Noname manuscript No.
(will be inserted by the editor)

Anytime Algorithm for Frequent Pattern Outlier Detection*

Arnaud Giacometti, Arnaud Soulet

Received: date / Accepted: date

Abstract Outlier detection consists in detecting anomalous observations from data. During the past decade, outlier detection methods were proposed using the concept of frequent patterns. Basically such methods require to mine all frequent patterns for computing the outlier factor of each transaction. This approach remains too expensive despite recent progress in pattern mining field to provide results within a short response time of only a few seconds. In this paper, we provide the first *anytime* method for calculating the frequent pattern outlier factor (FPOF). This method which can be interrupted at anytime by the end-user accurately approximates FPOF by mining a sample of patterns. It also computes the maximum error on the estimated FPOF for helping the user to stop the process at the right time. Experiments show the interest of this method for very large datasets where exhaustive mining fails to provide good approximate solutions. The accuracy of our anytime approximate method outperforms the baseline approach for a same budget in number of patterns.

1 Introduction

Outlier detection consists in detecting anomalous observations from data [17]. The outlier detection problem has important applications, such as detection of credit card fraud or network intrusions. During the past decade, outlier detection methods were proposed for categorical data [2,7,9,18,20,27,28]. The general principle is to build a model that reflects the majority of the dataset and to judge as outlier all data observations that deviate from this model. Some of these approaches use the concept of frequent patterns [18,20, 27] for building the model. Their key idea is to consider the number of frequent patterns supported by each data observation. A data observation is unlikely to be an outlier if it supports many frequent patterns since frequent patterns correspond to the "common features" of the dataset. Frequent pattern outlier detection methods first extract all frequent itemsets from the data and then assign an outlier score to each data observation based on the frequent itemsets it contains. These outlier detection methods follow the schema of pattern-based two-step methods.

Pattern-based two-step methods [19] aim at exhaustively mining all patterns (first step) in order to build models (second step) like pattern sets (e.g., classifier [21] or clustering [10]) or pattern-based measures (e.g., FPOF [18] or CPCQ index [22]). The completeness of pattern mining is often considered as a crucial advantage for constructing accurate models or measures. However, it also leads to three important issues that hinder the user interaction with the system:

1. **Threshold issue:** The completeness of the first step requires to adjust thresholds which is recognized as being very difficult. Typically, if the minimal support threshold is too low, the extraction becomes unfeasible. If it is too high, some essential patterns are missed.
2. **Accuracy issue:** Completeness leads to huge pattern volumes without guaranteeing not missing important patterns. For a smaller budget (in time or number of patterns), we claim that non-exhaustive methods can produce collections of patterns better adapted to the task of the second step. Interestingly,

Université François Rabelais Tours, LI EA 6300
3 place Jean Jaurès, F-41029 Blois, France
firstname.lastname@univ-tours.fr

a non-exhaustive method can even guarantee a certain quality on the second step.

3. **Runtime issue:** The exhaustive mining of all patterns requires to explore the search space in a certain fashion that extracts either very general patterns first (breadth-first search) or very similar patterns to each other (depth-first search). For having patterns regularly covering the search space, it is necessary to wait for the end of the extraction step before starting the model construction. As this first step is very time consuming, it prevents the user to have an immediate answer.

In order to cope with these issues, we propose an approach for pattern-based outlier detection that do not rely on exhaustive mining. This paper extends the previous version [14] by taking into account an anytime constraint i.e., a terminating condition based on a budget in time or patterns rather than a maximum error. However, the maximum error is still communicated to the user to help stop the process at the right time.

This paper revisits the calculation of the Frequent Pattern Outlier Factor (FPOF) with an anytime constraint by benefiting from recent pattern sampling techniques. Our goal is not to propose a new outlier detection factor. Rather, we want to better approximate, and at anytime, the FPOF. Although this factor has several limitations, it remains popular and our approach can be applied to other. Furthermore, the main limitation of FPOF is clearly its computational cost that raises the three above issues. To tackle this problem, our proposal is to propose an *anytime* algorithm, i.e. algorithm that can be interrupted at any point of time to supply an answer whose quality increases with computational time [4]. To this purpose, the key idea of our proposal is to mine a pattern sample instead of mining the exhaustive collection of frequent patterns. We then reformulate the FPOF by considering the current sample of patterns. Using Bennett's inequality, this method guarantees a maximum error for a given confidence at anytime. Experimental study shows the efficiency of our sampling-based method on benchmarks coming from UCI Machine Learning repository and FIMI repository by considering evaluation criteria of anytime algorithms [32]:

- **Accuracy:** The result of our sampling-based anytime algorithm converges to the exact FPOF when time tends to infinity. In particular, the Kendall's tau which evaluates the similarity between the rankings induced by the approximate and exact FPOF increases rapidly and smoothly with pattern budget.
- **Certainty:** The error estimated stemming from Bennett's inequality is relatively close to the true error. The end-user therefore has an objective interestingness measure in order to help stop the algorithm at the right time.
- **Stability:** Even if the proposed algorithm is non-deterministic, the variability (evaluating by the standard deviations of accuracy measures) decreases with sample size and time. This means that multiple executions give approximately the same answer.

The outline of this paper is as follows. Section 2 reviews some related work about outlier detection, pattern sampling and anytime algorithms. Section 3 introduces the basic definitions about the FPOF. Section 4 motivates and states the problem of its anytime approximate calculation. We introduce our anytime approximate method based on sampling in Section 5. Section 6 provides experimental results. We conclude in Section 7.

2 Related Work

2.1 Pattern-based outlier detection

The outlier detection methods are primarily based on the construction of a model that describes the majority of data observations. A new data observation is then considered abnormal when it strongly deviates from this model. In this paper, we mainly focus on the outlier detection methods dedicated to categorical data. A broader view of outlier detection is provided by surveys including [17]. Different frameworks are dedicated to categorical data for the construction of the model including the Minimum Description Length framework [2], the probability framework (using Hidden Markov Models (HMM) [7], joint probabilities [9] or a random walk on attributes [28]) and the pattern-based framework [18,20,27]. Pattern-based methods benefit from the progress of pattern mining made over the past two decades. The key idea is that as the frequent patterns reflect the distribution of the dataset, they form a representative model of the dataset. Such methods remain efficient for high-dimensional spaces unlike other methods dedicated to categorical data.

The first pattern-based approach [18] introduced the frequent pattern outlier factor that exploits the complete collection of frequent itemsets (while [27] uses an opposite approach by considering non-frequent itemsets). More recently, [20] replaces the collection of frequent itemsets by the condensed representation of Non-Derivable Itemsets (NDI) which is more compact and less expensive to mine. We would go further by showing that the frequent pattern outlier factor proposed

in [18] can be approximated efficiently by extracting a small sample of patterns.

This paper benefits from FPOF which remains a popular outlier detection factor despite its known limits. Unlike other methods, it does not exploit the data structure which is often used to improve the detection of abnormal data: an organization as attribute-value [2,9,28] and in a most original way, sequentiality [7]. Moreover, recent experiments [28] have shown that the FPOF is not well-suited for identifying abnormal data when data are noisy or attributes have very different distributions. Finally, the main flaw of FPOF that we already discussed in the introduction is its computational cost. In addition, it is necessary to wait until the end of the execution to know what are the outliers. Note that even, non-pattern-based outlier detection methods which are polynomial with the dataset size suffer from the same drawbacks. By offering an anytime algorithm, our proposal gives a first result in a short response time and if there is enough time, it converges to a result as good as would give the original FPOF method.

2.2 Pattern sampling

Previous methods for pattern-based outlier detection enumerates exhaustively all patterns satisfying a given selection predicate, called constraint [24] (e.g., minimal frequency). As mentioned in introduction, it is recognized that constraint-based pattern mining leads to threshold and runtime issues which are sometimes a severe bottleneck. Recently, there has been a resurgence in pattern mining for non-exhaustive methods [12] through pattern sampling [6,8]. Pattern sampling aims at accessing the pattern space $\mathcal{L}$ by an efficient sampling procedure simulating a distribution $\pi : \mathcal{L} \rightarrow [0,1]$ that is defined with respect to some interestingness measure m: $\pi(.) = m(.)/Z$ where Z is a normalizing constant (formal framework and algorithms are detailed in [6]). In this way, the user has a fast and direct access to the entire pattern language and with no parameter (except possibly the sample size). Pattern sampling has been introduced to facilitate interactive data exploration [31]. As constraint-based pattern mining, pattern sampling problem has been declined for different languages like itemsets [6] and graphs [15], and different interestingness measures including support [6,15], area [6,26], discriminative measure [15,6] or utility measure [6,25,26].

To the best of our knowledge, there are only two proposals benefiting from pattern sampling to instantly build pattern-based global models: representative set of patterns [8] and tiling [26]. In this paper, we investigate the use of pattern sampling for assigning an outlier score to each transaction (a kind of model). But we go further by refining this model over time to finally tend to the exact model. With a lower (pattern or time) budget than that of an exhaustive method, we obtain a higher quality with a bounded error.

2.3 Anytime algorithms for pattern mining

Introduced in the field of real-time system design [4, 32], *anytime* algorithms have more recently been used in the field of data mining [3,5,11,16,23], and more specifically for pattern mining [30]. Most of the time, anytime algorithms have been used to build global models (e.g., classifiers, clusterings or rankings) when the computation time required to obtain a first model is very important. One approach to build global models using anytime algorithms is to enumerate the set of all possible solutions and keep anytime the best solution, i.e. the best global model. For example, using depth-first search based algorithms, this approach has been used to build Bayesian networks [23], or to extract groups with maximum coverage from spatio-temporal data of mobile users [30]. Another approach to build global models using anytime algorithms is to compute first a rough solution and then to refine this solution over time. For example, this approach is used in [5] to build an anytime density-based clustering algorithm and in [16] to provide high quality subspace clusterings of data streams. This approach is also used in this paper to extract outliers. Indeed, using pattern sampling, our algorithm refines the FPOF of transactions over time.

To the best of our knowledge, only the works in [3] adresses the problem of outlier detection using anytime algorithms. In [3], the authors propose an anytime algorithm to determine within any period of time whether an object in a data stream is anomalous or not. The more time is available, the more reliable the predictions are. Compared to this work, in this paper, we do not propose an algorithm to detect outliers in data streams, but in very large datasets. However, we have the same property, meaning that the accuracy of our predictions (a transaction is an outlier or not) increases with time. Finally, to the best of our knowledge, only the works in [30] uses anytime algorithms for pattern mining. Nevertheless, compared to our work, this work solves a very different problem, i.e. finding groups of users with maximum coverage in the context of spatio-temporal data mining.

$\mathcal{D}$

Trans.	Items
t_1	A B
t_2	A B
t_3	A B
t_4	C

$\mathcal{D}'$

Trans.	Items
t_1	A B
t_2	A B
t_3	A B
t_4	C
t_5	**A** **B**

$\mathcal{D}''$

Trans.	Items
t_1	A B **D**
t_2	A B **D**
t_3	A B **D**
t_4	C

Table 1 Three toy datasets with slight variations

3 Frequent Pattern Based Outlier Detection

3.1 Basic definitions

Let $\mathcal{I}$ be a set of distinct literals called *items*, an itemset (or a pattern) is a subset of $\mathcal{I}$. The language of itemsets corresponds to $\mathcal{L} = 2^{\mathcal{I}}$. A transactional dataset is a multi-set of itemsets of $\mathcal{L}$. Each itemset, usually called *transaction*, is a data observation. For instance, Table 1 gives three transactional datasets with 4 or 5 transactions t_i described by until 4 items A, B, C and D.

Pattern discovery takes advantage of interestingness measures to evaluate the relevancy of a pattern. The *support* of a pattern X in the dataset $\mathcal{D}$ is the proportion of transactions covered by X [1]: $supp(X, \mathcal{D}) = |\{t \in \mathcal{D} : X \subseteq t\}|/|\mathcal{D}|$. A pattern is said to be *frequent* when its support exceeds a user-specified minimal threshold. The set of all frequent patterns for σ as minimal threshold in $\mathcal{D}$ is denoted by $\mathcal{F}_\sigma(\mathcal{D})$:

$$\mathcal{F}_\sigma(\mathcal{D}) = \{X \in \mathcal{L} : supp(X, \mathcal{D}) \geq \sigma\}$$

In the following, we manipulate pattern multisets which are collections of patterns admitting several occurrences of the same pattern. The representativeness of a pattern multiset $\mathcal{P}$, denoted by $Supp(\mathcal{P}, \mathcal{D})$, is the sum of the support of each pattern in $\mathcal{P}$:

$$Supp(\mathcal{P}, \mathcal{D}) = \sum_{X \in \mathcal{P}} supp(X, \mathcal{D})$$

The range of $Supp(\mathcal{P}, \mathcal{D})$ is $[0, |\mathcal{P}|]$. Given a cardinality, high representativeness means the multiset contains very common patterns of the dataset. For comparing the content of two pattern multisets, we use the semi-join, denoted by $\mathcal{P}_2 \triangleright \mathcal{P}_1$, that returns all the patterns of $\mathcal{P}_2$ occurring in $\mathcal{P}_1$:

$$\mathcal{P}_2 \triangleright \mathcal{P}_1 = \{X \in \mathcal{P}_2 : X \in \mathcal{P}_1\}$$

For instance, $\{A, AB, A, D\} \triangleright \{C, A, B\} = \{A, A\}$.

3.2 Frequent Pattern Outlier Factor

Intuitively, a transaction is more representative when it contains many patterns which are very frequent within the dataset. In contrast, an outlier contains only few patterns and these patterns are not very frequent. The frequent pattern outlier factor [18] formalizes this intuition:

Definition 1 (FPOF) The frequent pattern outlier factor of a transaction t in $\mathcal{D}$ is defined as follows:

$$fpof(t, \mathcal{D}) = \frac{Supp(2^t, \mathcal{D})}{\max_{u \in \mathcal{D}} Supp(2^u, \mathcal{D})}$$

The range of *fpof* is $[0, 1]$ where 1 means that the transaction is the most representative transaction of the dataset while a value near 0 means that the transaction is an outlier. Other normalizations (denominator) are possible like $Supp(\mathcal{L}, \mathcal{D})$ or $\sum_{t \in \mathcal{D}} Supp(2^t, \mathcal{D})$. Whatever the normalization method, two transactions remain ordered in the same way (so it does not affect the Kendall's tau that we use to evaluate our method). Under a certain Markov model, the score $fpof(t, \mathcal{D})$ is also the proportion of time that an analyst would dedicate to study the transaction t considering the collection of frequent itemsets [13].

In the first dataset provided by Table 1, t_1 is covered by $\emptyset$ ($supp(\emptyset, \mathcal{D}) = 1$) and, A, B and AB whose support equals to 0.75 ($Supp(\{\emptyset, A, B, AB\}, \mathcal{D}) = 3.25$) while t_4 is only covered by $\emptyset$ and C ($Supp(\{\emptyset, C\}, \mathcal{D}) = 1.25$). Consequently, $fpof(t_1, \mathcal{D}_1) = 3.25/3.25$ and $fpof(t_4, \mathcal{D}_1) = 1.25/3.25$. In this example, t_4 appears to be an outlier. It is easy to see that increasing the frequency of the patterns covering the first transactions (e.g., dataset $\mathcal{D}'$) decreases the FPOF of t_4. Similarly, increasing the number of patterns covering the first transactions also decreases the FPOF factor of t_4 (e.g., dataset $\mathcal{D}''$).

4 Problem Formulation

4.1 Exact FPOF computation problem

Given a dataset $\mathcal{D}$, the outlier detection problem consists in computing the FPOF for each transaction $t \in \mathcal{D}$. In practice, this exact calculation of the frequent pattern outlier factor was performed by mining all patterns appearing at least once in the dataset (i.e., with $\sigma = 1/|\mathcal{D}|$) [18]. Of course, this expensive task is not possible for very large datasets. Recently, it has been demonstrated that the FPOF can be reformulated in order to calculate the exact FPOF in polynomial time [14].

$\mathcal{D}$	Exhaustive Time (s)	Non-enumerative Time (s)
chess	439.5	**1.1**
connect	748.5	**577.7**
mushroom	**0.4**	5.9
pumsb	time out	**1,970.5**
retail	**8.7**	5,969.9
sick	0.8	**0.5**

Table 2 Time comparison of two exact methods for calculating FPOF

To calculate the FPOF of a transaction t, Definition 1 formulates the problem in terms of frequent patterns appearing in t. The idea is to reformulate this factor by considering what each transaction u brings to the transaction t. For instance, in dataset $\mathcal{D}$, the FPOF of the first transaction relies on $Supp(\{\emptyset, A, B, AB\}, \mathcal{D})$ which is equal to $|\{\emptyset, A, B, AB,\ \emptyset, A, B, AB,\ \emptyset, A, B, AB,\ \emptyset\}|/4$. Each subset $\{\emptyset, A, B, AB\}$ or $\{\emptyset\}$ results from the intersection of patterns covering t_1 with those covering another transaction $u \in \mathcal{D}$. Thereby, $Supp(\{\emptyset, A, B, AB\}, \mathcal{D}) = |\{\bigcup_{u\in\mathcal{D}} 2^{t_1} \cap 2^u\}|/|\mathcal{D}| = |\{\bigcup_{u\in\mathcal{D}} 2^{t_1\cap u}\}|/|\mathcal{D}|$. Given a dataset $\mathcal{D}$, this observation leads to reformulate the frequent pattern outlier factor as follows for all transaction $t \in \mathcal{D}$:

$$fpof(t, \mathcal{D}) = \frac{\sum_{u\in\mathcal{D}} 2^{|t\cap u|}}{\max_{v\in\mathcal{D}} \sum_{u\in\mathcal{D}} 2^{|v\cap u|}}$$

From a conceptual point of view, it is interesting to note that ultimately, the FPOF of a transaction is just the sum of its similarity with each of transactions (where similarity between t and u is $2^{|t\cap u|}$). This measure is therefore very close to traditional methods relying on pair-wise distance among data observations.

Table 2 reports the running time required for calculating the exact FPOF using the classical *exhaustive* method [18] and the *non-enumerative* method [14] (respectively the 2nd and the 3rd column) based on the experimental setting described in Section 6. Note that the exact exhaustive method (as baseline) benefits from LCM which is one of the most recognized frequent itemset mining algorithm. The non-enumerative method is effective and rivals the exact exhaustive one. Its main advantage is to calculate the exact FPOF with datasets where the exact exhaustive method fails (e.g., pumsb where the execution was aborted after 5h).

However, even with a polynomial method, Table 2 shows that the exact calculation remains time-consuming. It is clear that the exact FPOF calculation cannot be guaranteed in a short response time. Thus, it makes sense to propose approximate algorithms for the FPOF computation.

4.2 Approximate FPOF computation problem

Let us focusing on a classical approach used in literature to approximate the FPOF. Instead of using the complete collection of patterns, FPOF is usually approximated with a collection of frequent patterns i.e., with a higher minimal support threshold:

Definition 2 (σ-Exhaustive FPOF) Given a minimal support threshold σ, the σ-exhaustive FPOF of a transaction t in $\mathcal{D}$ is defined as follows:

$$fpof_\sigma(t, \mathcal{D}) = \frac{Supp(\mathcal{F}_\sigma(\mathcal{D}) \triangleright 2^t, \mathcal{D})}{\max_{u\in\mathcal{D}} Supp(\mathcal{F}_\sigma(\mathcal{D}) \triangleright 2^u, \mathcal{D})}$$

The approximation becomes accurate with very low minimal support thresholds. Figure 1 (top) plots the Kendall's tau of $fpof_\sigma$ in comparison with *fpof* for some benchmarks[1]. Unfortunately, this approximate method suffers from two issues. When the minimal support threshold becomes very low, the number of patterns (see the bottom plot of Figure 1) and the extraction time explode. Sometimes the extraction of additional patterns leads to a deterioration of the results (inaccuracy issue). Furthermore, the approximation error is not estimated. The user does not know if the returned approximate FPOF is far from the exact FPOF (uncertainty issue). With a smaller budget, we claim that it is possible to approximate more precisely FPOF while having a bound on the error.

4.3 Anytime FPOF computation problem

Figure 1 shows that the Kendall's tau varies significantly depending on the dataset for a same minimal support threshold. It means that this threshold is not easy to fix for obtaining a good compromise between efficiency and quality. It clearly hinders the user interactivity. Therefore, it seems interesting to rephrase the approximate FPOF problem by opting for an anytime perspective. In this context, the method informs the user with a feedback on the maximum error on the current approximate FPOF. Then, the user will choose the right time to stop the method.

Given a dataset $\mathcal{D}$ and a real δ, return *at anytime* k a function $\widetilde{fpof_k}$ approximating the frequent pattern outlier factor and a maximum bound ϵ_k such that:

- **$|fpof(t, \mathcal{D}) - \widetilde{fpof_k}(t, \mathcal{D})| \leq \epsilon_k$ for each transaction $t \in \mathcal{D}$, with confidence $1 - \delta$ and**

[1] It is the proportion of pairs of transactions which would be ranked similarly with the approximate FPOF and with the true FPOF (see Section 6 for a formal definition).

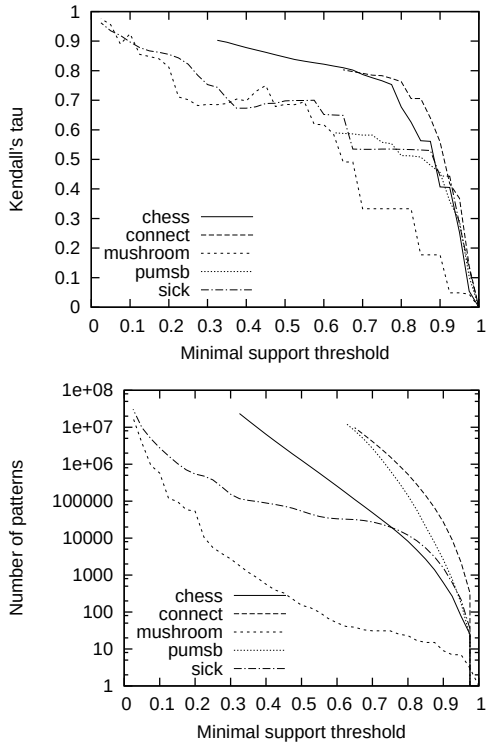

Fig. 1 Kendall's tau and number of patterns with minimal support threshold

– $\epsilon_{k+1} \leq \epsilon_k$ **where** $\lim_{k \to +\infty} \epsilon_k = 0$.

This problem aims at assigning at anytime an approximate FPOF to each transaction by guaranteeing a maximum error ϵ_k with the probability $1 - \delta$. Note that, in the following, we will express all budgets in patterns (denoted by k, here). The requirement of returning a maximum bound that monotonically decreases and converges to 0 avoids the drawbacks described in the above approach based on frequent patterns where the accuracy of the approximation may decrease (despite a higher pattern budget) and where the certainty is unknown.

5 Anytime Sampling Method

This section addresses the above problem by using pattern sampling. First, we propose a method for approximating FPOF from a pattern sample drawn according to frequency. Then we show how to estimate the error of this approximation. Finally, we detail our sampling-based anytime algorithm.

5.1 Pattern sampling for FPOF

In Section 4, we showed that the use of the most frequent patterns is insufficient to approximate accurately FPOF. The most frequent patterns do not measure the singularity of each transaction that also relies on more specific patterns (whose frequency varies from small to average). Conversely do not considering frequent patterns would also be a mistake because they contribute significantly to FPOF. A reasonable approach is to select patterns randomly with a probability proportional to their weight in the calculation of FPOF. Typically, in the dataset $\mathcal{D}$ of Table 1, the itemset AB is 3 times more important than itemset C in the calculation of FPOF due to their frequency.

In recent years pattern sampling techniques have been proposed to randomly draw patterns in proportion to their frequency [6]. Such approaches are ideal to bring us a well-adapted collection of patterns. Of course, it remains the non-trivial task of approximating FPOF starting from this collection. This is what provides the following definition:

Definition 3 (k-Sampling FPOF) Given an integer $k > 0$, a k-sampling frequent pattern outlier factor of a transaction t in $\mathcal{D}$ is defined as follows:

$$fpof_k(t, \mathcal{D}) = \frac{|\mathcal{S}_k(\mathcal{D}) \triangleright 2^t|}{\max_{u \in \mathcal{D}} |\mathcal{S}_k(\mathcal{D}) \triangleright 2^u|}$$

where $\mathcal{S}_k(\mathcal{D})$ is a sample of k patterns drawn from $\mathcal{D}$ according to support: $\mathcal{S}_k(\mathcal{D}) \sim supp(\mathcal{L}, \mathcal{D})$.

It is important to note that $|.|$ is used here instead of $Supp(., \mathcal{D})$ as done in Definition 1. As the sampling technique already takes into account the frequency when it draws patterns, it is not necessary to involve the support here. Indeed, the draw is with replacement for the correct approximation of FPOF (without this replacement the most frequent patterns would be disadvantaged). It induces that the same pattern can have multiple occurrences within the sample $\mathcal{S}_k(\mathcal{D})$.

For the same sample size k and for the same transaction t, it is possible to calculate different values of a k-sampling FPOF due to $\mathcal{S}_k(\mathcal{D})$. But, the higher the threshold k, the less the difference between values stemming from two samples is high. Furthermore, the greater the sample size k, the better the approximation:

Property 1 (Convergence) Given a dataset $\mathcal{D}$, a k-sampling FPOF converges to the FPOF for each transaction $t \in \mathcal{D}$.

Proof $\mathcal{S}_k(\mathcal{D}) \sim supp(\mathcal{L}, \mathcal{D})$ means that there exists a constant $\alpha > 0$ such that $\forall X \in \mathcal{L}$, $\lim_{k \to \infty} |\mathcal{S}_k(\mathcal{D}) \triangleright \{X\}| = \alpha supp(X, \mathcal{D})$. Then, for each

transaction t, we obtain that: $\lim_{k\to\infty} |\mathcal{S}_k(\mathcal{D}) \triangleright 2^t| = \alpha \sum_{X\in 2^t} supp(X,\mathcal{D}) = \alpha Supp(2^t,\mathcal{D})$. By injecting this result into Definition 3, we conclude that Property 1 is right. □

Beyond convergence, the interest of this approach is the speed of convergence far superior to that of the σ-exhaustive frequent pattern outlier factor as shown in the experimental study (see Section 6). This speed is accompanied by a good efficiency due to a reasonable complexity of pattern sampling:

Property 2 (Complexity) A k-sampling FPOF of all transactions can be calculated in time $O(k \times |\mathcal{I}| \times |\mathcal{D}|)$.

Proof Pattern sampling according to frequency is performed in time $O(|\mathcal{I}| \times |\mathcal{D}| + k(|\mathcal{I}| + \ln |\mathcal{D}|))$ [6] and the FPOF calculation for all transactions consists in finding the transactions containing each sampled pattern. Thus, it is calculated in time $O(k \times |\mathcal{I}| \times |\mathcal{D}|)$. □

Given a number of patterns k (which is the allocated pattern budget), a k-sampling FPOF is therefore effective to calculate an accurate approximation. The next section goes further by ensuring certainty of this approximation.

5.2 Bounding the error

This section shows how to provide a feedback for helping the user in his/her decision to stop the algorithm. The idea is to draw a sample and to bound the maximum error of FPOF using a statistical result known as Bennett's inequality. This maximum error is provided to the end-user given an initial confidence. If he/she judges that the quality is sufficiently good, he/she interrupts the algorithm that returns an approximate FPOF based on the current sample. Otherwise, the sampling FPOF is refined by increasing the sample size and so on.

We use Bennett's inequality to estimate the current error because it is true irrespective of the probability distribution. After k independent observations of real-valued random variable r with range $[0,1]$, Bennett's inequality ensures that, with confidence $1-\delta$, the true mean of r is at least $\overline{r} - \epsilon$ where $\overline{r}$ and $\overline{\sigma}$ are respectively the observed mean and variance of the samples and

$$\epsilon = \sqrt{\frac{2\overline{\sigma}\ln(1/\delta)}{k}} + \frac{\ln(1/\delta)}{3k}$$

In our case, the random variable is the average number of patterns within a sample $\mathcal{S}_k \sim supp(\mathcal{L},\mathcal{D})$ that cover the transaction t. It is denoted by $cov_{\mathcal{S}_k}(t)$ and defined as follows: $cov_{\mathcal{S}_k}(t) = |\mathcal{S}_k \triangleright 2^t|/k$. It is easy to see that a k-sampling FPOF factor can be rewritten using $cov_{\mathcal{S}_k}$: $fpof_k(t,\mathcal{D}) = cov_{\mathcal{S}_k}(t)/\max_{v\in\mathcal{D}} cov_{\mathcal{S}_k}(v)$. Using Bennett's inequality and the above definition enables us to bound FPOF:

Property 3 (FPOF Bounds) Given a dataset $\mathcal{D}$ and confidence $1-\delta$, the FPOF of transaction t is bounded as follows:

$$\underbrace{\max\left\{0, \frac{cov_{\mathcal{S}_k}(t)-\epsilon_t}{cov_{\mathcal{S}_k}(u)+\epsilon_u}\right\}}_{m_k(t)} \leq fpof(t,\mathcal{D}) \leq \underbrace{\min\left\{\frac{cov_{\mathcal{S}_k}(t)+\epsilon_t}{cov_{\mathcal{S}_k}(u)-\epsilon_u}, 1\right\}}_{M_k(t)}$$

where $\mathcal{S}_k \sim supp(\mathcal{L},\mathcal{D})$, $u = \arg\max_{v\in\mathcal{D}} cov_{\mathcal{S}_k}(v)$ and $\epsilon_t = \sqrt{2\overline{\sigma_t}\ln(1/\delta)/k} + \ln(1/\delta)/(3k)$ with $\overline{\sigma_t}$ which is the empirical standard deviation of $cov_{\mathcal{S}_k}(t)$.

Proof Given a confidence $1-\delta$ and a transaction $t \in \mathcal{D}$, Bennett's inequality gives that $cov_{\mathcal{S}_k}(t) - \epsilon_t \leq Supp(2^t,\mathcal{D}) \leq cov_{\mathcal{S}_k}(t) + \epsilon_t$ with $\epsilon_t = \sqrt{2\overline{\sigma_t}\ln(1/\delta)/k} + \ln(1/\delta)/(3k)$. In particular, this inequality holds for $u = \arg\max_{v\in\mathcal{D}} cov_{\mathcal{S}_k}(v)$ and then, we obtain:

$$\frac{cov_{\mathcal{S}_k}(t)-\epsilon_t}{cov_{\mathcal{S}_k}(u)+\epsilon_u} \leq \frac{Supp(2^t,\mathcal{D})}{Supp(2^u,\mathcal{D})} \leq \frac{cov_{\mathcal{S}_k}(t)+\epsilon_t}{cov_{\mathcal{S}_k}(u)-\epsilon_u}$$

As the FPOF lies within the interval $[0,1]$, we conclude that Property 3 is right. □

In other words, Property 3 allows us to approximate the exact FPOF starting from a sample of patterns randomly drawn according to frequency. Indeed, the exact FPOF is between $m_k(t)$ and $M_k(t)$ and the current k-sampling FPOF approximates it with a maximum bounded error $(M_k(t) - m_k(t))/2$. Rather than presenting to the end-user this estimated error for each transaction, we provide the average maximum error in experimental study (see Section 6). Nevertheless, we can go further by approximating the Kendall's tau that is often used in practice to estimate the quality of a ranking. The Kendall's tau compares the ranking stemming from an approximate method f with that stemming from the exact FPOF as follows:

$$\tau(f) = \frac{|\{(t,u) \in \mathcal{D}^2 : sgn(f(t)-f(u)) = sgn(fpof(t)-fpof(u)\}|}{|\mathcal{D}|^2}$$

Using the lower and upper bounds for each transaction stemming from Property 3, we can compute the pessimistic value of the Kendall's tau considering the current sample:

Property 4 (Kendall's tau Bound) Given a dataset $\mathcal{D}$ and confidence $1-\delta$, the Kendall's tau of a k-sampling FPOF, denoted by $\tau(fpof_k)$, is lower bounded as follows:

$$\frac{\{(t,t') \in \mathcal{D}^2 : m_k(t) \geq M_k(t') \vee M_k(t) \leq m_k(t')\}}{|\mathcal{D}|^2} \leq \tau(fpof_k)$$

Proof This property is a direct corollary of Property 3. For each pair of transactions t and t', we are sure that the ranking of the approximate method is correct when the lower bound of one transaction is higher than the upper bound of the other. Property 3 provides these bounds. □

Algorithm 1 Anytime FPOF Computation

Input: A dataset $\mathcal{D}$, a confidence $1-\delta$
Output: A k-sampling frequent pattern outlier factor of all transactions in $\mathcal{D}$ with an estimated error for a confidence $1-\delta$

1: $\tilde{\epsilon} \leftarrow 1$; $\mathcal{S} \leftarrow \emptyset$
2: **repeat**
3: $\quad \mathcal{S} \leftarrow \mathcal{S} \cup \{X\}$ where $X \sim supp(\mathcal{L}, \mathcal{D})$ *// add a pattern in the sample*
4: $\quad m \leftarrow \arg\max_{t \in \mathcal{D}} cov_{\mathcal{S}}(t)$ *// select the most covered transaction*
$\quad$ *// estimate the maximal error on* $cov_{\mathcal{S}}$
5: $\quad \epsilon_t \leftarrow \sqrt{2\overline{\sigma_t} \ln(1/\delta)/|\mathcal{S}|} + \ln(1/\delta)/(3|\mathcal{S}|)$ for each $t \in \mathcal{D}$
$\quad$ *// estimate the maximal error on FPOF*
6: $\quad \tilde{\epsilon} \leftarrow \max_{t \in \mathcal{D}}\{\min\{1; (cov_{\mathcal{S}}(t) + \epsilon_t)/(cov_{\mathcal{S}}(m) - e_m)\} - cov_{\mathcal{S}}(t)/cov_{\mathcal{S}}(m)\}$
7: $\quad \tilde{\epsilon} \leftarrow \max_{t \in \mathcal{D}}\{cov_{\mathcal{S}}(t)/cov_{\mathcal{S}}(m) - \max\{0; (cov_{\mathcal{S}_k}(t) - \epsilon_t)/(cov_{\mathcal{S}}(m) + e_m)\}; \tilde{\epsilon}\}$
8: $\quad$ Print the estimated bounds about error per transaction and Kendall's tau as feedback
9: **until** The user stops the process
10: **return** $\langle cov_{\mathcal{S}}(t)/\max_{u \in \mathcal{D}} cov_{\mathcal{S}}(u)\rangle_{t \in \mathcal{D}}$

Property 4 enables us to bound the true Kendall's tau of our approach. Unfortunately, it is not possible to estimate similar bounds about evaluation metrics that rely on the ground truth because this ground truth is obviously not known in advance by the approximate approach. For instance, it is impossible to estimate the false alarm rate or the detection rate as these measures require to know the true outliers. An outlier threshold α is used in order to define these true outliers in the experimental section (see Section 6.2).

Properties 3 and 4 provide bounds which are used in the algorithm of the next section.

5.3 Anytime algorithm

Algorithm 1 returns, at anytime, an approximate FPOF of all transactions of the dataset $\mathcal{D}$ by guaranteeing a bounded error with confidence $1-\delta$. Basically, the main loop (line 2-9) is iterated until that the user interrupts the process (line 9). Lines 4-7 calculate the maximal error $\tilde{\epsilon}$ using Property 3 and line 8 prints the current approximated bounds described in the previous section as feedback for helping the user. When the user interrupts the process, line 10 returns the k-sampling FPOF with the current sampling $\mathcal{S}$. Otherwise, one more pattern is drawn (line 3) and so on.

As desired in Section 4.3, Algorithm 1 approximates the FPOF of all transactions for a pattern budget k:

Property 5 (Correctness) Given a dataset $\mathcal{D}$ and a confidence $1-\delta$, Algorithm 1 returns for for all pattern budgets k the k-sampling FPOF $fpof_k$ that approximates the exact FPOF such that:

- $|fpof(t,\mathcal{D}) - fpof_k(t,\mathcal{D})| \leq \epsilon_k$ for each transaction $t \in \mathcal{D}$, with a confidence of $1-\delta$ and
- $\lim_{k \to +\infty} \epsilon_k = 0$.

where $\epsilon_k = \max_{t \in \mathcal{D}}(M_k(t,\mathcal{D}) - m_k(t,\mathcal{D}))/2$.

Proof This property is a direct corollary of Property 3. The proposed bounds justify the above definition of error ϵ_k and ensure that $|fpof(t,\mathcal{D}) - fpof_k(t,\mathcal{D})| \leq \epsilon_k$ for $t \in \mathcal{D}$ with $1-\delta$ as confidence. Furthermore, as the bounds are refined when the budget k increases, it gives that $\lim_{k \to +\infty} \epsilon_k = 0$. □

Next section also provides experiments showing that $\epsilon_{k+1} \leq \epsilon_k$ even if it is not possible to formally prove this result due to the empirical variance that may increase.

$\mathcal{D}$	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{I}\rvert$	Avg. number of patt. per sec.
chess	3,196	75	29.0k
connect	67,557	129	1.2k
hepatic	155	45	219.3k
german	1,000	76	78.6k
mushroom	8,124	119	17.9k
pumsb	49,096	7,117	1.7k
retail	88,162	16,470	1.5k
sick	2,800	58	29.4k

Table 3 Performance issue of pattern sampling

6 Experimental Study

The goal of this paper is not to define a new outlier detection factor, but to improve the computing of FPOF that is well established. For this reason, we do not provide new experiments showing the interest and the limits of FPOF for detecting outliers as this aspect is already detailed in literature (see related work in Section 2). Experiments exclusively focus on the study of the quality of the approximate FPOF provided by our sampling-based anytime algorithm in comparison with the exact FPOF used as reference. The exact FPOF is computed by the polynomial method described in Section 4.1.

Experiments are conducted on datasets coming from the UCI Machine Learning repository[2] and the FIMI repository[3]. Table 3 gives the main features of datasets in the first 3 columns. All experiments are performed on a 2.5 GHz Xeon processor with the Linux operating system and 2 GB of RAM memory. Algorithms are implemented in C++ language.

In the following, we only consider budgets in patterns but considering the average number of sampled

[2] archive.ics.uci.edu/ml
[3] fimi.ua.ac.be

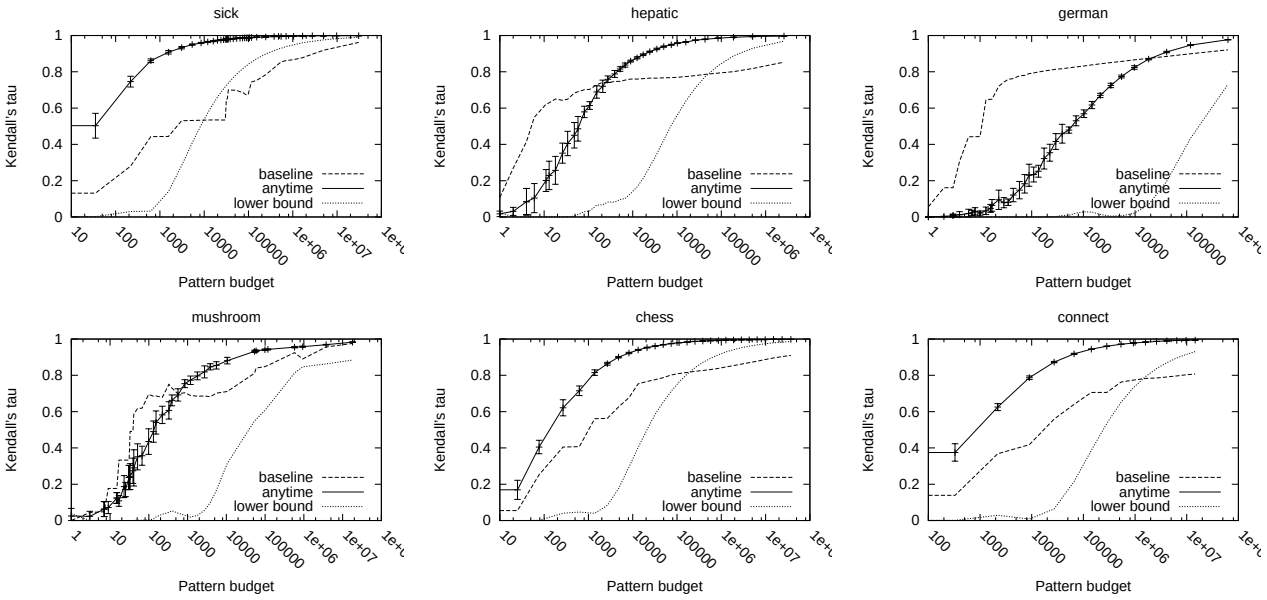

Fig. 2 Kendall's tau per transaction with a time budget

patterns per second (see the last column of Table 3), it is easy to convert pattern budgets into time budgets due to the linearity of pattern sampling.

6.1 Anytime approximation vs the state-of-the-art approximation

This section compares the abilities of the two below methods to approximate the exact FPOF according to a given pattern budget k:

- **Baseline:** This method relies on σ-Exhaustive FPOF (see Definition 2) where σ is defined for considering the set of top-k frequent patterns.
- **Sampling-based method:** This method draws k patterns according to frequency and then, approximates the exact FPOF based on the formula of Definition 3.

More precisely, we confront these two methods considering 3 quality criteria: 1) *Accuracy:* regularity and rapidity of the convergence; 2) *Certainty:* precision of the lower bounds and 3) *Stability:* reproducibility of the approximate FPOF for the same sample size. For this purpose, in Figure 2, we recall that the Kendall's tau for comparing the ranking stemming from an approximate method f with that stemming from the exact FPOF (calculated with an exact method):

$$\tau(f) = \frac{|\{(t,u) \in \mathcal{D}^2 : sgn(f(t) - f(u)) = sgn(fpof(t) - fpof(u)\}|}{|\mathcal{D}|^2}$$

Figure 2 reports the Kendall's tau of the sampling-based method (in plain line, *anytime*) and the baseline (in dashed line, *baseline*) according to a given pattern budget. We also report the lower bound of the Kendall's tau computed using Property 4 (in dotted line, *lower bound*). For the sampling-based method which is not deterministic, each reported evaluation measure is the arithmetic mean of 10 repeated measurements with its confidence interval.

In the same way, we also compute the average error per transaction between the approximate FPOF f and the exact FPOF:

$$\varepsilon(f, \mathcal{D}) = \frac{\sum_{t \in \mathcal{D}} |f(t, \mathcal{D}) - fpof(t, \mathcal{D})|}{|\mathcal{D}|}$$

Figure 3 reports this average error per transaction for the sampling-based method (in plain line, *anytime*) and the baseline (in dashed line, *baseline*) according to a given pattern budget. In Figure 3, we also report the upper bound of the average maximum error per transaction (in dotted line, *upper bound*) that is computed online using Property 3.

Accuracy To assess the speed of the convergence, we consider the increase of the Kendall's tau and the decrease of the true error. As expected, the two approximate methods converge to the exact FPOF when the pattern budget increases but, the convergence of the sampling-based anytime method is smoother and faster. Indeed, while the FPOF error of the baseline may increase by considering more patterns (see `german` or `mushroom` in Figure 3, for instance), the higher the pattern budget k, the better the approximation of the sampling-based method.

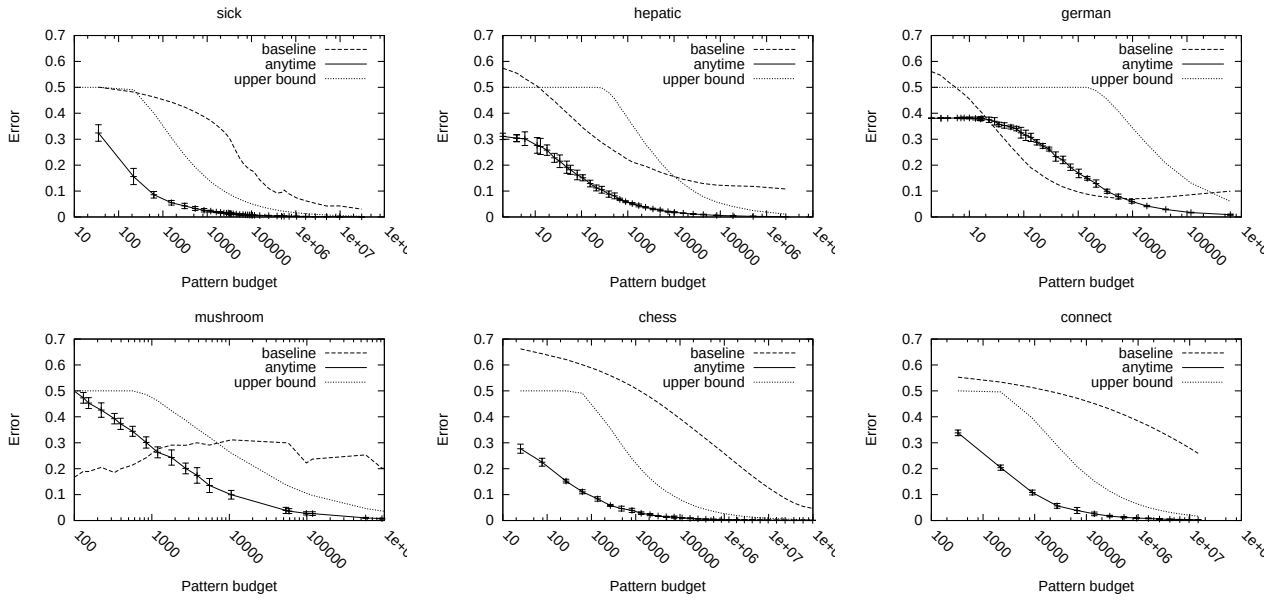

Fig. 3 Average FPOF error per transaction with a time budget

In certain datasets, when the pattern budget is small, the baseline is more effective considering the Kendall's tau especially (e.g., `hepatic` or `german`). As it considers the most frequent patterns first (in particular, items), it tends to cover more rapidly the entire dataset. It would be appropriate to propose a hybrid method where items are considered before using sampling.

Certainty Only the sampling-based method provides guarantees on the approximate FPOF computed at anytime for helping the end-user to interrupt the algorithm and to analyze the result. In Figure 3, we observe that the lower bound of the Kendall's tau is quite pessimistic (i.e., it is always much lower than the true Kendall's tau). Similarly, the true average error per transaction of the approximate method is lower than the estimated one (see Figure 3). This difference results from the Bennett's inequality that makes no assumption about the distribution. It is also interesting to note that for 2 datasets (i.e., for `chess`and `connect`), the average error per transaction of the baseline is always above the estimated error. It means that the use of the most frequent itemsets is a worse strategy than a random uniform sampling. Conversely, the sampling strategy based on frequency has higher results and in addition, this method offers some guarantees on the certainty of the approximation.

Stability To measure the stability of the sampling-based anytime method, we consider the confidence intervals of the Kendall's tau and the average error per transaction in Figures 2 and 3. Of course, the smaller the confidence interval, the better the result. Although the sampling-based method is not deterministic, the obtained results are really stable. For certain datasets (e.g., `german` or `mushroom`), the instability increases in a first phase and then gradually dwindles in a second phase. The first phase is the progressive coverage of all transactions by at least one pattern that brings instability (the approximate FPOF goes from 0 (no approximation) to 1 (first rough approximation). In the second phase, the new drawn patterns refines preliminary approximations.

	Predicted outliers $fpof_k(t,\mathcal{D}) \leq \beta$	Predicted normal $fpof_k(t,\mathcal{D}) > \beta$
Outliers $fpof(t,\mathcal{D}) \leq \alpha$	True Positive (TP)	False Negative (FN)
Normal $fpof(t,\mathcal{D}) > \alpha$	False Positive (FP)	False Negative (TN)

Table 4 Confusion matrix of 4 possible outcomes of a prediction

6.2 ROC analysis of anytime approximation

In the previous section, all transactions are considered equivalently while in practice, the final goal is to detect outliers (transactions having the lowest approximate FPOF). We now evaluate the accuracy of the approximation with a fixed pattern budget when the FPOF threshold varies between 0 and 1. Outlier detection algorithms are often evaluated using ROC analysis

[29]. In this context, given a threshold α, we consider that all transactions having an exact FPOF (computed with an exact method) below α are the outliers of the dataset (while other are normal transactions). At the same time, we predict that a transaction is an outlier when its approximate FPOF (computed with our sampling-based anytime method) is below β. Table 4 describes the four possible outcomes between ground truth and prediction. Thereby, we define the False Positive Rate (denoted by FPR) and the True Positive Rate (denoted by TPR) as follows:

$$FPR = \frac{FP}{FP + TN} \qquad TPR = \frac{TP}{TP + FN}$$

The FPR and TPR are also referred respectively as the false alarm rate and the detection rate.

For a budget of $10k$ patterns, Figure 4 reports the receiver operating characteristic (ROC) curves of the sampling-based method by varying the minimal FPOF threshold β for different ground truths $\alpha \in \{0.1, 0.2, 0.3\}$. Note that there is no outlier for $\alpha = 0.1$ in `german`.

Whatever the choice of the threshold α that determines the true outliers, the sampling-based approximation works well overall. The method tends to quickly isolate outliers (i.e., the detection rate increases very quickly when the false alarm rate is low). We even see it isolates even better the outliers when they are very few (i.e., with the lowest value of α, here 0.1).

7 Conclusion and Discussion

We revisited the FPOF calculation with an anytime constraint by benefiting from the recent advances in pattern sampling. Our approximate method using a sampling technique outperforms exhaustive method based on the most frequent patterns. It also provides additional guarantees on the result with a maximum bound on the error using the Bennett's inequality. The experiments have shown the interest of this approach in terms of accuracy (fast and smooth convergence to the exact FPOF), certainty (reasonable estimated error) and stability (good reproducibility of approximations) compared to the usual exhaustive approach where the most frequent patterns are mined.

Despite the challenge of anytime constraint, our proposal therefore combines the proven power of pattern-based methods by adding a guarantee on the quality of results thanks to sampling techniques. Of course, there is still room for improvement in particular the approach could take into account the frequent items to have a more reliable approximation at the very beginning. But, as FPOF has disadvantages, it would be interesting to apply this approach with other outlier detection methods dedicated to categorical data. For pattern-based methods, a similar design based on sampling according to frequency can be exploited. For other methods, it is really less natural to determine which space should be sampled for achieving an approximation. However, we also think our sampling-based anytime approach can be generalized to other measures involving patterns (e.g., CPCQ index [22]) or pattern-based models (e.g., CBA [21]). We would also like to adapt this approach to integrate the user feedback. In the case of FPOF, it consists in showing the transactions consider as the most probable outliers to the user at the very beginning of the process. By confirming or not that the shown transactions are outliers, the sampling process should focus its effort on other less known transactions.

Acknowledgements. This work has been partially supported by the Prefute project, PEPS 2016, CNRS.

References

1. R. Agrawal, R. Srikant, et al. Fast algorithms for mining association rules. In *International conference on Very Large Data Bases*, volume 1215, pages 487–499, 1994.
2. L. Akoglu, H. Tong, J. Vreeken, and C. Faloutsos. Fast and reliable anomaly detection in categorical data. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 415–424. ACM, 2012.
3. I. Assent, P. Kranen, C. Baldauf, and T. Seidl. Anyout: Anytime outlier detection on streaming data. In *Proceedings of the 17th International Conference on Database Systems for Advanced Applications - Volume Part I*, DASFAA'12, pages 228–242, Berlin, Heidelberg, 2012. Springer-Verlag.
4. M. Boddy and T. L. Dean. Deliberation scheduling for problem solving in time-constrained environments. *Artif. Intell.*, 67(2):245–285, June 1994.
5. C. Böhm, J. Feng, X. He, and S. T. Mai. Efficient anytime density-based clustering. In *Proceedings of the 13th SIAM International Conference on Data Mining, May 2-4, 2013. Austin, Texas, USA.*, pages 112–120. SIAM, 2013.
6. M. Boley, C. Lucchese, D. Paurat, and T. Gärtner. Direct local pattern sampling by efficient two-step random procedures. In *ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 582–590, 2011.
7. L. Cao, Y. Ou, P. S. Yu, and G. Wei. Detecting abnormal coupled sequences and sequence changes in group-based manipulative trading behaviors. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 85–94. ACM, 2010.
8. V. Chaoji, M. A. Hasan, S. Salem, J. Besson, and M. J. Zaki. ORIGAMI: A novel and effective approach for mining representative orthogonal graph patterns. *Statistical Analysis and Data Mining*, 1(2):67–84, 2008.
9. K. Das and J. Schneider. Detecting anomalous records in categorical datasets. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 220–229. ACM, 2007.
10. N. Durand and B. Crémilleux. Ecclat: a new approach of clusters discovery in categorical data. In *Research and Development in Intelligent Systems XIX*, pages 177–190. Springer, 2003.

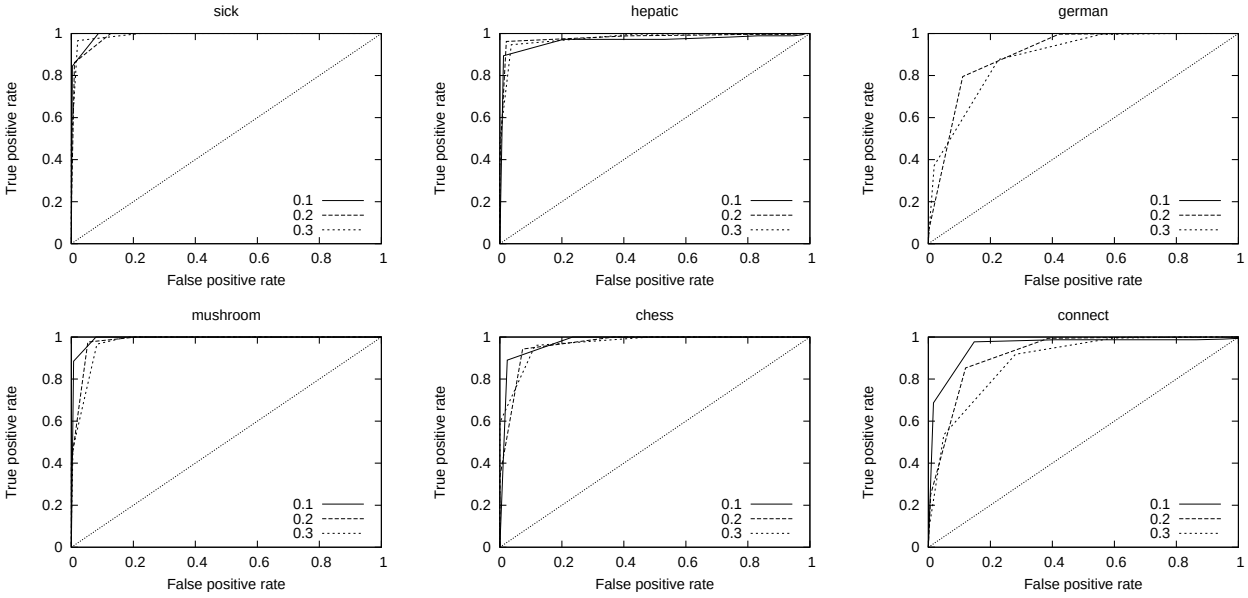

Fig. 4 ROC analysis of sampling-based anytime method according to $\alpha \in \{0.1, 0.2, 0.3\}$

11. S. Esmeir and S. Markovitch. Anytime learning of anycost classifiers. *Machine Learning*, 82(3):445–473, 2011.
12. A. Giacometti, D. H. Li, P. Marcel, and A. Soulet. 20 years of pattern mining: a bibliometric survey. *ACM SIGKDD Explorations Newsletter*, 15(1):41–50, 2014.
13. A. Giacometti, D. H. Li, and A. Soulet. Balancing the analysis of frequent patterns. In *Advances in Knowledge Discovery and Data Mining*, pages 53–64. Springer, 2014.
14. A. Giacometti and A. Soulet. Frequent pattern outlier detection without exhaustive mining. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 196–207. Springer, 2016.
15. M. A. Hasan and M. J. Zaki. Output space sampling for graph patterns. *PVLDB*, 2(1):730–741, 2009.
16. M. Hassani, P. Kranen, R. Saini, and T. Seidl. Subspace anytime stream clustering. In *Proceedings of the 26th International Conference on Scientific and Statistical Database Management*, SSDBM '14, pages 37:1–37:4, New York, NY, USA, 2014. ACM.
17. D. M. Hawkins. *Identification of outliers*, volume 11. Springer, 1980.
18. Z. He, X. Xu, Z. J. Huang, and S. Deng. FP-outlier: Frequent pattern based outlier detection. *Computer Science and Information Systems*, 2(1):103–118, 2005.
19. A. Knobbe, B. Crémilleux, J. Fürnkranz, and M. Scholz. From local patterns to global models: The lego approach to data mining. In *From local patterns to global models: proceedings of the ECML PKDD 2008 Workshop*, pages 1–16, 2008.
20. A. Koufakou, J. Secretan, and M. Georgiopoulos. Non-derivable itemsets for fast outlier detection in large high-dimensional categorical data. *Knowledge and information systems*, 29(3):697–725, 2011.
21. B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *International conference on Knowledge Discovery and Data mining*, 1998.
22. Q. Liu and G. Dong. CPCQ: contrast pattern based clustering quality index for categorical data. *Pattern Recognition*, 45(4):1739–1748, 2012.
23. B. Malone and C. Yuan. A depth-first branch and bound algorithm for learning optimal bayesian networks. In *Revised Selected Papers of the Third International Workshop on Graph Structures for Knowledge Representation and Reasoning - Volume 8323*, pages 111–122, New York, NY, USA, 2014. Springer-Verlag New York, Inc.
24. H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data mining and knowledge discovery*, 1(3):241–258, 1997.
25. S. Moens and M. Boley. Instant exceptional model mining using weighted controlled pattern sampling. In *IDA*, pages 203–214, 2014.
26. S. Moens, M. Boley, and B. Goethals. Providing concise database covers instantly by recursive tile sampling. In *International Conference on Discovery Science*, pages 216–227. Springer, 2014.
27. M. E. Otey, A. Ghoting, and S. Parthasarathy. Fast distributed outlier detection in mixed-attribute data sets. *Data Mining and Knowledge Discovery*, 12(2-3), 2006.
28. G. Pang, L. Cao, and L. Chen. Outlier detection in complex categorical data by modelling the feature value couplings. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, volume 2016, pages 9–15, 2016.
29. F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine learning*, 42(3):203–231, 2001.
30. S. G. Vadlamudi, P. P. Chakrabarti, and S. Sarkar. Anytime algorithms for mining groups with maximum coverage. In *Proceedings of the Tenth Australasian Data Mining Conference - Volume 134*, AusDM '12, pages 209–219, Darlinghurst, Australia, Australia, 2012. Australian Computer Society, Inc.
31. M. van Leeuwen. Interactive data exploration using pattern mining. In *Interactive Knowledge Discovery and Data Mining in Biomedical Informatics*, pages 169–182. Springer, 2014.
32. S. Zilberstein and S. Russell. Optimal composition of real-time systems. *Artificial Intelligence*, 82(1):181 – 213, 1996.