i.e., there are more records satisfying the rule's antecedence, and
(3) allow additional pruning of search space thus making otherwise intractable tasks tractable.

In particular for the last purpose, the frequency threshold is often set to higher values than it would be necessary for reasons (1) and (2) alone. Clearly, this can harm the analytical value of the resulting patterns, because patterns that are interesting according to a primary criterion (e.g., confidence) might be pruned without any statistical reason purely for the sake of performance. Thus, it would be desirable to set the frequency threshold to the minimum value that is reasonable with respect to reasons (1) and (2).

For the above example observe that for all three datasets the interesting rules have an expected support that lies before that point of when the plot finally turns to a purely exponential behavior. Thus, a user setting the frequency threshold to this point is expected to not miss most of them. Although this may not always be the case, this strategy generally aims to find the most conservative threshold that preserves most interesting rules while it suppresses most of the statistically insignificant sets. Moreover, the plots lead to a small set of reasonable candidate frequency points, namely the sockets before steep exponential slopes. Clearly, this is an improvement when compared to the uninformed approach of trial-and-error parameter twiddling.

2.3. Related Work and Objections

An important related approach to raise user control in frequent set mining is listing only the top-K frequent sets proposed by Wang, Han, Lu and Tzvetkov (2005). While this approach takes care of output size control and scheduling, it fails to provide the global overview of computing a frequency plot for all possible frequency values—a task that cannot be done using a top-K miner (in particular on dense datasets) because the same restrictions apply to them as stated in the introduction for ordinary exhaustive miners. Moreover, the most frequent sets are not necessarily the most interesting ones (Tatti, 2008; Boley, Horváth and Wrobel, 2009).

Still, this may inspire a general objection to the above motivation: Why is it useful to compute regions of the plot that correspond to frequent set families so large that they are impossible to list in a subsequent mining step? The answer to this lies in the fact that frequency is usually only a subsidiary interestingness criterion. While the frequency plot shows the effect of that criterion alone, the final pattern set may result from a conjunction of several constraints, e.g., confidence or lift, many of which can be used for pruning already during frequent set listing (see Pei and Han, 2000 or Hämäläinen and Nykänen, 2008). Thus, the resulting output family, albeit its low frequency threshold, can in fact be listed effectively. Moreover, if effective knowledge post-processing is employed (e.g., based on visual analytics as in Blanchard, Guillet and Briand, 2007) it is often reasonable to produce the maximum number of frequent sets than can possibly be computed within a given time budget.

A different approach for replacing the minimum frequency threshold by a more intuitive paramater has been proposed by Zhang, Wu, Zhang and Lu (2008). In their work, user-specified “fuzzy” thresholds like “more or less frequent” or “highly frequent” are translated into traditional frequency thresholds. This translation is based on an approximation to the average support that, in turn, relies on an independence assumption for the distribution of single items. While this method simplifies parameter setting for the user, it does not provide control over the output size and the corresponding computation time.

3. Preliminaries

In this section we recall and fix notions from frequent set mining, randomized approximation algorithms, and Markov chains that are needed in the subsequent discussion.

Frequent Set Mining and Prefix Trees Let E be a finite set. A *dataset* $\mathcal{D}$ over E is a finite multiset with $D \subseteq E$ for all $D \in \mathcal{D}$. In the context of frequent pattern mining the elements of E are often called *items* and the elements of $\mathcal{D}$ *transactions.* For a set $F \subseteq E$ we define its *support multiset* as $\mathcal{D}[F] = \{D \in \mathcal{D} \colon D \supseteq F\}$. *Frequency thresholds* can be specified as absolute or relative thresholds depending on what is more convenient in a given situation. For an (absolute) integer threshold $f \in \{1, \ldots, |\mathcal{D}|\}$, F is called *f-frequent* (or frequent) in $\mathcal{D}$ if $|\mathcal{D}[F]| \geq f$ respectively for a (relative) real threshold $f \in (0,1)$, if $|\mathcal{D}[F]| \geq f\,|\mathcal{D}|$. The family of all f-frequent sets in $\mathcal{D}$ is denoted $\mathcal{F}(\mathcal{D}, f)$ or just as $\mathcal{F}$ when $\mathcal{D}$ and f are clear from the context.

Unless stated differently we denote the number of items by n and identify E with the set $\{1, \ldots, n\}$ throughout this article. This allows use to use the natural order on $\{1, \ldots, n\}$ and particularly to use the symbols $\max F$ and $\min F$ for $F \subseteq E$. In addition we assume without loss of generality that the items are numbered in descending order of their frequency, i.e, $1 \leq i < j \leq n \Rightarrow |\mathcal{D}[\{i\}]| \geq |\mathcal{D}[\{j\}]|$. For a set $F \subseteq E$ and $i \in \{0, \ldots, n\}$ we denote by $F_i = F \cap \{1, \ldots, i\}$ the *i-prefix* of F. The *prefix tree* of $\mathcal{D}$ is a labeled directed tree (arborescence) $T = (V, E, \phi)$ with nodes $V = \{D_i : \ D \in \mathcal{D}, 0 \leq i \leq n\}$, i.e., the set of all prefixes occurring in $\mathcal{D}$, edges $E = \{(D_i, D_j) : \ D \in \mathcal{D}, D_j = D_i \cup \{j\}\}$, node labels $\phi(X) = |\{D \in \mathcal{D} \colon \exists i, D_i = X\}|$, and edge labels $\phi((D_i, D_j)) = j$.

Probabilistic Approximation Algorithms A *bounded probability (BP) algorithm* for a problem with instances X and possible solutions Y specified by a correctness relation $R \subseteq X \times Y$ is a probabilistic algorithm $\mathcal{A}$ such that it holds that $\mathbb{P}[(x, \mathcal{A}(x)) \in R] \geq 3/4$ where $\mathcal{A}(x)$ denotes the output of $\mathcal{A}$ on input x. Now we consider the case when Y is the set of natural numbers $\mathbb{N}$. A *randomized approximation scheme* for a mapping $g : X \to \mathbb{N}$ is a BP-algorithm $\mathcal{A}$ taking arguments $x \in X$ and $\epsilon \in (0,1)$ satisfying the relaxed correctness predicate $R_\epsilon = \{(x,y) \colon \ (1-\epsilon)g(x) \leq y \leq (1+\epsilon)g(x)\}$ (this is sometimes just described by the phrase "y is ϵ-close to $g(x)$"), i.e., $\mathcal{A}$ satisfies

$$\mathbb{P}[(1-\epsilon)g(x) \leq \mathcal{A}(x,\epsilon) \leq (1+\epsilon)g(x)] \geq 3/4 \ . \tag{1}$$

Such an algorithm is called *fully polynomial* if its time complexity is bounded in a polynomial in size(x) and $1/\epsilon$. The constant $3/4$ appearing in the definition has no significance other than being strictly between $1/2$ and 1. Any two success probabilities from this interval can be reached from one another by a small number of repetitions of the corresponding algorithm and returning the median of the results (see Jerrum, Valiant and Vazirani, 1986).

A weaker notion of approximation is given by the following definition: An algorithm $\mathcal{A}$ is called an *α-factor approximation* of g (or said to *approximate g within α*) if it satisfies the correctness relation $R_\alpha = \{(x,y) \colon \ g(x)/\alpha(x) \leq y \leq \alpha(x)g(x)\}$ where $\alpha \colon X \to \mathbb{R}$ is a function that may grow in the size of x. Clearly, an efficient approximation scheme can act as an efficient c-factor approximation algorithm for all constants $c > 1$.

Markov Chains A (discrete) *Markov chain* on state space Ω is a sequence of discrete random variables $\mathfrak{M} = X_1, X_2, \ldots$ with domain Ω satisfying the Markov condition, i.e.,

$$\mathbb{P}[X_{n+1} = x | X_1 = x_1, \ldots, X_n = x_n] = \mathbb{P}[X_{n+1} = x | X_n = x_n]$$

for all $n \in \mathbb{N}$ and $x, x_1, \ldots, x_n \in \Omega$ satisfying $\mathbb{P}[X_1 = x_1, \ldots, X_n = x_n] > 0$. In this article we only consider finite state spaces Ω. Thus, given a probability distribution on the initial state, $\mathfrak{M}$ is completely specified by the *state transition probabilities* $P(x,y) = \mathbb{P}[X_{n+1} = y | X_n = x]$ of all $x, y \in \Omega$ that do not depend on n. The $(|\Omega| \times |\Omega|)$-matrix containing $P(x,y)$ in column x and row y is a stochastic matrix we denote by P. The t-th power of this matrix contains the probability of going from x to y in t steps $P^t(x,y) = \mathbb{P}[X_{n+t} = y | X_n = x]$. We call a state

$y \in \Omega$ *reachable* from a state $x \in \Omega$ if there is a $t \in \mathbb{N}$ such that $P^t(x, y) > 0$. A Markov chain $\mathfrak{M}$ is called *aperiodic* if for all $x, y \in \Omega$ with x is reachable from y there is a $t_0 \in \mathbb{N}$ such that for all $t \geq t_0$ it holds that $P^t(x, y) > 0$, and it is called *irreducible* if any two states are reachable from one another. Finally, $\mathfrak{M}$ is called *ergodic* if it is irreducible and aperiodic.

Any ergodic Markov chain has a unique *limiting stationary distribution* $\pi\colon \Omega \to [0, 1]$, i.e., for all states $x, y \in \Omega$ it holds that $\lim_{t\to\infty} P^t(x, y) = \pi(y)$. Moreover, if there is a function $\pi'\colon \Omega \to [0, 1]$ satisfying the *detailed balance condition* $\forall x, y \in \Omega, \pi'(x)P(x, y) = \pi'(y)P(y, x)$ then π' is a stationary distribution. It follows that ergodic Markov chains with *symmetric* transition probabilities always converge to the *uniform distribution*. The distance from the t-step distribution of a Markov chain with $X_0 = x$ to its stationary distribution can be measured by the *total variation distance* $\|P^t(x, \cdot), \pi\|_{\mathrm{tv}} = 1/2 \sum_{y\in\Omega} |P^t(x, y) - \pi(y)|$. Using this definition we can define the *mixing time* of $\mathfrak{M}$ by

$$\tau(\epsilon) = \max_{x\in\Omega} \min\{t_0 \in \mathbb{N}\colon \forall t \geq t_0,\ \|P^t(x, \cdot), \pi\|_{\mathrm{tv}} \leq \epsilon\}$$

as the minimum number of steps one has to simulate $\mathfrak{M}$ until the resulting distribution is guaranteed to be ϵ-close to its stationary distribution. For more details and results about Markov chains and their mixing time we refer to Randall's survey (Randall, 2006).

4. Problem Complexity

Gunopulos et al. proved **#P**-hardness[3] of #-FREQUENT SETS implying that there is no exact algorithm for that problem unless **P** = **NP** (Gunopulos, Khardon, Mannila, Saluja, Toivonen and Sharma, 2003). They did this using a reduction from the **#P**-complete problem of computing the number of satisfying truth assignments of a given monotone 2-CNF formula, i.e., a conjunctive normal form formula containing only two positive literals per clause. It was shown by Zuckerman (1996) that this number and in fact even its logarithm is hard to approximate within a factor of n^ϵ for instances of size n.

The reduction in (Gunopulos et al., 2003), however, transforms a 2-CNF formula into a transaction dataset with n items such that the number of satisfying truth assignments corresponds to the number of sets that are *not* 1-frequent, and then it uses the fact that the number of infrequent sets is equal to 2^n minus the number of frequent sets. Hence, the construction is highly non-parsimonious, i.e., the numbers usually change drastically without any reasonable bound. As a consequence relative approximation guarantees are not preserved by that reduction and it does not lead to a hardness result for approximating #-FREQUENT SETS. Still, it is an important side note that the two aforementioned theorems together do imply the strong result that there is no efficient approximation algorithm for counting the number of infrequent sets even if the (absolute) frequency threshold is fixed to 1. This is an interesting difference to the same restriction for #-FREQUENT SETS: When restricted to frequency threshold 1 approximating the number of frequent sets becomes equivalent to approximating the number of satisfying assignments of a given DNF-formula, and for this problem

---

[3] Essentially, **#P** is the class of problems asking to count the number of solutions where it is possible in polynomial time to verify a single solution (Valiant, 1979).

there is a fully polynomial randomized approximation scheme (Karp, Luby and Madras, 1989).

Now, for acquiring a hardness result for the number of frequent sets we have to choose a different starting point, namely the hardness of approximating a frequent set of maximum cardinality. With this approach we can show the following result:

Theorem 3. Unless for all $\epsilon > 0$ and for all problems in **NP** there is a BP-algorithm that runs in time $2^{n^{\epsilon}}$ for instances of size n, the following holds: There is a constant $\delta_{\#\mathrm{F}}$ such that there is no polynomial time BP-algorithm that, given a dataset $\mathcal{D}$ over n items and a frequency threshold f, approximates $\log|\mathcal{F}(\mathcal{D},f)|$ within $n^{\delta_{\#\mathrm{F}}}$.

Proof. It was shown in (Boley, 2007) that under the same assumption as in the claim there is no polynomial time algorithm approximating a frequent set of maximum cardinality within $n^{\delta_{\mathrm{BC}}}$. That result was based on Khot's seminal inapproximability result for Bipartite Clique, which in fact ruled out BP-algorithms under the above assumption (see Khot, 2004 where you can also find more information about the magnitude of δ_{BC}). Furthermore, it is easy to prove that approximating only the maximum number k such that there is a frequent set of size k is polynomially equivalent to the actual construction of a corresponding set. Thus, it is sufficient to show that an algorithm for approximating the logarithm of $|\mathcal{F}|$ can be used to approximate this number k.

Since all subsets of a maximizing frequent set $F \in \mathcal{F}$ with $|F| = k$ are also frequent, it holds that $|\mathcal{F}| \geq 2^k$. On the other hand, all frequent set are of size at most k and thus k also induces an upper bound for $|\mathcal{F}|$, namely n^k. This can be seen as follows for the case $k \leq n/2$ (the other case, only a little more complicated, is omitted here for the sake of simplicity).

$$\begin{aligned} |\mathcal{F}| &\leq \binom{n}{k} + \binom{n}{k-1} + \dots + \binom{n}{0} \leq k\binom{n}{k} \\ &= \frac{n(n-1)\dots(n-k+1)}{(k-1)!} \leq n^k \quad . \end{aligned}$$

Now suppose a BP-algorithm $\mathcal{A}$ approximates $\log|\mathcal{F}|$ within n^{δ}. It follows that

$$\begin{aligned} & & k \leq \log|\mathcal{F}| & \quad \leq k\log n \\ \Leftrightarrow & & \log|\mathcal{F}| \,/\log n \leq k & \quad \leq \log|\mathcal{F}| \\ \Leftrightarrow & & \mathcal{A}(\mathcal{D},f)/(n^{\delta}\log n) \leq k & \quad \leq n^{\delta}\mathcal{A}(\mathcal{D},f) \quad . \end{aligned}$$

Now observe that for any $\delta < \delta_{\mathrm{BC}}$, the expression $n^{\delta}\log n$ is asymptotically dominated by $n^{\delta_{\mathrm{BC}}}$; say starting from the constant $n(\delta)$. Choose $\delta_{\#\mathrm{F}}$ to be any number strictly between 0 and δ_{BC}. Then modify $\mathcal{A}$ such that it looks up the true result of all (finitely many) instances of size less than $n(\delta_{\#\mathrm{F}})$ in a hard-coded table. Then $\mathcal{A}$ is a BP-algorithm approximating the maximum cardinality of a frequent set within $n^{\delta_{\mathrm{BC}}}$ as required. □

Although the complexity assumption of this theorem is stronger than $\mathbf{P} \neq \mathbf{NP}$ it is still a widely believed standard assumption. Moreover, non-existence of an α-approximation of the logarithm of a number implies non-existence of an 2^{α}-approximation to the actual number. Thus, we have strong evidence that there is no reasonable approximation algorithm for the general #-FREQUENT SETS problem and in particular no fully polynomial approximation scheme.

That said, there may still be algorithms allowing a good approximation for a wide range of practical relevant datasets. With this in mind, we are going to construct a randomized approximation algorithm in the next section.

5. Monte Carlo Estimation

The perhaps simplest Monte Carlo approach for counting the number of frequent set would be the following: Uniformly generate an element $F \subseteq E$, return 1 if $F \in \mathcal{F}$, and return 0 otherwise. The expected value of this experiment is $|\mathcal{F}| / 2^{|E|}$. Thus, taking the mean of sufficiently many independent repetitions and multiplying it by $2^{|E|}$ is a correct randomized approximation scheme. It is, however, not polynomial. This is due to the fact that $|\mathcal{F}| / 2^{|E|}$ can be as small as $1/2^n$ for an instance of size n. For such instances the expected number of trials before the first 1 turn-out appears is not bounded by a polynomial in n. But as long as the returned result is 0 the solution does not satisfy any relative approximation guarantee and in particular not Equation (1).

5.1. Estimator

The standard solution to the problem above is to *partition* the result into a number of factors, each of which having a reasonable lower bound (see Jerrum and Sinclair, 1997). In our case such a partitioning can simply be done as follows. For $i \in \{1, \dots, n\}$ let

$$\mathcal{F}_i = \{F \subseteq E_i \colon |\mathcal{D}[F]| \geq f\}$$

be the family of frequent sets containing only elements from the first i items (recall that we denote the i-prefix of a set $F \subseteq E$ by F_i). With this we can rewrite the quantity to compute $|\mathcal{F}| = |\mathcal{F}_n|$ as the product

$$|\mathcal{F}_s| \prod_{i=s+1}^{n} \frac{|\mathcal{F}_i|}{|\mathcal{F}_{i-1}|} = |\mathcal{F}_n| \tag{2}$$

with some starting index $s \in \{1 \dots, n-1\}$. It follows directly from the definition that for all $i \leq n$ it holds that $\mathcal{F}_{i-1} \subseteq \mathcal{F}_i$. Let us denote the i-th reciprocal ratio of the product in Equation 2 by

$$r_i = |\mathcal{F}_{i-1}| \,/\, |\mathcal{F}_i| \enspace .$$

A constant non-zero lower bound for the ratios r_i is implied by the following observation: For distinct sets $F \neq F'$ with $F, F' \in \mathcal{F}_i \setminus \mathcal{F}_{i-1}$ the sets $F \setminus \{i\}$ and $F' \setminus \{i\}$ are distinct elements of $\mathcal{F}_{i-1}$. Hence, $|\mathcal{F}_i \setminus \mathcal{F}_{i-1}|$ can be mapped injectively into $|\mathcal{F}_{i-1}|$ implying $|\mathcal{F}_i \setminus \mathcal{F}_{i-1}| \leq |\mathcal{F}_{i-1}|$, and thus it holds that

$$1 \geq |\mathcal{F}_{i-1}| \,/\, |\mathcal{F}_i| = r_i \geq 1/2 \enspace . \tag{3}$$

With this we can design a Monte Carlo algorithm as follows: Approximate each of the ratios r_i, count $|\mathcal{F}_s|$ for an appropriate s exhaustively, and then compute $|\mathcal{F}|$ through Equation (2).

A trivial solution for counting the starting factor $|\mathcal{F}_s|$ is to set $s = 0$ resulting in $|\mathcal{F}_0| = |\{\emptyset\}| = 1$. There are, however, better choices in practice as we will discuss in Section 7.1. The ratios r_i are approximated as follows: assume we can sample a set F from $\mathcal{F}_i$ according to a distribution $\mathcal{D}_i$ satisfying

$$|\mathbb{P}_{F\sim\mathcal{D}_i}[F \in \mathcal{F}_{i-1}] - r_i| \leq b \tag{4}$$

Note that this condition is for instance satisfied by the uniform distribution $\mathcal{U}(\mathcal{F}_i)$ or by a distribution $\mathcal{D}$ having a total variation distance of at most b from the uniform distribution, i.e., $\|\mathcal{D}, \mathcal{U}(\mathcal{F}_i)\|_{\mathrm{tv}} \leq b$. The latter approach is commonly referred to as *almost uniform sampling.* Let Z_i denote the random variable that takes on value 1 if $F \in \mathcal{F}_{i-1}$ and 0 otherwise. Then

$$\bar{Z}_i = (Z_i^{(1)} + \cdots + Z_i^{(t)})/t$$

with $t \geq 1$ and $Z_i^{(j)}$ independent copies of Z_i is a (biased) estimator of r_i satisfying $\left|\mathbb{E}\left[\bar{Z}_i\right] - r_i\right| \leq b$. With this we can write our final estimator for $|\mathcal{F}|$ as $|\mathcal{F}_s|\, Z^{-1}$ where Z denotes the product of all ratio estimators

$$Z = \bar{Z}_{s+1} \ldots \bar{Z}_n \ .$$

It is easy to see that if Z is ϵ-close to the product of all ratios r_i then this implies the same for the complete estimator $|\mathcal{F}_s|\, Z^{-1}$ and $|\mathcal{F}|$, i.e., an algorithm simulating it is a correct randomized approximation scheme. Moreover, as the maximum bias b approaches 0 and the number of independent trials t of each Bernoulli experiment Z_i approaches infinity that guarantee will eventually hold.

5.2. Performance

In this subsection we discuss for what values of b and t an algorithm that simulates the estimator $|\mathcal{F}|\, Z^{-1}$ is a correct randomized approximation scheme for $|\mathcal{F}|$ as specified by Equation 1.

We start with the bias b and the relative deviation it causes for the mean of the ratio estimators $\bar{Z}_i$. It follows from the absolute deviation as specified in Equation (4) and the lower bound of r_i from Equation (3) that

$$(1-2b)r_i \leq \mathbb{E}\left[\bar{Z}_i\right] \leq (1+2b)r_i \ .$$

This can be used to bound the deviation of Z's mean from the product $r = \prod_{i=s+1}^{n} r_i$ of all reciprocal ratios from Equation 2. Suppose we enforce a maximum absolute bias

$$b \leq \mathrm{bias}(n, s, \epsilon) = \epsilon/12(n-s).$$

Then using the independence of the random variables Z_i and basic bounds of the exponential function we can deduce:

$$\mathbb{E}[Z] = \prod_{i=s+1}^{n} \mathbb{E}\left[\bar{Z}_i\right] \leq \left(1 + \frac{\epsilon}{6(n-s)}\right)^{n-s} r \leq \exp\left(\frac{\epsilon}{6}\right) r \leq \left(1 + \frac{\epsilon}{5}\right) r$$

If in addition $Z \leq (1 + \epsilon 2/3)\mathbb{E}[Z]$ we arrive at the required upper bound for the deviation of Z from r

$$Z \leq \left(1 + \frac{1}{5}\epsilon\right)\left(1 + \frac{2}{3}\epsilon\right) r < \left(1 + \frac{2\epsilon}{3} + \frac{\epsilon}{5} + \frac{2\epsilon}{15}\right) r = (1+\epsilon) r$$

and the lower bound follows similarly.

It remains to assure that Z is $2\epsilon/3$-close to its mean with probability at least $3/4$. The required number of trials t can be calculated by instantiating Chebycheff's inequality as follows

$$\mathbb{P}\left[|Z - \mathbb{E}[Z]| \geq \frac{2}{3}\epsilon\mathbb{E}[Z]\right] \leq \frac{9\mathbb{V}[Z]}{4\epsilon^2\mathbb{E}[Z]^2} \ . \tag{5}$$

So a bound on this probability can be established by appropriately bounding the ratio of Z's variance to the square of its expectation. For the estimator of each factor $\bar{Z}_i$ we know that

$$\begin{aligned}\frac{\mathbb{V}[\bar{Z}_i]}{\mathbb{E}[\bar{Z}_i]^2} &= \frac{\mathbb{E}[\bar{Z}_i]\,(1-\mathbb{E}[\bar{Z}_i])}{t\mathbb{E}[\bar{Z}_i]^2} = \frac{1}{t}(\mathbb{E}[\bar{Z}_i]^{-1} - 1)\\ &\leq \frac{1}{t}\left(\left(\frac{1}{2} - b\right)^{-1} - 1\right) = \frac{1+2b}{t(1-2b)}\end{aligned}$$

where the first equality uses the fact that $\bar{Z}_i$ follows a normalized binomial distribution, and the last inequality follows from Equations (3) and (4). Thus, if we set the number of trials t to

$$t = \mathrm{trials}(s,\epsilon) = \left\lceil \frac{10(n-s)(1+2b)}{(1-2b)\epsilon^2} \right\rceil$$

we can deduce for the product:

$$\begin{aligned}\frac{\mathbb{V}[\bar{Z}_{s+1}\dots\bar{Z}_n]}{(\mathbb{E}[\bar{Z}_{s+1}]\dots\mathbb{E}[\bar{Z}_n])^2} &= \frac{\mathbb{E}[(\bar{Z}_{s+1}\dots\bar{Z}_n)^2]}{(\mathbb{E}[\bar{Z}_{s+1}]\dots\mathbb{E}[\bar{Z}_n])^2} - 1 = \frac{\mathbb{E}[\bar{Z}_{s+1}^2]\dots\mathbb{E}[\bar{Z}_n^2]}{(\mathbb{E}[\bar{Z}_{s+1}]\dots\mathbb{E}[\bar{Z}_n])^2} - 1\\ &= \prod_{i=s+1}^{n}\left(1 + \frac{\mathbb{V}[\bar{Z}_i]}{\mathbb{E}[\bar{Z}_i]^2}\right) - 1 \leq \left(1 + \frac{1+2b}{t(1-2b)}\right)^{n-s} - 1\\ &= \left(1 + \epsilon^2/10(n-s)\right)^{n-s} - 1 \leq \exp\left(\epsilon^2/10\right) - 1\\ &\leq (\epsilon^2/9) \ .\end{aligned}$$

Plugging this bound into Equation (5) it follows that Z is $2\epsilon/3$-close to its mean with probability at least $3/4$ as required.

Note that there is a tradeoff between the constant appearing in the minimum number of trials and that in the bias bound. Thus, in case one can assure a stricter bias bound for a similar cost it is worthwhile to recompute the corresponding trial number.

6. Frequent Set Sampling

In the naive Monte Carlo algorithm sketched in the beginning of Section 5 the necessary number of trials was prohibitive, while the required uniform sampling from the power set did not pose a problem. Now the situation is different: The required number of trials is polynomially bounded, but it is unclear how to sample from the frequent set families $\mathcal{F}_i(\mathcal{D}, f)$ for $i = s+1, \dots, n$ according to a distribution $\mathcal{D}_i$ satisfying Equation 4 for $b = \epsilon/12(n-s)$ as required for estimating the factors r_i. In fact a general sampling algorithm satisfying this condition—in particular uniform or almost uniform sampling algorithms—with a worst-case polynomial running time cannot be expected to exist in the light of Theorem 3 and the reduction of counting frequent sets to sampling them from the previous subsection. Therefore we design a sampling algorithm that performs well in practice while its worst-case convergence time is exponential.

6.1. Markov Chain

We approach this problem by setting $F_0 = \emptyset$ and then repeatedly applying the following randomized perturbation procedure $F_j \mapsto F_{j+1}$:

1. set F' to F_j
2. uniformly draw an $k \in \{1, \ldots, i\}$
3. if $k \in F'$ set F' to $(F' \setminus \{k\})$; otherwise:
4. if $(F' \cup \{k\}) \in \mathcal{F}$ then set F' to $(F' \cup \{k\})$.
5. with probability 1/2 set F_{j+1} to F_j; otherwise: set F_{j+1} to F'

This procedure simulates one step of a Markov chain $\mathfrak{M}_i$ on $\mathcal{F}_i$ with state transition probabilities

$$P_i(F, F') = 1/2i \text{ for } F, F' \in \mathcal{F} \text{ with } |F \Delta F'| = 1$$

where Δ denotes symmetric difference. All “remaining” probability is assigned to the self-loops, i.e., $P_i(F, F) = 1 - |\{F' \in \mathcal{F} \colon |F \Delta F'| = 1\}| / 2i$. So the transition probabilities and thus the corresponding state transition matrix P_i as well as the reachability relation of $\mathfrak{M}_i$ are symmetric.

Together with the fact that $\mathcal{F}$ is closed under taking subsets this implies that $\mathfrak{M}_i$ is irreducible because all states are reachable from $\emptyset$. Moreover, there are non-zero self-loop probabilities for every state. This implies that $\mathfrak{M}_i$ is also aperiodic and together with irreducibility this means that $\mathfrak{M}_i$ is ergodic.

For an ergodic Markov chain we know that there is a unique distribution π that it converges to and that is stationary, i.e., $P_i \pi = \pi$. Since P_i is symmetric, π must be the uniform distribution on $\mathcal{F}_i$. Hence, simulating $\mathfrak{M}_i$ for sufficiently many steps can be used to sample a frequent set from $\mathcal{F}_i$ uniformly at random as sufficient for satisfying Condition (4).

The question is, however, for how many steps l we have to simulate $\mathfrak{M}_i$ until $P_i^l(\emptyset, \cdot)$ is “close enough” to the uniform distribution. The standard approach in approximate counting (by almost uniform sampling) is to derive an upper bound on the mixing time $\tau(\epsilon/12(n-s))$ and then use this upper bound to compute the necessary number of simulation steps. The resulting distribution is guaranteed to satisfy Condition (4) because the total variation distance from the uniform distribution is an upper bound to the maximum bias b. As stated earlier, for our problem this approach is infeasible: in line with Theorem 3, a good worst-case bound to the mixing time τ cannot be expected. Indeed, we can observe the following:

Proposition 4. For $n \in \mathbb{N}$ the Markov chain $\mathfrak{M}_{2n}$ with frequency threshold $f = 1$ and $\mathcal{D} = \{\{1\}, \{2, \ldots, n\}, \{n+2, \ldots, 2n\}\}$ on items $E = \{1, \ldots, 2n\}$ has mixing time $\tau(\epsilon')$ of at least $2^{n-1} \log(1/2\epsilon')$.

Proof. Let $\mathcal{P}(X)$ denote the power set of a set X. For $n \in \mathbb{N}$ the 1-frequent sets of the dataset given in the claim are

$$\mathcal{F} = \{\{1\}\} \cup \mathcal{P}(\{2, \ldots, n\}) \cup \mathcal{P}(\{n+2, \ldots, 2n\})$$

with the cardinality $|\mathcal{F}| = 2^n$. It is a well-known fact (see for instance Randall, 2006) that the *conductance* of a Markov chain induces a lower bound on its mixing time. The conductance of $\mathfrak{M}_{2n}$ is defined as

$$\Phi = \min\{\Phi_S \colon \mathcal{S} \subseteq \mathcal{F}(\mathcal{D}, 1),\ \pi(\mathcal{S}) \leq 1/2\}$$

where $\Phi_{\mathcal{S}} = (1/\pi(\mathcal{S})) \sum_{x \in \mathcal{S}, y \in \mathcal{F} \setminus \mathcal{S}} \pi(x) P(x, y)$ is the probability according to the stationary distribution of leaving $\mathcal{S}$ given the current state is an element of $\mathcal{S}$. The mixing time is then bounded from below as follows:

$$\tau(\epsilon) \geq \frac{1}{4\Phi} \log \frac{1}{2\epsilon} \quad . \tag{6}$$

Now choose $\mathcal{S} = \{\{1\}\} \cup \mathcal{P}(\{2, \ldots, n\}) \setminus \{\emptyset\}$. Then $|\mathcal{S}| = 2^{n-1}$ and consequently $\pi(\mathcal{S}) = 1/2$.

$$\begin{aligned} \Phi \leq \Phi_{\mathcal{S}} &= \frac{\sum_{x \in \mathcal{S}, y \notin \mathcal{S}} \pi(x) P(x, y)}{\pi(\mathcal{S})} = 2 \sum_{x \in \{\{1\}, \ldots, \{n\}\}} \pi(x) P(x, \emptyset) \\ &= 2n \frac{1}{2^n} \frac{1}{2(2n)} = \frac{1}{2^{n+1}} \end{aligned}$$

Plugging this bound into Equation 6 yields the claim. □

Intuitively, the reason for the slow mixing time on these instances is that the probability of crossing over from one of the two "blocks" $\mathcal{P}(\{2, \ldots, n\})$ and $\mathcal{P}(\{n+1, \ldots, 2n-1\})$ to the other is very low compared to their sizes.

6.2. Heuristic Step Bound

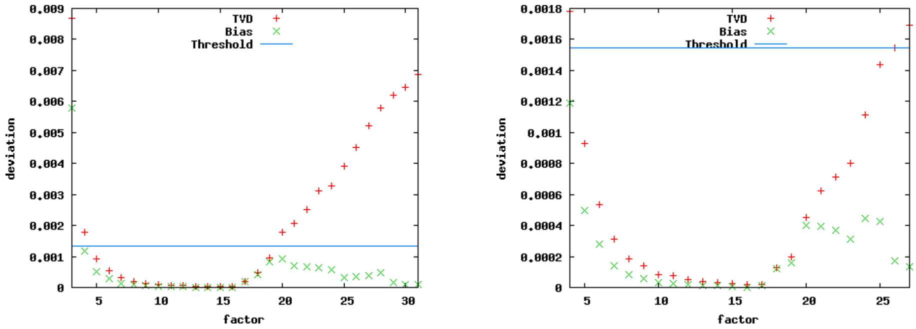
In order to circumvent the negative implications of Proposition 4 we will simulate the Markov chain only for a heuristic number of steps—a number much smaller than the best theoretical worst-case bound on the mixing time would yield. The justification for this approach is twofold:

- The situation constructed in the proof of Proposition 4 is obviously rather artificial and it is a reasonable assumption that most real-world datasets do not possess such strictly separated blocks.
- The mixing time is equal to a guaranteed number of steps such that the total variation distance is close to the uniform distribution. This distance, however, is only an upper bound to the bias term, and, as we will see below, in practice the bias can be much smaller than the total variation distance.

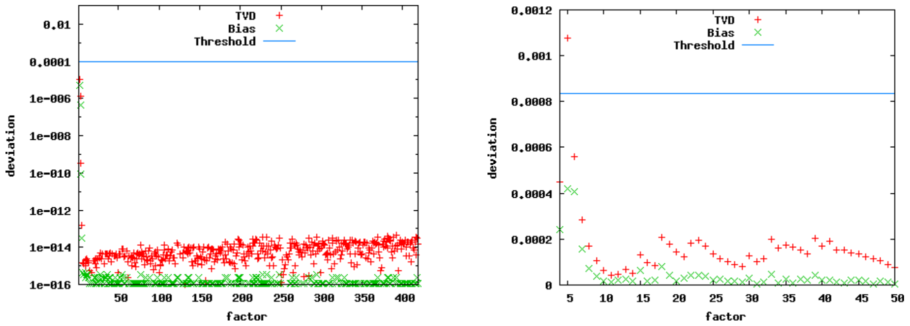
The heuristic for the number of steps we simulate the chain $\mathfrak{M}_i$ is

$$\mathrm{steps}(i, \epsilon) = 2\epsilon^{-1} i \ln i \ ,$$

i.e., we use $P_i^l(\emptyset, \cdot)$ with $l = 2\epsilon^{-1} i \ln i$ in place of the distributions $\mathcal{D}_i$. This is of the same order as the expected number of steps until each item has been drawn at least once (coupon collector's theorem)—a reasonable minimum requirement. Clearly, there are other possible choices for $\mathrm{steps}(i, \epsilon)$; in particular when there is prior knowledge of the input dataset.



(a) **‘chess’** (b) **‘connect’**

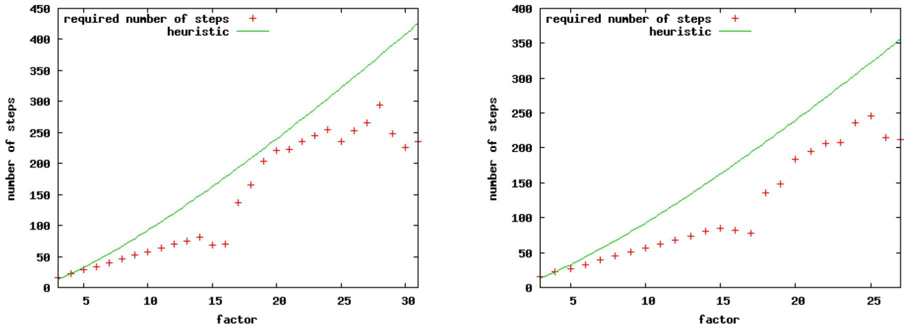


(c) **‘bms webview-1’** (d) **‘pc60pa10’**

Fig. 2. biases

We evaluated this heuristic using $\epsilon = 0.5$ on several test datasets (see Section 8 for a description of these datasets). The relative thresholds used and the resulting numbers of frequent sets are 0.62/166581 (*chess*), 0.81/375384, (*connect*), 0.03/420 (*bms webview*), and 0.2/22021 (*pc60pa10*), respectively. These moderate state space sizes allowed us to explicitly compute the distributions $P^t(\emptyset, \cdot)$. Figure 2 shows the results. The exact resulting biases (blue crosses) are compared to the required maximum bias $b = 1/24n$ (blue line) for each of the ratios r_i. Also, these figures show the total variation distance from the uniform distribution (red crosses). Indeed we can observe a significant gap between the quantities for some of the factors. Consequently, the total variation distances sometimes violate the maximum bias requirement while the actual biases do always satisfy this condition.

To give an idea of how conservative the heuristic is we created another diagram, which is presented in Figure 3. Here, the minimum number of steps that is resulting in order to satisfy the bias requirement (red crosses) is compared to the number of steps computed by the heuristic (green line). We can observe that the heuristic is much less conservative for the first factors as it is for the later factors. This is due to the fact that the heuristic number of steps does not depend on the overall total number of factors while the maximum allowed bias does. Instead the heuristic only depend on the index of the ratio to be approximated, which is approaching the number of factors only for the later ratios. As we will see below



(a) **‘chess’** (b) **‘connect’**

Fig. 3. required steps

this is no problem for our method, because finally we will end up with a hybrid counting algorithm that does not approximate the first factors at all.

7. Algorithmic Details and Improvements

The basic algorithm assembled from the partitioning from Section 5 and the sampling from Section 6 has on overall time complexity of $O\big(\epsilon^{-3}mn^3\ln n\big)$ if we denote by m the number of transactions $|\mathcal{D}|$ and account $O(mn)$ for one frequency check. While this time complexity does not depend on the number of frequents sets as desired, it performs rather poorly with respect to the input size. In this section we present several techniques that lead to a substantial performance improvement in practice. That said, they do not affect the overall asymptotic behavior.

A pseudocode incorporating all ideas is given with Algorithm 1. Note that the requirement stated there is a little stronger than necessary in that it demands the bias bound to hold for all ratios. In fact the bound is only needed for those ratios r_i that are actually approximated, i.e., for $i > s$, and usually s is chosen larger than 0 (see Section 7.1).

7.1. Hybrid Counting

Our estimator requires a starting factor $|\mathcal{F}_s|$ that has to be counted exhaustively. As indicated in Section 5.1 there are more effective choices for s than just choosing $|\mathcal{F}_0| = 1$. This is due to the fact that exhaustive counting of $\mathcal{F}_s$ replaces the estimation of the ratios $r_1, \dots, r_s$, each of which having a time complexity that depends on the complete number of items n. In contrast exhaustive counting of $\mathcal{F}_s$ only runs on a reduced dataset and has to enumerate at most 2^s frequent sets. This is more efficient as long as s is sufficiently small.

Deciding what choice for s will result in the minimum total running time requires some analysis of the involved time complexities. For that we regard one frequency check as unit step. For ease of notation we will drop all symbols that are constant throughout one call of the algorithm from the parameter lists of trials$(\cdot)$ and steps$(\cdot)$.

Algorithm 1 Frequent Set Counting

```
Input   : dataset $\mathcal{D}$ on items $E = \{e_1, \ldots, e_n\}$,
          frequency threshold $f \in \{1, \ldots, |\mathcal{D}|\}$,
          accuracy $\epsilon \in [0, 1]$
Require: $\left|\mathbb{P}_{F \sim P_i^l(\emptyset,\cdot)}[F \in \mathcal{F}_{i-1}] - r_i\right| \leq \epsilon/12n$ with $l = \mathrm{steps}(i, \epsilon)$
Output : $q$ with $\mathbb{P}[(1-\epsilon)\,|\mathcal{F}| \leq q \leq (1+\epsilon)\,|\mathcal{F}|] \geq 3/4$

main:
//exhaustive phase
$s \leftarrow \mathrm{argmin}_{s' \leq n}(2^{s'} + T_{\mathrm{apx}}(s'))$
$S \leftarrow \mathrm{exhaustive}(\mathcal{D}|_s, f)$
while $S < T_{\mathrm{apx}}(s) - T_{\mathrm{apx}}(s+1)$ do
    $S \leftarrow S + \mathrm{exhaustive}(\mathcal{D}[\{s+1\}]\,|_s, f)$
    $s \leftarrow s + 1$

//approximative phase:
$\mathcal{I} \leftarrow \emptyset$
for $i = s+1, \ldots, n$ do
    $r_i \leftarrow 0$
    for $k = 1, \ldots, \mathrm{trials}(s, \epsilon)$ do
        if $i \notin \mathrm{sample}(i)$ then $r_i \leftarrow r_i + 1$
    $r_i \leftarrow r_i/\mathrm{trials}(s, \epsilon)$
return $S \prod_{i=s}^{n} r_i^{-1}$

sample($i$):
$F \leftarrow \emptyset$;
buffer $\leftarrow$ generateRandString(steps($i, \epsilon$))
for $j = 0, \ldots,$ "last index of $i$ in buffer" do
    $e \leftarrow$ buffer[$j$]
    if $e \in F$ then
        $F \leftarrow F \setminus \{e\}$
    else if frequencyCheck($F, e$) then
        if $j =$ "next to last index of $i$ in buffer" then return $F$
        $F \leftarrow F \cup \{e\}$
return $F$

frequencyCheck($F, e$):
$L \leftarrow$ node list of $\max\{F \cup \{e\}\}$
if $|F| \geq \min_{I \in \mathcal{I}} |I| \wedge |L| > \mathrm{costEst}(\mathcal{I}, I)$ then
    if $F \in \mathcal{I}$ then return true
if $|\mathcal{D}[F \cup \{e\}]| \geq f$ then
    return true
else
    $\mathcal{I} \leftarrow \mathcal{I} \cup \{F \cup \{e\}\}$
    return false
```

Estimating r_i requires at most trials(s)steps(i) frequency checks. On the other hand, counting $|\mathcal{F}_s|$ exhaustively can take up to 2^s frequency checks in case all subsets of E_s are frequent. This rough bound can be used to make an initial choice for s by choosing it such that it minimizes the estimated overall running time $2^s + T_{\text{apx}}(s)$ with

$$T_{\text{apx}}(s) = \text{trials}(s) \sum_{i=s+1}^{n} \text{steps}(i)$$

denoting the expected time for approximating all remaining factors. Clearly, more knowledge about the used implementations—in particular that of the exhaustive miner—is likely to lead to an improvement of this choice.

Additionally, we can further improve our choice of the starting index s. Denote the index found by the considerations above as s^*. The loose a priori bound used there can be improved once we have counted $\mathcal{F}_{s^*}$. We know that $|\mathcal{F}_{s^*+1} \setminus \mathcal{F}_{s^*}| \leq |\mathcal{F}_{s^*}|$ (see Equation 3). Thus, in case

$$|\mathcal{F}_{s^*}| < T_{\text{apx}}(s^*) - T_{\text{apx}}(s^* + 1)$$

we can increase s to $s^* + 1$ and add $|\mathcal{F}_{s^*+1} \setminus \mathcal{F}_{s^*}|$ counted exhaustively to $|\mathcal{F}_s|$. Clearly, as long as the above condition remains true for the new s it amortizes to repeat this step. The resulting algorithm counts $\mathcal{F}_s$ for the final s “chunk-wise” as

$$|\mathcal{F}_s| = |\mathcal{F}_{s^*}| + |\mathcal{F}_{s^*+1} \setminus \mathcal{F}_{s^*}| + \cdots + |\mathcal{F}_s \setminus \mathcal{F}_{s-1}| \quad .$$

For the exhaustive counting tasks it is desirable to use one of the existing highly optimized frequent set listing algorithm. Let $\mathcal{D}|_i$ denote the dataset in which all transactions have been restricted to the first i items. Observe that $|\mathcal{F}(\mathcal{D}[\{s\}]\,|_{s-1})|$ is equal to $|\mathcal{F}_s(\mathcal{D}) \setminus \mathcal{F}_{s-1}(\mathcal{D})|$. Thus, one external call of the exhaustive miner with the dataset $\mathcal{D}[\{s\}]\,|_{s-1}$ suffices to compute the latter quantity.

7.2. Basic Chain Simulation Speedups

The central part of the approximative phase of Algorithm 1 are the Markov chain simulations. As a first optimization it is possible to do better than just simulating $\mathfrak{M}_i$ for $i = s+1, \ldots, n$ naively. Instead, we first buffer the required number steps(i, ϵ) of random items for a single random walk. Since the result F of a random walk is only used to evaluate the Bernoulli experiment Z_i of whether item i is an element of F, we can simply stop the Markov chain simulation after the last occurrence of i in the buffer, because afterwards the outcome of the experiment cannot change anymore. Similar, if i was put into F due to the next to last occurrence of i we can also stop the chain simulation at that point. Since in this case the last occurrence of i will surely cause the item to be removed from F again, we can directly report that result.

The really dominant operation, however, is the test of whether a set $I = F \cup \{e\}$ that is an augmentation of a frequent set F with a single item e remains frequent. This corresponds to step 4 of the Markov chain (see Section 6). Since this test has to be performed roughly every second step of each random walk, it is crucial for the overall performance of our algorithm. In order to decrease the cost of this operation, our implementation makes use of the data compression that is enabled by representing the dataset $\mathcal{D}$ as an fp-tree (Han and Kamber, 2000). An fp-tree is a prefix tree $T = (V, E, \phi)$ where, in addition, there are node lists L_i containing all nodes corresponding to prefixes $X \in V$ with $\max X = i$. Since overlapping transactions share a prefix-branch, this usually achieves a significant compression.

The frequency test for a set is done by selecting its least frequent item $\max I$, following its node list in the fp-tree, and adding the counts $\phi(X)$ of all nodes $X \in L_{\max I} = L$ with $X \supseteq I$, i.e., those corresponding to a prefix that contains I. The traversal of the node list can be stopped already after an initial part $L' \subseteq L$ in case the frequency state of I can already be determined by L', that is either if $l = \sum\{\phi(X)\colon\ X \in L',\ X \supseteq I\} \geq f$ returning "true", or if $l + \sum\{\phi(X)\colon\ X \in L \setminus L'\} < f$ returning "false" where the last sum can be computed by $|\mathcal{D}[\max I]| - \sum\{\phi(X)\colon\ X \in L'\}$.

7.3. Infrequent Set Cache

Even with the representation of $\mathcal{D}$ by an fp-tree, frequency checks remain an expensive operation. In order to reduce the computational effort for these operations one can store the infrequent sets that are visited throughout the various random walks. The thus created cache can then be used to quickly check whether $I = F \cup \{e\}$ is a superset of an already visited infrequent set I'—possibly avoiding the expensive frequency check. This test is invoked in line 2 of the frequency check procedure of Algorithm 1. Below we discuss how this step can be implemented.

Let us denote the family of visited infrequent sets by $\mathcal{I}$. Again it is useful to represent this family by a prefix tree. A superset test can then be implemented as a depth first search traversal of the tree that recurses only into child nodes via edges corresponding to items that are contained in I. Moreover, we can ignore edges corresponding to items $e' > e$ as long as the current node does not contain item e. Below this edge we can only find sets not containing e, non of which can be a subset of I. This is because we know that $I \setminus \{e\} = F$ is frequent and, thereby, can never be superset of an infrequent set.

It is, however, not always reasonable to perform this infrequency check. There are candidate sets I that require only very few nodes in the fp-tree to be checked using the standard frequency check; those with a relatively small $\max I$. As we assume the items to be ordered according to their frequency, these candidates are in addition rather unlikely to be infrequent. In order to decide for what candidate sets infrequency should be checked against $\mathcal{I}$ it is necessary to estimate the expected look-up time. For that observe that the recursion depth of the DFS is bounded by $\min\{|I|, d\}$ where $d = \max_{I' \in \mathcal{I}} |I'|$ is the depth of the prefix tree. Moreover, in a node X with outgoing edges $\delta^+(X)$ at level i the number of recursive calls is limited by $\min\{|I| - i, |\delta^+(X)|\}$. The overall cost for looking up the set I in the cache $\mathcal{I}$ can then be estimated as follows:

$$\mathrm{costEst}(I, \mathcal{I}) = \prod_{i=1}^{\min\{|I|, d\}} \min\{\Delta, |I| - i\}$$

where Δ is the average outdegree of the prefix tree representing $\mathcal{I}$. This quantity is compared to the number of nodes corresponding to $\max I$ in the fp-tree of $\mathcal{D}$ in line 2 of the frequency check procedure of Algorithm 1.

dataset	up. qrt.	median	lw. qrt.	max.	fail
bms-pos	0.003	0.004	0.004	0.04	0
bms-web	0.002	0.003	0.003	0.04	0
chess	0.011	0.025	0.056	0.17	0
connect	0.02	0.047	0.081	0.21	0
mushroom	0.001	0.010	0.024	0.069	0
pumsb	0.017	0.041	0.080	0.314	0
pumsb-star	0.0001	0.006	0.013	0.083	0
pc60pa10	0.009	0.025	0.037	0.134	0

Fig. 4. relative deviation from exact solution

8. Evaluation

In this section we present experiments contrasting the Markov chain Monte Carlo algorithm with counting via exhaustive enumeration. The experiments are performed with respect to performance as well as accuracy ($\epsilon = 0.5$ was used throughout all experiments). As a representative exhaustive miner, we used the modified FPgrowth (Han and Kamber, 2000) algorithm by Grahne and Zhu (2003), whose C++ implementation is publicly available. This implementation has shown to rank among the fastest exhaustive miners in the competitive workshop FIMI (Bayardo, Goethals and Zaki, 2004). In the following, we will refer to this implementation as "FpZhu". The benchmark datasets are also taken from the FIMI repository supplemented by synthetic datasets generated according to the beginner's guide process (Example 2) with different choices for the probabilities p_a and p_c each of which with 5000 transactions. We refer to these dataset as *pc60pa10* and *pc90pa10*, because they are generated by instantiations of this general process with probabilities $p_a = 0.1$ and $p_c = 0.6$respectively $p_c = 0.9$. For Algorithm 1 we used a Java implementation that is available online together with the artificial datasets (http://www-kd.iai.uni-bonn.de/index.php?page=people_details&id=16 or corresponding main page). During the exhaustive phase the external miner called is again FpZhu. The experiments were performed on an Intel Core 2 Duo E8400 with 3 GB of RAM running Windows XP. All figures below use relative frequency thresholds.

8.1. Accuracy

We now report a series of accuracy experiments that are summarized in Figure 4. Our algorithm was applied 100 times to each combination of one of eight test datasets with one of four frequency thresholds, resulting in a total of 400 runs per dataset. A run "fails" if the reported result deviates from the true number of frequent sets by more than $\epsilon = 0.5$, which we used as accuracy parameter throughout all experiments. If the overall approximation guarantee holds we expect the fraction of failed runs to be below 1/4 (Equation 1). As desired, this was the case on all of the eight test datasets. In fact, we had *no failed* runs among all experiments. Thus, the observed success rate was consistent with the probabilistic approximation guarantee for all datasets, and in fact we experienced much better error bounds. The table also shows the upper quartile ("up. qrt."), median ("median"), lower quartile ("lw. qrt."), and maximum ("max.") of the experienced relative deviation.

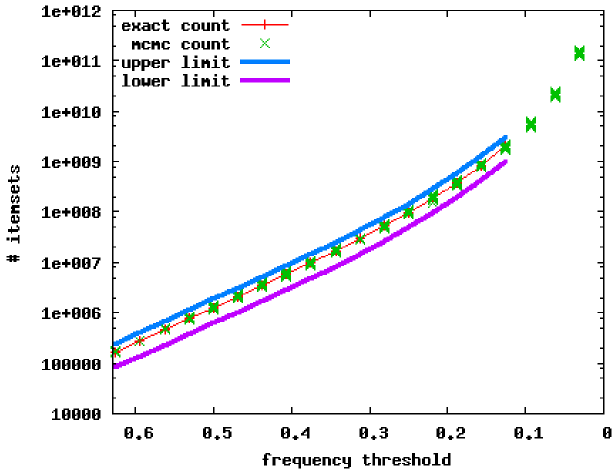


Fig. 5. accuracy on 'chess'

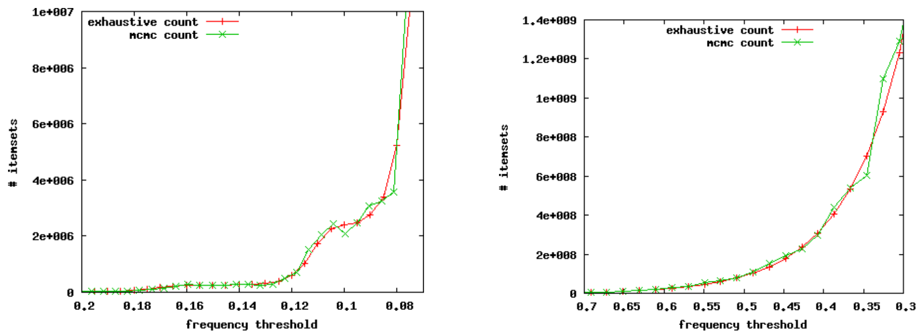
These results may lead to the impression that the upper absolute bias bound b and the number of trials t have been chosen to conservative in Section 5.1 and less restrictive numbers would still yield the overall approximation guarantee. But in fact this good performance is partially caused by mutual cancellation of the actual bias terms, i.e., some of them are negative and some are positive. Moreover, the number of trials t takes into account the worst-case $r_i = 1/2$. For the test datasets, however, many of the ratios r_i are close 1.

Figure 5 shows accuracy results on *chess* in more detail. For different frequency thresholds, it shows on a logarithmic scale the exact number of sets ("exact count") computed exhaustively, the upper and lower 0.5 deviation limits ("upper limit" and "lower limit"), as well as the result of the randomized algorithm in a series of 15 runs ("mcmc count"). The figure shows that in all randomized runs, the approximated result lies in the desired deviation interval. The figure does not contain the exact number of sets for the lowest thresholds, because these could not be computed by the exhaustive miner.

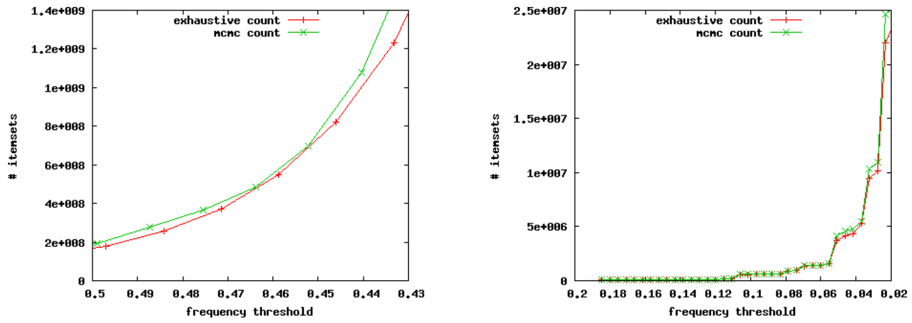
We close this subsection by presenting approximated frequency plots computed for four of the datasets. The results are shown in Figure 6 together with the exact curve. They illustrate that the randomized algorithm provides an adequate approximation of the exact plot, which was one motivating goal of this study (see Section 2).

8.2. Runtime

Next, we compare the runtime of our randomized algorithm with that of FpZhu on several datasets. We start with the results on *chess*, which are presented in Figure 7(a). In order to give an idea of the quality of FpZhu we also added the performance curve of an Apriori implementation (Bodon, 2003), which was another contender in the FIMI workshop. But most important the diagram shows that while on higher thresholds FpZhu is faster than our randomized algorithm, the latter outperforms the exact algorithm when the threshold becomes smaller.



(a) **‘pc60pa10’** (b) **‘connect’**



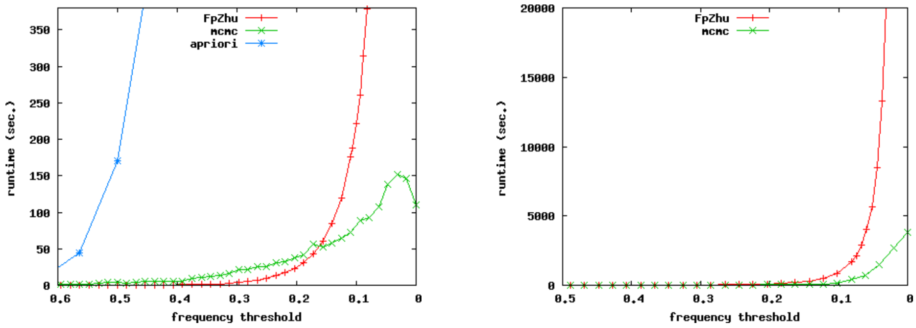
(c) **‘pumsb’** (d) **‘mushroom’**

Fig. 6. frequency plots

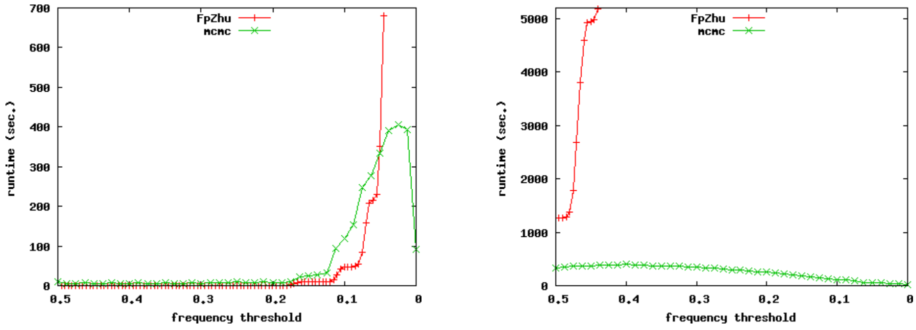
A similar behavior can be observed for *connect* (Figure 7(b)), *pc60pa10* (Figure 7(c)), and *pumsb* (Figure 7(e)). On *pc90pa10* (Figure 7(d)), which is extremely dense and as a result has a very high number of frequent sets, the randomized approach outperforms the exhaustive miner on all threshold. In fact, for all but the highest thresholds the runtime of the exhaustive miner becomes unacceptable.

However, the sparser the dataset is the later the randomized algorithm catches up to the exhaustive miner. On *mushroom* (Figure 7(f)) it is even dominated on all thresholds. Hence, it heavily depends on the dataset's density (the number of frequent sets) whether the randomized or the exhaustive approach performs better. This can roughly be explained and summarized as follows: While the complexity of the Monte Carlo algorithm, in contrast to any exhaustive miner, does not depend on the output size, it does not scale as well as the latter in the input size (see also the discussion in Section 9).

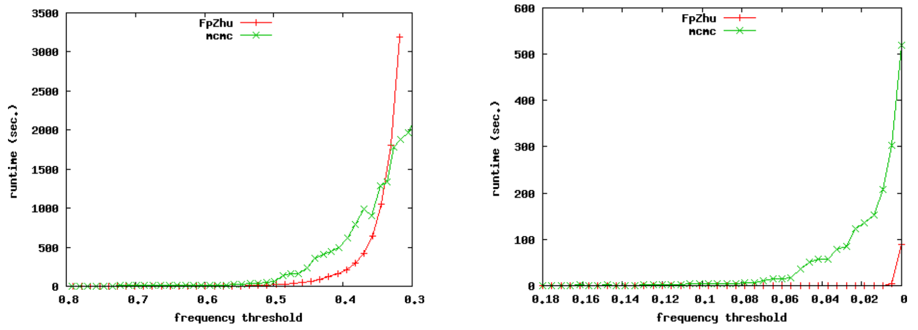
We close the performance study with the comparison of Algorithm 1 to implementations that a) do not use the infrequent set cache and b) that do neither use hybrid counting nor the infrequent set cache. Figure 8 shows the impact of these two speedups.



(a) **'chess'** (b) **'connect'**



(c) **'pc60pa10'** (d) **'pc90pa10'**

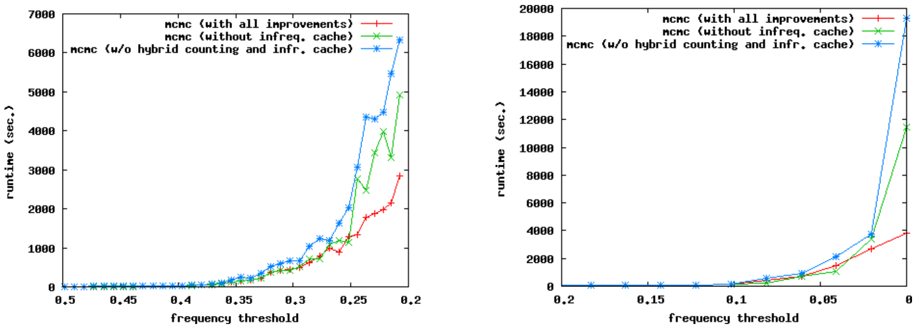


(e) **'pumsb'** (f) **'mushroom'**

Fig. 7. runtime comparisons

9. Conclusion

In this section we give a summarizing discussion of our study that emphasizes both: its contributions as well as its current limitations. Thereafter we conclude by presenting ideas for future research. In this course we outline promising directions to overcome the said limitations.



(a) **'pumsb'** (b) **'connect'**

Fig. 8. runtime using different improvements

9.1. Summary and Discussion

In this paper, we developed a randomized approximation scheme for counting the number of frequent sets using the Markov chain Monte Carlo method. It relies on sampling frequent sets in a way that satisfies a certain bias bound. We gave a corresponding sampling procedure that, albeit meeting this bound, in worst-case does so only after an exponential number of iterations. For that reason we switched to a heuristic but polynomially bounded number of steps that we validated on several test datasets. Although this heuristic approach works well on these datasets, by applying it, the overall counting algorithm loses its worst-case approximation guarantee. We have shown, however, by giving a negative complexity result that a general polynomial algorithm with a good approximation guarantee is unlikely to exists. Moreover, we experienced very good approximation rates on real-world and artificial dataset. In order to improve the performance of our method we described several techniques that can significantly speed up its running time when compared to a naive implementation. We demonstrated that for dense datasets/low frequency thresholds the randomized algorithm remains well applicable while exhaustive counting is infeasible.

Altogether our study should be regarded as an initial work on a computational problem that so far has seen little to no attention. As such its practical applicability, for instance as a general preprocessing method to frequent set mining, still suffers from serious limitations:

(1) The method scales very badly in the number of items, i.e., super-cubic, which makes it already inapplicable on current personal computers as that number approaches 1000.

(2) Even on datasets with a small number of items it is often outperformed by exhaustive counting. Particularly this is the case for sparse datasets and/or low frequency threshold.

While the first point appears to be inherent to the method and its circumvention is likely to require fundamental additional ideas, the second can probably be solved by some straightforward additions (see the paragraph about integrated exhaustive counting below). Beside larger constant factors in its complexity the relative weakness of the current implementation on some datasets/thresholds can be explained as follows: exhaustive mining is a systematic process that can optimally make use of reduced data portions. In contrast, for randomized counting it is inherent that the part of the data that is needed next is unpredictable. Consequently, the randomized method suffers more from the size of the input data (here, in particular from the number of transactions) than the exhaustive method does.

9.2. Directions for Future Research

Integrated Exhaustive Counting In principle issue (2) above is addressed by the hybrid counting technique presented in Section 7.1. But the approach of "plugging in" any exhaustive miner in our current implementation is causing a lot of unnecessary overhead by external calls and disk accesses. Clearly, this can be tackled by an integrated and well implemented exhaustive counting algorithm that, in addition, should have an internal enumeration order compatible to the subfamilies $\mathcal{F}_1, \mathcal{F}_2, \ldots$. Such an algorithm could pass on control to the approximative algorithm as soon as the cost of the next search level would outweigh the cost of approximating the next factor, thus finding an optimal starting index s. It is likely that this improvement alone leads to a hybrid counting algorithm that is never significantly outperformed by any exhaustive miner.

Improved Lower Bounds Finding better lower bounds for the means of the random variables $\bar{Z}_i$ respectively for the ratios of their variance to their squared expectation would allow to reduce the required number of trials, and thus to reduce the constants. This can either be done by monitoring these quantities empirically in a sequential sampling fashion (cp. Scheffer and Wrobel, 2002) or by an exhaustive analysis of certain parts of the input dataset. Specifically for the task of computing the complete frequency plot, i.e., solving #-FREQUENT SETS for all thresholds, one can acquire a lower bound for $\mathcal{F}_i(f)/\mathcal{F}_{i+1}(f)$ already while sampling from $\mathcal{F}_{i+1}(f-1)$ during the run for threshold $f-1$.

Better Scaling in the Number of Items One approach to achieve a better scaling in the number of items is to reduce the data in a way that (approximately) preserves the number of frequent sets. An example for this is the sketch matrix technique of Jin, McCallen, Breitbart, Fuhry and Wang (2009). Essentially, this method performs a bi-clustering on the items and transactions, and then approximates the number of frequent sets only based on statistics that are computed for each bi-cluster.

Another approach is to redefine the partitioning such that not only one but a block of k items is introduced at each level, i.e., $\mathcal{F}_i = \{1, \ldots, ik\}$. While this results in weaker lower bounds per factor, which ideally should be addressed by empirical estimates (see the paragraph above), it divides the number of factors by k and consequently, also reduces the requirement on the maximum bias.

Using More Structural Properties It is important to point out that both, the Monte Carlo framework as well as the Markov chain used for sampling, do not specifically rely on the fact that $\mathcal{F}$ is the family of frequent sets of a transactional dataset. Instead the top-level analysis only uses the property that $\mathcal{F}$ is closed "downward", i.e., under taking subsets. This is also true for most of the speedups presented in Section 7. Thus, the major part of Algorithm 1 is applicable to other pattern mining tasks that use an anti-monotone interestingness predicate. On the other hand this is a strictly more general problem as there are anti-monotone set families (independence systems) that cannot concisely be represented as the family of frequent sets of a transactional dataset (Sloan, Takata and Turán, 1998). Consequently, for the specific task of counting frequent sets a major route for potential improvement is to investigate whether there are partitionings as well as sampling methods that exploit more structural properties of the problem.

Beyond Frequent Sets On a more global scale, an important next step is to investigate whether the randomized counting approach can be extended to other pattern classes like closed sets or graph mining. The first one induces set systems that are not closed downward, and the second is not even representable as sets at all. Thus, a straightforward application of the techniques discussed in this paper is impossible. This extension, however, promises to be both, challenging and beneficial, as the cost of exhaustive mining generally increases with the pattern complexity.

Acknowledgments

The authors wish to thank Thomas Gärtner for the helpful insights he provided them into Markov chains. This research was partially supported by the European Commission under the project *iWebCare* (IST-2005-028055).

References

Bayardo, R., Goethals, B. and Zaki, M. J., eds (2004), *Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, 2004*, Vol. 126 of *CEUR Workshop Proceedings*, CEUR-WS.org.

Bernstein, A., Provost, F. and Hill, S. (2005), 'Toward intelligent assistance for a data mining process: An ontology-based approach for cost-sensitive classification', *IEEE Trans. on Knowl. and Data Eng.* **17**(4), 503–518.

Blanchard, J., Guillet, F. and Briand, H. (2007), 'Interactive visual exploration of association rules with rule-focusing methodology', *Knowledge and Information Systems* **13**(1), 43–75.

Bodon, F. (2003), A fast apriori implementation, *in* B. Goethals and M. J. Zaki, eds, 'Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'03)', Vol. 90 of *CEUR Workshop Proceedings*, Melbourne, Florida, USA.

Boley, M. (2007), On approximating minimum infrequent and maximum frequent sets, *in* 'Discovery Science', pp. 68–77.

Boley, M., Horváth, T. and Wrobel, S. (2009), Efficient discovery of interesting patterns based on strong closedness, *in* 'Proceedings of the SIAM International Conference for Data Mining (SDM)'.

Geerts, F., Goethals, B. and Bussche, J. V. D. (2005), 'Tight upper bounds on the number of candidate patterns', *ACM Trans. Database Syst.* **30**(2), 333–363.

Grahne, G. and Zhu, J. (2003), Efficiently using prefix-trees in mining frequent itemsets, *in* 'FIMI'03 Workshop on Frequent Itemset Mining Implementations'.

Gunopulos, D., Khardon, R., Mannila, H., Saluja, S., Toivonen, H. and Sharma, R. S. (2003), 'Discovering all most specific sentences', *ACM Trans. Database Syst.* **28**(2), 140–174.

Hämäläinen, W. and Nykänen, M. (2008), Efficient discovery of statistically significant association rules, *in* 'ICDM'.

Han, J. and Kamber, M. (2000), *Data Mining: Concepts and Techniques*, Morgan Kaufmann.

Jerrum, M. R., Valiant, L. G. and Vazirani, V. V. (1986), 'Random generation of combinatorial structures from a uniform distribution', *Theor. Comput. Sci.* **43**(2-3), 169–188.

Jerrum, M. and Sinclair, A. (1997), The markov chain monte carlo method: an approach to approximate counting and integration, *in* 'Approximation algorithms for NP-hard problems', PWS Publishing Co., Boston, MA, USA, pp. 482–520.

Jin, R., McCallen, S., Breitbart, Y., Fuhry, D. and Wang, D. (2009), Estimating the number of frequent itemsets in a large database, *in* 'Proceedings of 12th International Conference on Extending Database Technology (EDBT)'.

Karp, R. M., Luby, M. and Madras, N. (1989), 'Monte-carlo approximation algorithms for enumeration problems', *J. Algorithms* **10**(3), 429–448.

Khot, S. (2004), Ruling out ptas for graph min-bisection, densest subgraph and bipartite clique, *in* 'Foundations of Computer Science', IEEE Computer Society, Washington, DC, USA, pp. 136–145.

Li, W. and Mozes, A. (2004), Computing frequent itemsets inside oracle 10g, *in* 'VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases', VLDB Endowment, pp. 1253–1256.

Morik, K. and Scholz, M. (2002), The miningmart approach, *in* 'GI Jahrestagung', pp. 811–818.

Pei, J. and Han, J. (2000), Can we push more constraints into frequent pattern mining?, *in* 'KDD', pp. 350–354.

Randall, D. (2006), 'Rapidly mixing markov chains with applications in computer science and physics', *Computing in Science and Engineering* **8**(2), 30–41.

Scheffer, T. and Wrobel, S. (2002), 'Finding the most interesting patterns in a database quickly by using sequential sampling', *Journal of Machine Learning Research* **3**, 833–862.

Sloan, R. H., Takata, K. and Turán, G. (1998), 'On frequent sets of boolean matrices', *Annals of Mathematics and Artificial Intelligence* **24**(1-4), 193–209.

Tatti, N. (2008), 'Maximum entropy based significance of itemsets', *Knowledge and Information Systems* **17**(1), 57–77.

Utley, C. (2005), Introduction to sql server 2005 data mining, Technical report.

Valiant, L. G. (1979), 'The complexity of computing the permanent', *Theor. Comput. Sci.* **8**, 189–201.

Wang, J., Han, J., Lu, Y. and Tzvetkov, P. (2005), 'TFP: An efficient algorithm for mining top-k frequent closed itemsets', *IEEE Trans. Knowl. Data Eng.* **17**(5), 652–664.

Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A. F. M., Liu, B., Yu, P. S., Zhou, Z.-H., Steinbach, M., Hand, D. J. and Steinberg, D. (2008), 'Top 10 algorithms in data mining', *Knowledge and Information Systems* **14**(1), 1–37.

Yoshizawa, T., Pramudiono, I. and Kitsuregawa, M. (2000), 'Sql based association rule mining using commercial rdbms (ibm db2 udb eee)', *Data Warehousing and Knowledge Discovery* pp. 301–306.

Zhang, S., Wu, X., Zhang, C. and Lu, J. (2008), 'Computing the minimum-support for mining frequent patterns', *Knowledge and Information Systems* **15**(2), 233–257.

Zuckerman, D. (1996), 'On unapproximable versions of np-complete problems', *SIAM J. Comput.* **25**(6), 1293–1304.

Author Biographies

Mario Boley is currently a PhD student at the Fraunhofer Institute for Intelligent Analysis and Information Systems in Sankt Augustin. Particularly he is working in the group for Computational Aspects of Mining and Learning within the department for Knowledge Discovery. He received a Diplom with distinction in Computer Science (with minor subject Mathematics) from the University of Bonn in 2007. His reasearch interests are the computational aspects of intelligent and efficient data analysis systems.

Henrik Grosskreutz is currently working as a research fellow in the Knowledge Discovery Group at Fraunhofer IAIS in Sankt Augustin. He studied Computer Science at the Universities of Würburg, Caen and Bonn, where he received a Diplom in Computer Science in 1998. Subsequently he joined the knowledge-based systems group at the RWTH Aachen University, where he he finished his dissertation in 2002, advised by Prof. Gerhard Lakemeyer. Thereafter he worked several years as software developer and project manager in knowledge and innovation management software projects. His interests lie in the areas of agile software development, artificial intelligence, machine learning and knowledge discovery.

Correspondence and offprint requests to: Mario Boley, Fraunhofer IAIS, Schloss Birlinghoven, 53754 Sankt Augustin, Germany. Email: mario.boley@iais.fraunhofer.de