Information Sciences 339 (2016) 254–273

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Revisiting bound estimation of pattern measures: A generic framework

Lei Zhang[a], Ping Luo[b,*], Enhong Chen[c,**], Min Wang[d]

[a] Key Lab of Intelligent Computing and Signal Processing of Ministry of Education, School of Computer Science and Technology, Anhui University, Hefei 230039, China
[b] University of Chinese Academy of Sciences, Beijing 100080, China
[c] University of Science and Technology of China, Heifei 230026, China
[d] Google Research, Mountain View, CA 94043, USA

ARTICLE INFO

Article history:
Received 6 January 2015
Revised 26 November 2015
Accepted 26 December 2015
Available online 11 January 2016

Keywords:
Bound estimation
Utility
Occupancy
Constrained pattern mining

ABSTRACT

It is widely recognized that constrained pattern mining helps in the capture of a relatively large amount of semantics among different applications, and thus, increases the effectiveness of mining. One major challenge in this field is how the properties of pattern measures can be pushed deeply into the mining process to achieve improved efficiency. The usual solution to this challenge is to estimate the bound of a given pattern measure, $\mathcal{PM}$, for all the supersets of an itemset, X. However, in most previous studies, the authors estimated the bounds for their proposed pattern measures individually and a generic and unified framework that is applicable to any pattern measure has not been proposed. To this end, we revisit the problem of bound estimation and propose a general framework for it by summarizing the commonality among the estimation methods for different pattern measures. The basic idea is to maximize (or minimize) the measures by assigning any item labels to the items in the original supporting transactions. To achieve a balance between bound tightness and computational efficiency, we also propose techniques for addressing this tradeoff issue in order to improve the overall performance. As a case study, we applied this framework to two typical pattern measures: *utility* and *occupancy*. Additionally, we describe the application of our proposed techniques to other measures. The results of our extensive experimental evaluation on real and large synthetic datasets demonstrate the effectiveness of our proposed techniques.

1. Introduction

The mining of frequent patterns in large databases plays an essential role in many important data mining tasks, such as association rule-based classification and clustering. To improve the effectiveness and efficiency of mining, constrained pattern mining [4,11,14,16,21,22,28,30,36,38,41,45,49–51] is attracting increasing research interest, with the focus being on pushing the properties of the pattern measures deeply into the mining process to achieve better efficiency. To prune the search space, a usual approach is to estimate the bound on a given pattern measure $\mathcal{PM}$ for all supersets of an itemset X. If

* Corresponding author. Tel.: 010 62600537.
** Corresponding author.
E-mail addresses: zl@ahu.edu.cn (L. Zhang), luop@ict.ac.cn (P. Luo), cheneh@ustc.edu.cn (E. Chen), minwang@google.com (M. Wang).

http://dx.doi.org/10.1016/j.ins.2015.12.036
0020-0255/

this bound violates the constraint, then all supersets of X should be pruned. Thus, the bound estimation of pattern measures becomes one of the key tasks for constrained pattern mining.

Previous studies on constrained pattern mining frequently followed the research paradigm, where each new pattern measure is proposed individually (motivated by real-world applications) with its corresponding bound estimation method. For example, the measures of pattern utility [27,46], pattern occupancy [39], block pattern measures [18], tiling [19], and weight confidence [47] are all introduced individually. When researchers address a new pattern measure, they need to invest effort in designing efficient solutions for bound estimation. Thus, we argue, it is urgent to analyze the commonalities among these estimation methods and determine whether it is feasible to find a general framework that is applicable to the bound estimation of any pattern measure. With the increasing interest in constrained pattern mining, this general framework is extremely necessary to guide the fast development of the new constrained pattern mining algorithms with the new pattern measures.

To this end, we propose a unified framework to estimate the bound of any pattern measure. Specifically, given an itemset X and a pattern measure $\mathcal{PM}$, the proposed framework can guide the efficient estimation of the bound of $\mathcal{PM}$ for all the frequent patterns containing X. The key point of this framework is that, to achieve efficient estimation, we consider only the weights of item occurrences and ignore the item labels in the supporting transactions. Then, the estimation of the pattern measure bound is equivalent to the process in which we assign the item labels to the item occurrences so that the pattern measure in the resultant transactions is maximized or minimized.

In this study, we focus on the pattern measures that have two inputs, namely, the supporting transactions of a pattern and the item weights for all item occurrences. To the best of our knowledge, this general form covers all pattern measures proposed so far. Here, we consider the constraint with the most commonly used form $\mathcal{PM}(X)\theta\beta$, where $\theta \in \{\geq, \leq\}$ and $\beta \in \mathcal{R}^+$. Note that the other forms of constraints are outside the scope of this study. We first consider the case of $\theta =$ "$\geq$" for the constraint $\mathcal{PM}(X)\theta\beta$. Thus, we need only to estimate the *upper bound* of the pattern measures for pruning the search space. Then, we show that a similar idea can also be applied to estimate the *lower bound* of the pattern measures for the constraints with $\theta =$ "$\leq$." The contributions of our work can be summarized as follows.

- We argue that the bound estimation methods used in previous constrained pattern mining methods are rather dispersive (each new pattern measure is proposed individually with its corresponding bound estimation method) and lack a general framework that is applicable to the bound estimation of any pattern measure.
- We formulate the bound estimation problem *without the item labels*, and then, we propose a generic framework to efficiently address this problem for any pattern measure. We theoretically prove that the proposed general framework can give the tightest bound when the item labels in the supporting transactions are ignored. Additionally, we also propose techniques that achieve the balance between bound tightness and computational efficiency that is needed to improve the overall performance.
- To show the manner in which our proposed framework can guide the development of efficient bound estimation, we apply the proposed framework to two typical pattern measures, namely, *utility* [46] and *occupancy* [39], as a case study. In addition, we extend the traditional SQL-like pattern measures [10,34], such as *min, max, avg*, and *var*, to the corresponding *relative pattern measures* (see Section 2.3), and we also explain the application of the proposed techniques to these pattern measures.
- We systematically evaluate the extent to which the proposed techniques of bound estimation can improve the overall efficiency for the task of *mining top-k constrained frequent closed patterns*. The experiments were conducted on real and large synthetic datasets under different data characteristics and different problem settings.

The rest of the paper is organized as follows. In Section 2, we first introduce the background of pattern mining, in which the properties of all the commonly used pattern measures are summarized and the problem of *top-k constrained frequent closed pattern mining* is proposed. Then, we give the formal problem formulation of bound estimation of pattern measures in Section 3 and present its general solution in Section 4. We give two typical examples of pattern measures, i.e., utility and occupancy, and their bound estimation methods in Sections 5 and 6, respectively. We report our empirical studies to show the efficiency improvements achieved by the proposed techniques in Section 7. We present related work in Section 8 and conclude the paper in Section 9.

2. Background of pattern mining

In this section, we first give some notions and notations of pattern mining. Then, we give a summary of the constraints on the commonly used pattern measures proposed thus far. In addition, we extend the traditional SQL-like pattern measures, such as *min, max, avg*, and *var* to the corresponding relative forms. Finally, we describe the task of *top-k constrained frequent closed pattern mining* to educe the key problem of the bound estimation of pattern measures.

2.1. Notions and notations for pattern mining

A transaction database is a set of transactions, where each transaction is a set of items. Let $\mathcal{I}$ be the complete set of distinct items and $\mathcal{T}$ be the complete set of transactions. Any non-empty set of items is called an *itemset*, while any set

Table 1
Transaction database.

t_{id}	*Transaction*	$\mathcal{W}(t_i)$
t_1	a[4] b[3] c[5] e[2]	14
t_2	a[5] b[6] c[5]	16
t_3	a[5] b[3] c[5] e[3]	16
t_4	a[6] b[6] c[5]	17
t_5	a[3] b[4] c[5] d[2] e[4]	18
t_6	a[3] b[5] d[3] e[3] f[4]	18
t_7	a[6] b[3] e[3] f[2] g[3] h[3]	20
t_8	a[5] b[3] e[2] g[3] h[4] i[3]	20

of transactions is called a *transaction set*. The transactions that contain all the items in an itemset X are the supporting transactions of X, denoted by $\mathcal{T}_X$, i.e., the supporting transaction database of X. The *frequency* of an itemset X (denoted by $freq(X)$) is the number of transactions in $\mathcal{T}_X$, while the *support* of X is defined as $\sigma(X) = freq(X)/|\mathcal{T}|$. For a given minimum support threshold α ($0 < \alpha \leq 1$), X is said to be *frequent* if $\sigma(X) \geq \alpha$.

In certain application contexts, such as market-basket analysis, each transaction-item pair is usually associated with a weight, e.g., sales or profits obtained by selling the items. Thus, a *weight function* $\mathcal{W}$ is used to store the map of weights to items in the transaction database, which is defined on the cross product of the transaction set and itemset, denoted by $\mathcal{W} : 2^{\mathcal{T}} \times 2^{\mathcal{I}} \to \mathcal{R}_0^+$, where $\mathcal{R}_0^+$ is the set of non-negative real numbers.

Table 1 shows an example of a transaction database, where there are nine unique items $\{a, b, c, d, e, f, g, h, i\}$ ($|\mathcal{I}| = 9$) and eight transactions $\{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8\}$ ($|\mathcal{T}| = 8$). Suppose $X = \{a, b, c\}$, the supporting transaction database of X is $\mathcal{T}_X = \{t_1, t_2, t_3, t_4, t_5\}$, $freq(X) = 5$, and $\sigma(X) = 5/8$. X is frequent if α is set to 0.5. Note that the number in the square brackets followed by any item is the weight of the item occurrence in a transaction. For example, $\mathcal{W}(t_1, b)$ is 3 and $\mathcal{W}(t_4, b)$ is 6. The weight of an itemset X in a transaction t_j is defined as $\mathcal{W}(t_j, X) = \sum_{i \in X} \mathcal{W}(t_j, i)$, e.g., $\mathcal{W}(t_5, \{a, b, c\}) = 12$. The weight of a transaction t_j is defined as $\mathcal{W}(t_j) = \sum_{i \in t_j} \mathcal{W}(t_j, i)$, e.g., $\mathcal{W}(t_5) = 18$.

In this paper, we focus on the pattern measures, the computation of which is determined by the supporting transactions of a pattern and the item weight for each item occurrence in the supporting transactions. Namely, to compute the value of a pattern measure $\mathcal{PM}$ for an itemset X, we usually need its supporting transaction database $\mathcal{T}_X$ and the weight function $\mathcal{W}$. Thus, the *pattern measure* can be defined as

$$\mathcal{PM}(X, \mathcal{T}_X, \mathcal{W}) : 2^{\mathcal{I}} \times 2^{\mathcal{T}} \times (2^{\mathcal{I}} \times 2^{\mathcal{T}} \to \mathcal{R}_0^+) \to \mathcal{R}_0^+.$$

In the following, we give two interesting pattern measures motivated by real-world applications, i.e., *utiltiy* [46] and *occupancy* [39]. These two measures are used in the case study to show the proposed framework of bound estimation of pattern measures.

Definition 1 (Utility)**.** The *utility* of an itemset X is defined as

$$\mu(X, \mathcal{T}_X, \mathcal{W}) = \sum_{t \in \mathcal{T}_X} \sum_{i \in X} \mathcal{W}(t, i)$$

The utility of an itemset X is in fact the sum of the weight of X in all its supporting transactions. When the weight of each item occurrence represents the profit on this item, the utility of an itemset is in fact the total profit made by this itemset.

Definition 2 (Occupancy)**.** The *occupancy* of an itemset X is defined as

$$\phi(X, \mathcal{T}_X, \mathcal{W}) = \frac{1}{|\mathcal{T}_X|} \sum_{t \in \mathcal{T}_X} \frac{\sum_{i \in X} \mathcal{W}(t, i)}{\mathcal{W}(t)}$$

The occupancy of an itemset X is the average value on the quotient of the weights of X and its supporting transactions. In fact, it measures the extent to which X occupies its supporting transactions in terms of item weights. Interested readers can find the applications that motivate its use in [39].

For the transaction database in Table 1, we can calculate the utility and occupancy of the three itemsets $X_1 = \{a, b, c\}$, $X_2 = \{a, b, c, d\}$, and $X_3 = \{a, b, c, d, e\}$, respectively. Note that $X_1 \subset X_2 \subset X_3$, $\mathcal{T}_{X_1} = \{t_1, t_2, t_3, t_4, t_5\}$, $\mathcal{T}_{X_2} = \{t_5\}$, and $\mathcal{T}_{X_3} = \{t_5\}$.

$$\mu(X_1, \mathcal{T}_{X_1}, \mathcal{W}) = 12 + 16 + 13 + 17 + 12 = 70,$$
$$\mu(X_2, \mathcal{T}_{X_2}, \mathcal{W}) = 14,$$
$$\mu(X_3, \mathcal{T}_{X_3}, \mathcal{W}) = 18.$$
$$\phi(X_1, \mathcal{T}_{X_1}, \mathcal{W}) = \frac{1}{5} \cdot \left(\frac{12}{14} + \frac{16}{16} + \frac{13}{16} + \frac{17}{17} + \frac{12}{18} \right) \approx 0.87,$$

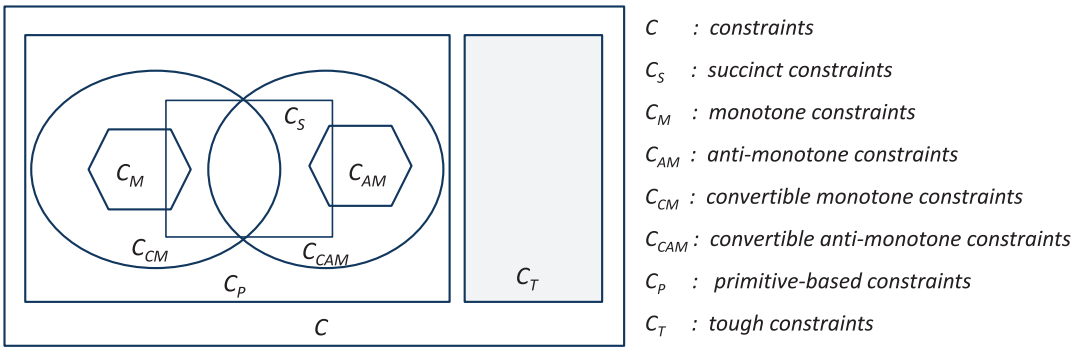


Fig. 1. Comparison of the classes of itemset constraints.

$$\phi(X_2, \mathcal{T}_{X_2}, \mathcal{W}) = \frac{14}{18} \approx 0.78,$$
$$\phi(X_3, \mathcal{T}_{X_3}, \mathcal{W}) = \frac{18}{18} = 1.$$

The above examples easily verify that the value of utility (or occupancy) does not increase or decrease monotonically when more items are appended to a given pattern.

A *pattern measure constraint* can be defined as a predicate:

$$\mathcal{C}(X, \mathcal{T}_X, \mathcal{W}) : 2^{\mathcal{I}} \times 2^{\mathcal{T}} \times (2^{\mathcal{I}} \times 2^{\mathcal{T}} \to \mathcal{R}_0^+) \to \{true, false\}$$

More specifically, the constraint can be formulated as

$$\mathcal{PM}(X, \mathcal{T}_X, \mathcal{W})\theta\beta$$

where $\theta \in \{\geq, \leq\}$ and $\beta \in \mathcal{R}^+$. Note that in this study we consider only the constraints in this form.

2.2. Summary of different constraints

There are many different constraints on pattern measures with different properties, such as *monotonicity* [8,29], *succinctness* [29], *convertible monotonicity* [34], *loose anti-monotonicity* [10], *flexible* [37,44], and *boundable* [35]. Fig. 1 summarizes the relationship of different properties of constraints.

One possible solution for constraints that are neither *anti-monotone* nor *monotone* is to convert them into anti-monotone or monotone ones (if they satisfy *convertible monotonicity*), or to loosen the constraints (if they satisfy *loose anti-monotonicity*), or split them into primitive-based ones (if they are *flexible*). If a constraint is neither anti-monotone nor monotone, cannot be converted to anti-monotone or monotone constraint, and cannot be split into primitive-based ones, we call this constraint a ***tough*** constraint.

For example, the constraints on utility and occupancy introduced in Section 2.1 are tough constraints. To prune the search space for these tough pattern measures (shown in the right grey rectangle in Fig. 1), a usual approach is to estimate the upper (resp. lower) bound on a given pattern measure $\mathcal{PM}$ for all the supersets of an itemset X. If the bound value is smaller (resp. larger) than the given specified minimal (resp. maximal) threshold, the subtree with the root node X in the lexicographic tree (see Fig. 2) should be pruned.

Table 2 gives a classification of commonly used constraints. These constraints are adopted from SQL-based constraints [10,34] (Nos. 1–15), primitive-based constraints [37] (Nos. 16–20), and other real applications motivated constraints [18,39,46] (Nos. 21–24).

2.3. Relative definitions of SQL-like pattern measures

Note that in the definition of occupancy ($\phi(X, \mathcal{T}_X, \mathcal{W}) = \frac{1}{|\mathcal{T}_X|} \sum_{t \in \mathcal{T}_X} \frac{\sum_{i \in X} \mathcal{W}(t,i)}{\mathcal{W}(t)}$), the local occupancy value of X in each of its supporting transactions ($\frac{\sum_{i \in X} \mathcal{W}(t,i)}{\mathcal{W}(t)}$) may be different. In the database shown in Table 1, suppose $X = \{a, b, c\}$ and the local occupancy of X in t_1 is $\frac{12}{14}$ and that in t_3 is $\frac{13}{16}$. Then, the final occupancy value is the average of these local values. In fact, the occupancy is a *relative* measure that captures the completeness of a pattern, which is better than the *absolute* measure *maximal length* of maximal frequent patterns in the application of print-area recommendation [39]. In fact, SQL-like pattern measures, such as *min*, *max*, *sum*, *avg*, *median*, *quartile*, and *var*, are all absolute pattern measures, where the attribute (or weight) of each item in each of its supporting transactions are all equal.

However, in certain application contexts, such as market-basket analysis, each transaction-item pair is usually associated with a weight, e.g., sales or profits obtained by selling a number of the item. Since different people may buy a different

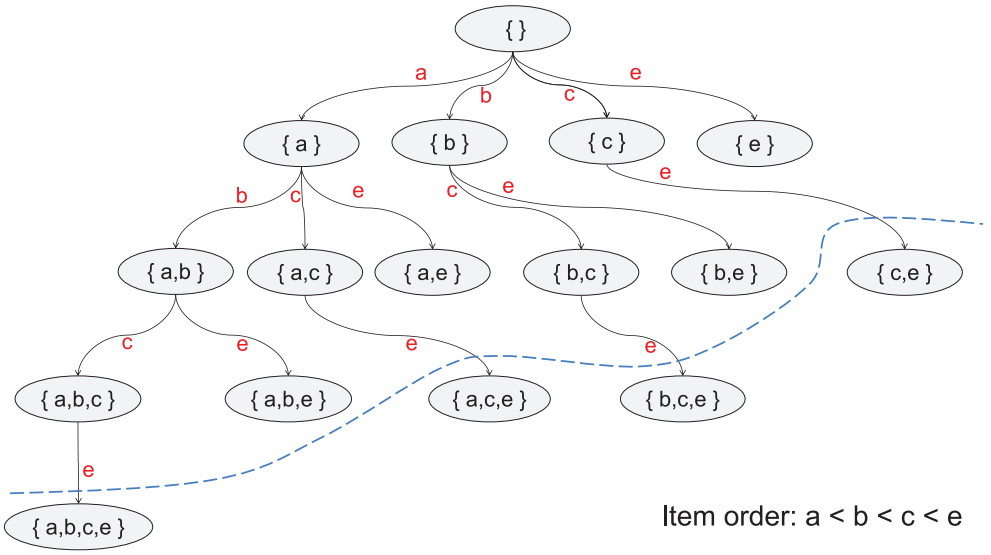


Fig. 2. Set-enumeration tree.

Table 2
Classification of commonly used constraints.

No.	*Constraints*	$\mathcal{C}_{AM}$	$\mathcal{C}_{A}$	$\mathcal{C}_{S}$	$\mathcal{C}_{CAM}$	$\mathcal{C}_{CM}$	$\mathcal{C}_{LAM}$	$\mathcal{C}_{FLE}$
1	$freq(X) \geq v$	yes	no	weakly	yes	no	yes	yes
2	$freq(X) \leq v$	no	yes	weakly	no	yes	no	yes
3	$count(X) \geq v$	no	yes	weakly	no	yes	no	yes
4	$count(X) \leq v$	yes	no	weakly	yes	no	yes	yes
5	$min(X.attr) \geq v$	yes	no	yes	yes	yes	yes	yes
6	$min(X.attr) \leq v$	no	yes	yes	yes	yes	yes	yes
7	$max(X.attr) \geq v$	no	yes	yes	yes	yes	yes	yes
8	$max(X.attr) \leq v$	yes	no	yes	yes	yes	yes	yes
9	$sum(X^{+}.attr) \geq v$	no	yes	no	no	yes	no	yes
10	$sum(X^{+}.attr) \leq v$	yes	no	no	yes	no	yes	yes
11	$avg(X.attr)\theta v$	no	no	no	yes	yes	yes	yes
12	$median(X.attr)\theta v$	no	no	no	yes	yes	yes	yes
13	$quartile(X.attr)\theta v$	no	no	no	yes	yes	yes	yes
14	$var(X.attr)\theta v$	no	no	no	no	no	yes	no
15	$md(X.attr)\theta v$	no	no	no	no	no	yes	no
16	$M_1 = \frac{min(X.attr)+max(X.attr)}{2}\theta v$	no	no	no	no	no	no	yes
17	$M_2 = \frac{sum(X^{+}.attr)}{count(X)}\theta v$	no	no	no	no	no	no	yes
18	$M_3 = \frac{freq(X)}{count(X)} \geq v$	yes	no	no	yes	no	yes	yes
19	$M_3 = \frac{freq(X)}{count(X)} \leq v$	no	yes	no	no	yes	no	yes
20	$area(X) = freq(X) \times count(X)\theta v$	no	no	no	no	no	no	yes
21	$BSize(X)\theta v$	no	no	no	no	no	no	yes
22	$BSum(X^{+}.attr)\theta v$	no	no	no	no	no	no	no
23	$utility(X^{+}.attr)\theta v$	no	no	no	no	no	no	no
24	$occupancy(X^{+}.attr)\theta v$	no	no	no	no	no	no	no

Note: X is a set of items with positive, zero, or negative numerical attribute values; X^{+} is a set of items with only non-negative numerical attribute values.

number of the item in one transaction, the weight of an item in each of its supporting transactions may be different. In other words, for a given itemset X, the SQL-like pattern measures of X in each of its supporting transactions may have different values. For example, as shown in Table 1, the *sum* of $\{a, b, c\}$ in t_1 and t_3 are 12 and 13, respectively. In that case, users may be interested in the average values of these SQL-like pattern measure values for the supporting transactions. Thus, we redefine these absolute pattern measures according to their corresponding relative pattern measures. For example, the relative pattern measure of *sum* is defined as

$$sum^{*}(X, \mathcal{T}_X, \mathcal{W}) = \frac{1}{|\mathcal{T}_X|} \sum_{t \in \mathcal{T}_X} \sum_{i \in X} \mathcal{W}(t, i) \tag{1}$$

Table 3
Relative definitions of commonly used pattern measures.

#	Relative measures
1	$min^*(X, \mathcal{T}_X, \mathcal{W}) = \frac{1}{\lvert\mathcal{T}_X\rvert}\sum_{t\in\mathcal{T}_X} \min_{i\in X} \mathcal{W}(t,i)$
2	$max^*(X, \mathcal{T}_X, \mathcal{W}) = \frac{1}{\lvert\mathcal{T}_X\rvert}\sum_{t\in\mathcal{T}_X} \max_{i\in X} \mathcal{W}(t,i)$
3	$sum^*(X, \mathcal{T}_X, \mathcal{W}) = \frac{1}{\lvert\mathcal{T}_X\rvert}\sum_{t\in\mathcal{T}_X} \sum_{i\in X} \mathcal{W}(t,i)$
4	$avg^*(X, \mathcal{T}_X, \mathcal{W}) = \frac{1}{\lvert\mathcal{T}_X\rvert}\sum_{t\in\mathcal{T}_X} \frac{1}{\lvert X\rvert}\sum_{i\in X} \mathcal{W}(t,i)$
5	$median^*(X, \mathcal{T}_X, \mathcal{W}) = \frac{1}{\lvert\mathcal{T}_X\rvert}\sum_{t\in\mathcal{T}_X} median_{i\in X}\mathcal{W}(t,i)$
6	$quartile^*(X, \mathcal{T}_X, \mathcal{W}) = \frac{1}{\lvert\mathcal{T}_X\rvert}\sum_{t\in\mathcal{T}_X} quartile_{i\in X}\mathcal{W}(t,i)$
7	$var^*(X, \mathcal{T}_X, \mathcal{W}) = \frac{1}{\lvert\mathcal{T}_X\rvert}\sum_{t\in\mathcal{T}_X} \frac{\sum_{i\in X}(\mathcal{W}(t,i)-avg(X))^2}{\lvert X\rvert}$
8	$std^*(X, \mathcal{T}_X, \mathcal{W}) = \frac{1}{\lvert\mathcal{T}_X\rvert}\sum_{t\in\mathcal{T}_X} \sqrt{\frac{\sum_{i\in X}(\mathcal{W}(t,i)-avg(X))^2}{\lvert X\rvert}}$
9	$var^*_{N-1}(X, \mathcal{T}_X, \mathcal{W}) = \frac{1}{\lvert\mathcal{T}_X\rvert}\sum_{t\in\mathcal{T}_X} \frac{\sum_{i\in X}(\mathcal{W}(t,i)-avg(X))^2}{\lvert X\rvert-1}$
10	$md^*(X, \mathcal{T}_X, \mathcal{W}) = \frac{1}{\lvert\mathcal{T}_X\rvert}\sum_{t\in\mathcal{T}_X} \frac{\sum_{i\in X}\lvert\mathcal{W}(t,i)-avg(X)\rvert}{\lvert X\rvert}$
11	$M_1^* = (\frac{min(X)+max(X)}{2})^* = \frac{1}{\lvert\mathcal{T}_X\rvert}\sum_{t\in\mathcal{T}_X} \frac{\min_{i\in X}\mathcal{W}(t,i)+\max_{i\in X}\mathcal{W}(t,i)}{2}$
12	$M_2^* = (\frac{sum(X)}{count(X)})^* = \frac{1}{\lvert\mathcal{T}_X\rvert}\sum_{t\in\mathcal{T}_X} \frac{\sum_{i\in X}\mathcal{W}(t,i)}{\lvert X\rvert}$

As compared with the pattern *utility* used to measure the total profits of a pattern in its supporting transactions, sum^* is used to measure the average profits brought by each user. The relative definitions for these commonly used pattern measures are summarized in Table 3. Note that the constraints on these relative pattern measures in this table are all *tough* ones.

2.4. Top-k constrained frequent closed pattern mining

In this study, we consider the following constrained pattern mining task. Its efficiency is strongly dependent on the performance of the bound estimation of pattern measures.

Top-k Constrained Frequent Closed Pattern Mining. Given a database $\mathcal{T}$, a minimum support threshold α, and a constraint $\mathcal{C}: \mathcal{PM}_1(X, \mathcal{T}_X, \mathcal{W})\theta\beta$, the problem of *constrained frequent closed pattern mining* is to find the complete set of *qualified itemsets* that are both frequent closed and satisfy $\mathcal{C}$.

In the above task, two pattern measures, namely, *support* and $\mathcal{PM}_1$, are used to identify all the qualified itemsets. Sometimes, we also need to rank all these qualified itemsets by another pattern measure, $\mathcal{PM}_2$ (usually the combination of *support* and $\mathcal{PM}_1$) and identify the top-k qualified itemsets in terms of the $\mathcal{PM}_2$ values. This leads to the problem of *top-k constrained frequent closed pattern mining*.

General Solution to Pattern Mining. The search space of the frequent pattern mining problem can be represented as a set-enumeration tree. Given a set of items $\mathcal{I} = \{i_1, i_2, ..., i_n\}$ and a total order of all items (suppose the item order $i_1 < i_2 < ... < i_n$), a set-enumeration tree can be constructed as follows. First, the root of the tree is created; second, the n child nodes of the root representing $n1$-itemsets are created, respectively; and third, for any node representing itemset $\{i_b, ..., i_e\}$ $(1 \le b \le e < n)$, the $(n-e)$ child nodes of the node representing itemsets $\{i_b, ..., i_e, i_{e+1}\}$, $\{i_b, ..., i_e, i_{e+2}\}$, ..., $\{i_b, ..., i_e, i_n\}$ are created. The third step is performed repeatedly until all leaf nodes are created.

Let us use the database in Table 1 as a running example. Suppose $freq_{min} = 3$. We remove all infrequent items (i.e., $\{d, f, g, h, i\}$) and sort frequent items in lexical order $(a < b < c < e)$. Then, the set-enumeration tree is shown in Fig. 2. The solution for mining the top-k qualified patterns usually adopts a depth-first search (DFS) with pruning techniques. Algorithm 1 gives the *top-k qualified pattern mining* (TKQ-Miner) process with the pruning techniques.

It is clear that the key step in the above pattern mining process is to estimate the upper bound of pattern measures for all supersets of a given itemset X. In the following sections, we formulate the problem of the upper bound estimation of pattern measures, and then, propose a general framework for it.

3. Pattern measure bound estimation without item labels

Given an itemset X and a pattern measure $\mathcal{PM}$, the task is to estimate the upper bound of $\mathcal{PM}$ for all supersets of X. For example, for the node $\{a, b\}$ in Fig. 2, this task is to estimate the upper bound of $\mathcal{PM}$ for all nodes in this subtree, including nodes $\{a, b\}$, $\{a, b, c\}$, $\{a, b, e\}$, and $\{a, b, c, e\}$.

A straightforward solution is to compute the pattern measures of all these itemsets and then select the biggest value as the exact upper bound. However, this approach is extremely computation intensive, since the computation of a pattern measure needs the supporting transactions of each itemset. Thus, we require that this estimation method be both efficient and effective. To this end, we consider only the weights of item occurrences in this estimation and ignore the item labels in the supporting transactions. Without the original item labels, we can assign any item labels to the items in the transaction, and the upper bound of the pattern measure is achieved when the value of the pattern measure in the resultant transactions with the new item labels is maximized.

Algorithm 1: TKQ-Miner (DFS).

input : the current node *node*,the heap H to maintain the top-k qualified itemsets searched(DFS) thus far.

1 $node_k \leftarrow$ the node with the $k$thlargest value on $\mathcal{PM}_2$ in H;
2 **if** $node.support \geq \alpha \wedge node.\mathcal{PM}_1 \geq \beta$ **then**
3 **if** $node.quality \geq node_k.quality$ **then**
4 replace $node_k$ with *node* in H;
5 **for** $node_c \in node.children$ **do**
6 $supp \leftarrow node_c.support$;
7 $clos \leftarrow$true if there exist closed patterns in the subtree rooted at $node_c$ and vice versa ;
8 $ub_1 \leftarrow$the upper bound on $\mathcal{PM}_1$ for the subtree rooted at $node_c$;
9 $ub_2 \leftarrow$the upper bound on $\mathcal{PM}_2$ for the subtree rooted at $node_c$;
10 **if** $(supp \geq \alpha) \wedge (clos == true) \wedge (ub_1 \geq \beta) \wedge (ub_2 \geq node_k.quality)$ **then**
11 TKQ-Miner ($node_c$, H);

a

t_{id}	transaction
t_1	a[4] b[3] c[5] e[2]
t_2	a[5] b[6] c[5]
t_3	a[5] b[3] c[5] e[3]
t_4	a[6] b[6] c[5]
t_5	a[3] b[4] c[5] d[2] e[4]
t_6	a[3] b[5] d[3] e[3] f[4]
t_7	a[6] b[3] e[3] f[2] g[3] h[3]
t_8	a[5] b[3] e[2] g[3] h[4] i[3]

b

t_{id}	Prefix	Suffix
t_1	a[4] b[3]	c[5] e[2]
t_2	a[5] b[6]	c[5]
t_3	a[5] b[3]	c[5] e[3]
t_4	a[6] b[6]	c[5]
t_5	a[3] b[4]	c[5] d[2] e[4]
t_6	a[3] b[5]	d[3] e[3] f[4]
t_7	a[6] b[3]	e[3] f[2] g[3] h[3]
t_8	a[5] b[3]	e[2] g[3] h[4] i[3]

c

t_{id}	Prefix	Suffix
t_1	[4] [3]	[5] [2]
t_2	[5] [6]	[5]
t_3	[5] [3]	[5] [3]
t_4	[6] [6]	[5]
t_5	[3] [4]	[5] [2] [4]
t_6	[3] [5]	[3] [3] [4]
t_7	[6] [3]	[3] [2] [3] [3]
t_8	[5] [3]	[2] [3] [4] [3]

d

t_{id}	PW	PL	SW	SL
t_1	< 4, 3 >	2	< 5,2 >	2
t_2	< 5, 6 >	2	< 5 >	1
t_3	< 5, 3 >	2	< 5,3 >	2
t_4	< 6, 6 >	2	< 5 >	1
t_5	< 3, 4 >	2	< 5, 2, 4 >	3
t_6	< 3, 5 >	2	< 5, 3, 4>	3
t_7	< 6, 3 >	2	< 3, 2, 3, 3 >	4
t_8	< 5, 3 >	2	< 2, 3, 4, 3>	4

Fig. 3. (a) Supporting transaction database $\mathcal{T}_{\{a,b\}}$. (b) Prefix itemsets and suffix itemsets of $\{a, b\}$ in $\mathcal{T}_{\{a,b\}}$. (c) Supporting transaction database $\mathcal{T}_{\{a,b\}}$ without item labels. (d) Statistical information obtained from (c).

Before elaborating this method, we first give some notions and notations, based on which we formulate the problem of *pattern measure bound estimation without item labels*.

3.1. Notions and notations for bound estimation

Given a set-enumeration tree and an itemset X, the supporting transaction database $\mathcal{T}_X$ and the weight function $\mathcal{W}$, we give the following definitions with respect to X.

In the set-enumeration tree, each itemset X' in the subtree rooted at X is called an *extension* of X. It is clear that X' is a superset of X. Suppose $|X' - X| = v$. We state that X' is a *v-extension* of X and v is the *extension length* of X' as compared to X.

For example, in the subtree rooted at $\{a, b\}$ in Fig. 2, $\{a, b, c\}$ is the 1-extension of $\{a, b\}$, $\{a, b, c, e\}$ is the 2-extension of $\{a, b\}$, and the extension length of $\{a, b, c, e\}$ as compared to $\{a, b\}$ is 2.

For any transaction t in $\mathcal{T}_X$ (note that each transaction in $\mathcal{T}_X$ is ordered according to the enumeration order used in Fig. 2), the *prefix itemset* of X in t is the appearance of X in t, while its *suffix itemset* is the remaining itemset in t, except X (denoted as t/X). For the supporting transaction database of $\{a, b\}$ in Fig. 3(a), the prefix and suffix itemset of $\{a, b\}$ in t_1 is $a[4]b[3]$ and $c[5]e[2]$, respectively. Fig. 3(b) shows all the prefix itemsets and suffix itemsets of $\{a, b\}$ in its supporting transactions. If we do not consider the item labels in $\mathcal{T}_X$ and consider only the weight of each item occurrence, then we obtain Fig. 3(c). Next, we define the following notions.

Definition 3 (PW & SW)**.** The *prefix itemset weight matrix* of X, denoted as PW, is used to record the weight vector of each prefix itemset in t ($t \in \mathcal{T}_X$). The *suffix itemset weight matrix* of X, denoted by SW, is used to record the weight vector of each suffix itemset in t ($t \in \mathcal{T}_X$).

Note that PW and SW are two matrices. Their first index refers to the transaction number and the second to the index of an item. As shown in Fig. 3(d), $\mathrm{PW}[5][2] = 4$ and $\mathrm{SW}[5][2] = 2$.

Definition 4 (PL & SL)**.** The *prefix itemset length vector* of X, denoted by PL, is used to record the number of items in each prefix itemset in t ($t \in \mathcal{T}_X$). The *suffix itemset length vector* of X, denoted by SL, is used to record the number of items in each suffix itemset in t ($t \in \mathcal{T}_X$).

PL and SL are two vectors. Their index refers to the transaction number. As shown in Fig. 3(d), $\mathsf{PL}[5] = 2$ and $\mathsf{SL}[5] = 3$. It is clear that PL and SL can be directly obtained from PW and SW. Their definitions are given only for convenience of description.

In the following, we need to frequently sort the values in a vector. For a vector V, $\mathsf{V}^{\downarrow}$ ($\mathsf{V}^{\uparrow}$) is the vector sorted in descending (ascending) order by values in V. Thus, $\mathsf{V}^{\downarrow}(i)$ is the ith biggest (smallest) value in V. For example, $\mathsf{SL}^{\downarrow}(1) = 4$ and $\mathsf{SL}^{\uparrow}(1) = 1$.

3.2. Problem formulation

Ignoring the item labels of the supporting transactions, we still have statistical information about them, as shown in Fig. 3(d). With only these statistical data, we formulate the problem of pattern measure bound estimation as follows.

Given an itemset X, the PW, SW *of X, and a pattern measure $\mathcal{PM}$, the problem is to estimate the upper bound of $\mathcal{PM}$ for all supersets of X.*

4. General framework

In this section, we present a generic framework to address the problem of bound estimation without item labels.

4.1. Basic idea

Given an itemset X, the PW, SW of X and a pattern measure $\mathcal{PM}$, the basic idea of this bound estimation is briefly described as follows.

First, let X' be any v-extension of X and u be the frequency of X'. For X', we propose a function $F(u, v, \mathsf{PW}, \mathsf{SW})$, which satisfies the following conditions.

$$\mathcal{PM}(X', \mathcal{T}_{X'}, \mathcal{W}) \leq F(u, v, \mathsf{PW}, \mathsf{SW}) \tag{2}$$

Next, we can enumerate all possible u, v for the above F function and then obtain the upper bound for any superset of X. That is, for any superset X' of X, we have

$$\mathcal{PM}(X', \mathcal{T}_{X'}, \mathcal{W}) \leq \max_{u,v} F(u, v, \mathsf{PW}, \mathsf{SW}) \tag{3}$$

The right part of Inequation (3) is in fact the upper bound on $\mathcal{PM}$ for all supersets of X.

It is very time-consuming to compute the upper bound $\max_{u,v} F(u, v, \mathsf{PW}, \mathsf{SW})$, since we have to enumerate all the possible values of u, v for the computation of $F(u, v, \mathsf{PW}, \mathsf{SW})$. In the following, we propose some properties to prune this enumeration space.

4.2. Proposed techniques

4.2.1. Relationship between u and v

For a superset X' of X, let u be the frequency and v the extension length of X'. Then, for a given value of u we can estimate the range of v for all the possible supersets of X according to SL. It is easy to prove that this range of v is $[\underline{v}(u), \hat{v}(u)]$, where $\underline{v}(u) = 0$ and $\hat{v}(u) = \mathsf{SL}^{\downarrow}(u)$.

As the example shown in Fig. 3(d), $\mathsf{SL}^{\downarrow} = \langle 4, 4, 3, 3, 2, 2, 1, 1\rangle$. Then, we have

$$\hat{v}(1) = 4, \quad \hat{v}(2) = 4, \quad \hat{v}(3) = 3, \quad \hat{v}(4) = 3$$

$\hat{v}(4) = 3$ informs us that the extension length of any superset of X with a frequency of 4 cannot be bigger than 3. Otherwise, if the maximal extension length of X' is larger than 3, this superset must not exist. For example, when $v = 4$, there are only two transactions (t_7 and t_8) that may contain X', since the length of the suffix itemset of X in t_7 and t_8 is not less than v (i.e., 4). In other words, the frequency of X' should be no larger than 2 and the frequency of X' is supposed to be 4; thus, a contradiction occurs.

Similarly, for a given value of v we can estimate the range of u for all the possible frequent supersets of X according to SL. It is easy to prove that this range of u is $[\underline{u}(v), \hat{u}(v)]$, where $\underline{u}(v) = freq_{min}$ and $\hat{u}(v) = \max\{u | \mathsf{SL}^{\downarrow}(u) \geq v\}$.

As the example also shown in Fig. 3(d), we have

$$\hat{u}(0) = 8, \quad \hat{u}(1) = 8, \quad \hat{u}(2) = 6, \quad \hat{u}(3) = 4$$

$\hat{u}(2) = 6$ informs us that the frequency of any superset of X's with an extension length of 2 cannot be bigger than 6.

Based on the relationship of u and v, the upper bound for any frequent superset of X is

$$\max_{u,v} F(u, v, \mathsf{PW}, \mathsf{SW}) = \max_{freq_{min} \leq u \leq |\mathcal{T}_X|, 0 \leq v \leq \hat{v}(u)} F(u, v, \mathsf{PW}, \mathsf{SW}) \tag{4}$$

Based on this upper bound, the search space can be greatly pruned. For example, Fig. 4 gives the search space of u, v for the estimation task in Fig. 3(d). As shown in this figure, we need to check only 18 points (the points with the circles). Originally, we needed to check 30 points ($3 \leq u \leq 8$ and $0 \leq v \leq 4$).

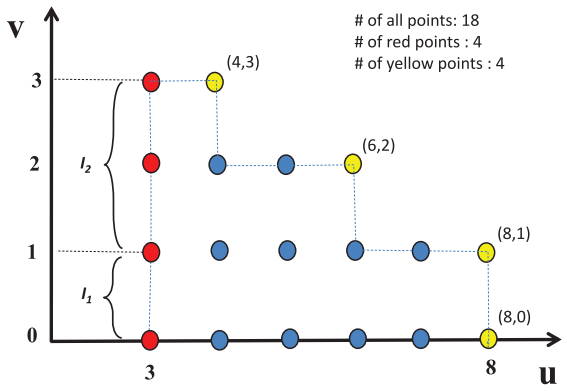


Fig. 4. Search space for computing upper bound by enumerating each u, v s.t. $3 \le u \le 8$ and $0 \le v \le \hat{v}(u)$.

4.2.2. Properties of $F(u, v, PW, SW)$ with respect to u

We can also explore the properties of $F(u, v, \mathrm{PW}, \mathrm{SW})$ to further prune the search space. For some pattern measures, the proposed F function monotonically increases or decreases with respect to u (Sections 5 and 6 give concrete examples of such functions). In the following, we propose properties that can be used to further reduce the search space.

Property 1. When the F function monotonically increases with respect to u, i.e.,

$$F(u, v, \mathrm{PW}, \mathrm{SW}) \le F(u+1, v, \mathrm{PW}, \mathrm{SW}), \tag{5}$$

we have

$$\mathcal{PM}(X', \mathcal{T}_{X'}, \mathcal{W}) \le \max_{0 \le v \le \hat{v}(freq_{min})} F(\hat{u}(v), v, \mathrm{PW}, \mathrm{SW}) \tag{6}$$

for any superset X' of X, where $\hat{u}(v) = \max\{u | \mathrm{SL}^{\downarrow}(u) \ge v\}$.

Property 2. When the F function monotonically decreases with respect to u, i.e.,

$$F(u, v, \mathrm{PW}, \mathrm{SW}) \ge F(u+1, v, \mathrm{PW}, \mathrm{SW}), \tag{7}$$

we have

$$\mathcal{PM}(X', \mathcal{T}_{X'}, \mathcal{W}) \le \max_{0 \le v \le \hat{v}(freq_{min})} F(\underline{u}(v), v, \mathrm{PW}, \mathrm{SW})) \tag{8}$$

for any superset X' of X, where $\underline{u}(v) = freq_{min}$.

Properties 1 and 2 give two kinds of pruning method. If the condition in Property 1 holds, we can check only the points on the right-most boundary (the four yellow points in Fig. 4). If the condition in Property 2 holds, we can check only the points on the left-most boundary (the four red points in Fig. 4). Clearly, using Properties 1 or 2 can further reduce the search space.

It is worth mentioning that the upper bounds proposed in Properties 1 and 2 are the tightest upper bounds for any frequent superset of X if we use the values of only PW, SW to compute the bounds. Their proofs can be found in Section 5 and Section 6, respectively.

4.2.3. Tradeoff between tightness and efficiency

To alleviate this situation that we still have many points that need to be checked, we propose a new technique that achieves a balance between the bound tightness and computational efficiency. The basic idea is as follows. Instead of enumerating every possible value of v in the range of $[0, \hat{v}(freq_{min})]$, we split this large interval into m smaller intervals $[v_0, v_1 - 1]$, $\cdots$, $[v_{m-1}, v_m - 1]$ ($v_0 = 0, v_m = \hat{v}(freq_{min}) + 1$). Then, for each interval $[v_{k-1}, v_k - 1]$ we can obtain a loose bound on $\mathcal{PM}$, namely, $F(u, v_{k-1}, v_k - 1, \mathrm{PW}, \mathrm{SW})$. Finally, we unify the m (much smaller than $\hat{v}(freq_{min})$) bounds and obtain a final result. Using a proper value of m, we achieve the tradeoff between the efficiency and tightness of the bound with the following property.

Property 3. Let X' be any frequent superset of X. For any integers

$$0 = v_0 < v_1 < \cdots < v_m = \hat{v}(freq_{min}) + 1,$$

we have

$$\mathcal{PM}(X', \mathcal{T}_{X'}, \mathcal{W}) \le \max_{u, 0 < k \le m} \hat{F}(u, v_{k-1}, v_k - 1, \mathrm{PW}, \mathrm{SW}) \tag{9}$$

Table 4
Notations used in this paper.

X	Root pattern for a subtree
X'	Any frequent pattern in the subtree of X
$\mathcal{T}_X$	Supporting transactions of X
$freq_{min}$	Minimal frequency threshold ($\lceil \alpha \cdot \lvert\mathcal{T}\rvert \rceil$)
PI	Prefix itemset of X in t
SI	Suffix itemset of X in t
PL	Prefix itemset length vector of X, $\mathsf{PL}(i) = \lvert\mathsf{PI}_i\rvert$
SL	Suffix itemset length vector of X, $\mathsf{SL}(i) = \lvert\mathsf{SI}_i\rvert$
PW	Prefix itemset weight matrix of X
SW	Suffix itemset weight matrix of X
u	u frequently denotes $\lvert\mathcal{T}_{X'}\rvert$, i.e., the frequency of X'
v	v frequently denotes $\lvert X'\rvert - \lvert X\rvert$, i.e., the extension length of X' w.r.t. X
$\underline{u}(v)$	Minimal u when v is fixed, $\underline{u}(v) = freq_{min}$
$\widehat{u}(v)$	Maximal u when v is fixed, $\widehat{u}(v) = \max\{u \mid \mathsf{SL}^{\downarrow}(u) \geq v\}$
$\underline{v}(u)$	Minimal v when u is fixed, $\underline{v}(u) = 0$
$\widehat{v}(u)$	Maximal v when u is fixed, $\widehat{v}(u) = \mathsf{SL}^{\downarrow}(u)$
$\mathsf{V}^{\uparrow/\downarrow}$	Vector V sorted in ascending/descending order

Note that m in Inequation (9) controls the tightness of the upper bound. For example, the bound becomes "tightest" when m is set to 1 and loosest when m is set to $+\infty$. It is worth mentioning that the tightest upper bound has a good pruning effect in the search process, but its computational complexity is high (it needs $\widehat{v}(freq_{min}) + 1$ times computation of $\widehat{F}$). In contrast, the loosest upper bound has a worse pruning effect, but its computational complexity is low (it needs only one computation of $\widehat{F}$). The overall running time is proportional to the pruning effect, while inversely proportional to the cost of the computation of the upper bounds. As shown in Fig. 4, instead of enumerating every possible value of v in the range of [0, 3], we split it into two intervals, i.e., $I_1 = [0, 1]$ and $I_2 = (1, 3]$. Although the search space is reduced to 2, the upper bound is loose.

Note that the techniques in Sections 4.2.1, 4.2.2, and 4.2.3 can be jointly used to prune the search space. In fact, the techniques proposed in Sections 4.2.1 and 4.2.3 are independent of any pattern measure. However, the technique proposed in Section 4.2.2 depends on $F(u, v, \mathsf{PW}, \mathsf{SW})$ for different pattern measures. The notations used in this paper are summarized in Table 4. In the following, we show the manner in which these techniques can be applied to any pattern measure.

4.3. Summary of upper bounds for pattern measures

Table 5 gives a summary of the upper bounds for commonly used pattern measures obtained by using the above proposed techniques under a unified framework. Note that $freq(X)$ and $count(X)$ are two basic measures. The upper bounds for $freq(X)$ and $count(X)$ are $\lvert\mathcal{T}_X\rvert$ and $\lvert X\rvert + \widehat{v}(freq_{min})$, respectively. The proposed upper bounds for pattern measures Nos. 3 to 19 in Table 5 can be the "tightest" ones if we do not consider the item labels in the supporting transactions. We can see that the upper bounds on pattern measures Nos. 3 to 15 adopt the techniques in Section 4.2.1 and Property 2 in Section 4.2.2, in which the pattern measure *occupancy* is a typical one. However, the upper bounds on pattern measures Nos. 16 to 20 adopt the techniques in Section 4.2.1 and Property 1 in Section 4.2.2, in which the pattern measure *utility* is a representative one (note that the pattern measure No. 17 (No. 18) is the special case of the pattern measure No. 19 (No. 20) when the weights of PW and SW are all set to 1). Furthermore, we can apply the tradeoff techniques proposed in Section 4.2.3 to these pattern measures to further improve the efficiency.

As a case study, for two typical *tough* pattern measures, *utility* and *occupancy*, we show how the function $F(u, v, \mathsf{PW}, \mathsf{SW})$ and their corresponding upper bounds are developed in Section 5 and Section 6, respectively.

5. Utility

In this section, we develop the F function for the pattern measure of *utility*. We can also theoretically prove that the proposed utility upper bound is the tightest one if we do not consider the item labels in the supporting transactions.

5.1. Basic idea

We are given a pattern measure $\mathcal{PM}$, the data of PW, SW. For an itemset X' with frequency u and extension length v, we aim to propose a function F s.t. $\mathcal{PM}(X', \mathcal{T}_{X'}, \mathcal{W}) \leq F(u, v, \mathsf{PW}, \mathsf{SW})$. The basic idea is as follows. Since we do not know the item labels in the supporting transactions, we are free to assign any labels to them so that the value of the pattern measure for the data of PW, SW is maximized. In other words, from the entries in the matrix SW, we attempt to identify u rows such that each row contains at least v entries and the value of the pattern measure on these $u \times v$ entries reaches its maximum.

Table 5
Summary of upper bounds for commonly used pattern measures.

No.	Pattern measures	Upper bounds
1	$freq(X)$	$\lvert\mathcal{T}_X\rvert$
2	$count(X)c$	$\lvert X\rvert + \widehat{v}(freq_{min})$
3	$min^*(X)$	$\max_{0\le v\le\widehat{v}(freq_{min})}\{F_1(\underline{u}(v),v,PW,SW) = \max_{l_1,l_2,\ldots,l_{\underline{u}(v)}EL[l_i]\ge v}\frac{1}{\underline{u}(v)}\sum_{i=1}^{\underline{u}(v)}\min(X\cup SW[l_i]^{\downarrow}[1]\ldots\cup SW[l_i]^{\downarrow}[v])\}$
4	$max^*(X)$	$\max_{0\le v\le\widehat{v}(freq_{min})}\{F_2(\underline{u}(v),v,PW,SW) = \max_{l_1,l_2,\ldots,l_{\underline{u}(v)}EL[l_i]\ge v}\frac{1}{\underline{u}(v)}\sum_{i=1}^{\underline{u}(v)}\max(X\cup SW[l_i]^{\downarrow}[1]\ldots\cup SW[l_i]^{\downarrow}[v])\ \}$
5	$sum^*(X)$	$\max_{0\le v\le\widehat{v}(freq_{min})}\{F_3(\underline{u}(v),v,PW,SW) = \max_{l_1,l_2,\ldots,l_{\underline{u}(v)}EL[l_i]\ge v}\frac{1}{\underline{u}(v)}\sum_{i=1}^{\underline{u}(v)}(\sum_{j=1}^{PL[l_i]}PW[l_i][j]+\sum_{j=1}^{v}SW[l_i]^{\downarrow}[j])\}$
6	$avg^*(X)$	$\max_{0\le v\le\widehat{v}(freq_{min})}\{F_4(\underline{u}(v),v,PW,SW) = \max_{l_1,l_2,\ldots,l_{\underline{u}(v)}EL[l_i]\ge v}\frac{1}{\underline{u}(v)}\sum_{i=1}^{\underline{u}(v)}\frac{\sum_{j=1}^{PL[l_i]}PW[l_i][j]+\sum_{j=1}^{v}SW[l_i]^{\downarrow}[j]}{\lvert X\rvert+v}\}$
7	$median^*(X)$	$\max_{0\le v\le\widehat{v}(freq_{min})}\{F_5(\underline{u}(v),v,PW,SW) = \max_{l_1,l_2,\ldots,l_{\underline{u}(v)}EL[l_i]\ge v}\frac{1}{\underline{u}(v)}\sum_{i=1}^{\underline{u}(v)}median(X\cup SW[l_i]^{\downarrow}[1]\ldots\cup SW[l_i]^{\downarrow}[v])\}$
8	$quartile^*(X)$	$\max_{0\le v\le\widehat{v}(freq_{min})}\{F_6(\underline{u}(v),v,PW,SW) = \max_{l_1,l_2,\ldots,l_{\underline{u}(v)}EL[l_i]\ge v}\frac{1}{\underline{u}(v)}\sum_{i=1}^{\underline{u}(v)}quartile(X\cup SW[l_i]^{\downarrow}[1]\ldots\cup SW[l_i]^{\downarrow}[v])\}$
9	$var^*(X)$	$\max_{0\le v\le\widehat{v}(freq_{min})}\{F_7(\underline{u}(v),v,PW,SW) = \max_{l_1,l_2,\ldots,l_{\underline{u}(v)}EL[l_i]\ge v}\frac{1}{\underline{u}(v)}\sum_{i=1}^{\underline{u}(v)}\max_{j=1}^{v}var(X\cup E(j,v)^*\}$
10	$std^*(X)$	$\max_{0\le v\le\widehat{v}(freq_{min})}\{F_8(\underline{u}(v),v,PW,SW) = \max_{l_1,l_2,\ldots,l_{\underline{u}(v)}EL[l_i]\ge v}\frac{1}{\underline{u}(v)}\sum_{i=1}^{\underline{u}(v)}\max_{j=1}^{v}std(X\cup E(j,v)^*\}$
11	$var^*_{N-1}(X)$	$\max_{0\le v\le\widehat{v}(freq_{min})}\{F_9(\underline{u}(v),v,PW,SW) = \max_{l_1,l_2,\ldots,l_{\underline{u}(v)}EL[l_i]\ge v}\frac{1}{\underline{u}(v)}\sum_{i=1}^{\underline{u}(v)}\max_{j=1}^{v}var_{N-1}(X\cup E(j,v)^*\}$
12	$md^*(X)$	$\max_{0\le v\le\widehat{v}(freq_{min})}\{F_{10}(\underline{u}(v),v,PW,SW) = \max_{l_1,l_2,\ldots,l_{\underline{u}(v)}EL[l_i]\ge v}\frac{1}{\underline{u}(v)}\sum_{i=1}^{\underline{u}(v)}\max_{j=1}^{v}md(X\cup E(j,v)^*\}$
13	$M_1^* = (\frac{min(X)+max(X)}{2})^*$	$\max_{0\le v\le\widehat{v}(freq_{min})}\{F_{11}(\underline{u}(v),v,PW,SW) = \max_{l_1,l_2,\ldots,l_{\underline{u}(v)}EL[l_i]\ge v}\frac{1}{\underline{u}(v)}\sum_{i=1}^{\underline{u}(v)}\frac{\min(X\cup SW[l_i]^{\downarrow}[1])+\max(X\cup SW[l_i]^{\downarrow}[1]}{2}\}$
14	$M_2^* = (\frac{sum(X)}{count(X)})^*$	$\max_{0\le v\le\widehat{v}(freq_{min})}\{F_{12}(\underline{u}(v),v,PW,SW) = \max_{l_1,l_2,\ldots,l_{\underline{u}(v)}EL[l_i]\ge v}\frac{1}{\underline{u}(v)}\sum_{i=1}^{\underline{u}(v)}\frac{\sum_{j=1}^{PL[l_i]}PW[l_i][j]+\sum_{j=1}^{v}SW[l_i]^{\downarrow}[j]}{\lvert X\rvert+v}\}$
15	**occupancy(X)**	$\max_{0\le v\le\widehat{v}(freq_{min})}\{F_{13}(\underline{u}(v),v,PW,SW) = \max_{l_1,l_2,\ldots,l_{\underline{u}(v)}EL[l_i]\ge v}\frac{1}{\underline{u}(v)}\sum_{i=1}^{\underline{u}(v)}\frac{\sum_{j=1}^{PL[l_i]}PW[l_i][j]+\sum_{j=1}^{v}SW[l_i]^{\downarrow}[j]}{\sum_{j=1}^{PL[l_i]}PW[l_i][j]+\sum_{j=1}^{SL[l_i]}SW[l_i][j]}\}$
16	$M_3 = \frac{freq(X)}{count(X)}$	$\max_{0\le v\le\widehat{v}(freq_{min})}\{F_{14}(\widehat{u}(v),v,PW,SW) = \max_{l_1,l_2,\ldots,l_{\widehat{u}(v)}EL[l_i]\ge v}\frac{\widehat{u}(v)}{\lvert X\rvert+v}\}$
17	$area = freq(X)\times count(X)$	$\max_{0\le v\le\widehat{v}(freq_{min})}\{F_{15}(\widehat{u}(v),v,PW,SW) = \max_{l_1,l_2,\ldots,l_{\widehat{u}(v)}EL[l_i]\ge v}\sum_{i=1}^{\widehat{u}(v)}(\lvert X\rvert+v)\}$
18	$BSize(X)$	$\max_{0\le v\le\widehat{v}(freq_{min})}\{F_{15}(\widehat{u}(v),v,PW,SW) = \max_{l_1,l_2,\ldots,l_{\widehat{u}(v)}EL[l_i]\ge v}\sum_{i=1}^{\widehat{u}(v)}(\lvert X\rvert+v)\}$
19	$BSum(X)$	$\max_{0\le v\le\widehat{v}(freq_{min})}\{F_{16}(\widehat{u}(v),v,PW,SW) = \max_{l_1,l_2,\ldots,l_{\widehat{u}(v)}EL[l_i]\ge v}\sum_{i=1}^{\widehat{u}(v)}(\sum_{j=1}^{PL[l_i]}PW[l_i][j]+\sum_{j=1}^{v}SW[l_i]^{\downarrow}[j])\}$
20	**utility(X)**	$\max_{0\le v\le\widehat{v}(freq_{min})}\{F_{16}(\widehat{u}(v),v,PW,SW) = \max_{l_1,l_2,\ldots,l_{\widehat{u}(v)}EL[l_i]\ge v}\sum_{i=1}^{\widehat{u}(v)}(\sum_{j=1}^{PL[l_i]}PW[l_i][j]+\sum_{j=1}^{v}SW[l_i]^{\downarrow}[j])\}$

Note that $E(j,v)^*$ is a v-length set consisting of the top j biggest values and top $(v-j)$ smallest values from $\mathsf{SW}[l_i]$.

Table 6
Steps for computing $F(3,2,\mathsf{PW},\mathsf{SW})$ on utility.

t_{id}	PW	SW	Step (1)	Step (2)	Step (3)
t_1	⟨ **4**, **3** ⟩	⟨ **5**, **2** ⟩	✓	14	
t_2	⟨ 5, 6 ⟩	⟨ 5 ⟩			
t_3	⟨ **5**, **3** ⟩	⟨ **5**, **3** ⟩	✓	16	✓
t_4	⟨ 6, 6 ⟩	⟨ 5 ⟩			
t_5	⟨ **3**, **4** ⟩	⟨ **5**, 2, **4** ⟩	✓	16	✓
t_6	⟨ **3**, **5** ⟩	⟨ **3**, 3, **4** ⟩	✓	15	✓
t_7	⟨ **6**, **3** ⟩	⟨ **3**, 2, **3**, 3 ⟩	✓	15	
t_8	⟨ **5**, **3** ⟩	⟨ 2, **3**, **4**, 3 ⟩	✓	15	

We give the following example to show how this upper bound is developed for the measure of utility. Given the data of PW, SW in Table 6, we aim to find the utility upper bound of any itemset X' with frequency 3 (i.e., $u=3$) and extension length 2 (i.e., $v=2$). The procedure is presented as follows.

(1) Identify the transactions that have a suffix itemset containing at least two items. If the length of the suffix itemset in a transaction is less than 2, this transaction cannot contain X'. Thus, six transactions, t_1, t_3, t_5, t_6, t_7, and t_8, are selected, while transactions t_2 and t_4 are filtered out.
(2) In order to obtain the maximal value for the utility for each selected transaction, we can select the top two biggest weights from the suffix itemset. For example, for the transaction t_5, the entries of 5 and 4 (in bold) are selected. Thus, the maximal utility value in this transaction is $(3+4)+(5+4)$. We conduct this process for each selected transaction from Step 1 and obtain its possibly maximal utility value, as shown in the 5th column of Table 6.

(3) Finally, from the six transactions we need to identify the three that have the top three biggest utility values. Then, the transactions t_3, t_5, and t_6 are selected. By summing up these three values, we get this upper bound, i.e., $F(3, 2, \mathsf{PW}, \mathsf{SW}) = 16 + 16 + 15 = 47$.

5.2. $F(u, v, \mathsf{PW}, \mathsf{SW})$ for utility

By translating the above procedure into mathematical language, we obtain the F function for the utility upper bound.

Property 4. Let X' be any v-extension of X with frequency u ($|\mathcal{T}_{X'}| = u$). Then, the utility of X' satisfies that

$$\mu(X', \mathcal{T}_{X'}, \mathcal{W}) \le F(u, v, \mathsf{PW}, \mathsf{SW}) \tag{10}$$

where

$$F(u, v, \mathsf{PW}, \mathsf{SW}) = \max_{\substack{l_1, l_2, \ldots, l_u \\ \mathsf{SL}[l_i] \ge v}} \sum_{i=1}^{u} \left(\sum_{j=1}^{\mathsf{PL}[l_i]} \mathsf{PW}[l_i][j] + \sum_{j=1}^{v} \mathsf{SW}[l_i]^{\downarrow}[j] \right) \tag{11}$$

Proof. Let us consider the utility of X',

$$\mu(X', \mathcal{T}_{X'}, \mathcal{W}) = \sum_{t \subseteq \mathcal{T}_{X'}} \sum_{i \in X'} \mathcal{W}(t, i) \tag{12}$$

$$= \sum_{t \subseteq \mathcal{T}_{X'}} \left(\sum_{i \in X} \mathcal{W}(t, i) + \sum_{i \in X'/X} \mathcal{W}(t, i) \right) \tag{13}$$

$$\le \sum_{t \subseteq \mathcal{T}_{X'}} \left(\sum_{j=1}^{\mathsf{PL}[l_t]} \mathsf{PW}[l_t][j] + \sum_{j=1}^{v} \mathsf{SW}[l_t]^{\downarrow}[j] \right) \tag{14}$$

$$\le \max_{\substack{l_1, l_2, \ldots, l_u \\ \mathsf{SL}[l_i] \ge v}} \sum_{i=1}^{u} \left(\sum_{j=1}^{\mathsf{PL}[l_i]} \mathsf{PW}[l_i][j] + \sum_{j=1}^{v} \mathsf{SW}[l_i]^{\downarrow}[j] \right) \tag{15}$$

Note that l_t is the corresponding index of t in PW and SW. Since X' is a v-extension of X ($|X' - X| = |X'/X| = v$) for any $t \in \mathcal{T}_{X'}$, $\sum_{i \in (X'/X)} \mathcal{W}(t, i) \le \sum_{j=1}^{v} \mathsf{SW}[l_t]^{\downarrow}[j] \le \sum_{i \in (t/X)} \mathcal{W}(t, i)$, Inequality (14) follows. Since $X \subseteq X'$, by the anti-monotonicity of frequent pattern, $\mathcal{T}_{X'} \subseteq \mathcal{T}_X$, and for any supporting transaction t of X', the length of the suffix itemset of X in t should be no less than v, i.e., $\mathsf{SL}[l_t] \ge v$. Thus, Inequality (15) follows. □

We can also prove that this F function monotonically increases with respect to u, as shown in the following property.

Property 5. If F is the function defined in Eq. (11), then we have $F(u + 1, v, \mathsf{PW}, \mathsf{SW}) \ge F(u, v, \mathsf{PW}, \mathsf{SW})$.

Proof. *Let* $M_i = \max_{l_i}(\sum_{j=1}^{\mathsf{PL}[l_i]} \mathsf{PW}[l_i][j] + \sum_{j=1}^{v} \mathsf{SW}[l_i]^{\downarrow}[j])$,

$$F(u + 1, v, \mathsf{PW}, \mathsf{SW}) - F(u, v, \mathsf{PW}, \mathsf{SW}) \tag{16}$$

$$= \sum_{i=1}^{u+1} M_i - \sum_{i=1}^{u} M_i \tag{17}$$

$$= \sum_{i=1}^{u} M_i + M_{u+1} - \sum_{i=1}^{u} M_i \tag{18}$$

$$\ge 0 \tag{19}$$

Inequality (19) follows since $M_{u+1} \ge 0$. □

With Properties 4 and 5, we easily obtain the utility upper bound for all the supersets of X, denoted by $\widehat{Util}(X)$,

$$\widehat{Util}(X) = \max_{0 \le v \le \hat{v}(freq_{min})} F(\hat{u}(v), v, \mathsf{PW}, \mathsf{SW}) \tag{20}$$

The following example shows the manner in which this upper bound is computed for the data in Table 6. Suppose $freq_{min} = 3$. Then, $\hat{v}(freq_{min}) = 3$. We can also obtain $\hat{u}(0) = 8$, $\hat{u}(1) = 8$, $\hat{u}(2) = 6$, and $\hat{u}(3) = 4$. According to Eq. (20), the utility upper bound for all the frequent supersets of $\{a, b\}$ is

$$\widehat{Util}(\{a, b\}) = \max_{0 \le v \le 3} F(\hat{u}(v), v, \mathsf{PW}, \mathsf{SW})$$

According to Eq. (11), we can obtain

$$F(8, 0, \text{PW}, \text{SW}) = 7 + 11 + 8 + 12 + 7 + 8 + 9 + 8 = 70$$
$$F(8, 1, \text{PW}, \text{SW}) = 12 + 16 + 13 + 17 + 12 + 12 + 12 + 12 = 106$$
$$F(6, 2, \text{PW}, \text{SW}) = 14 + 16 + 16 + 15 + 15 + 15 = 91$$
$$F(4, 3, \text{PW}, \text{SW}) = 18 + 18 + 18 + 18 = 72$$

Thus, $\widehat{U}(\{a, b\}) = \max\{70, 106, 91, 72\} = 106$. This means that the utility of all the frequent supersets of $\{a, b\}$, including $\{a, b\}$, $\{a, b, c\}$, and $\{a, b, e\}$, is less than 106.

In addition, we can theoretically prove that $\widehat{Util}(X)$ is the tightest upper bound if the values of only PW and SW of X are used in the computation. Namely, we have the following property.

Property 6. Given an itemset X and the PW, SW of X, $\widehat{Util}(X)$ is the tightest utility upper bound for any superset of X.

Proof. We prove this by contradiction. Suppose that there exists another upper bound $\widetilde{F}$ s.t. $\widetilde{F}(u, v, \text{PW}, \text{SW}) < F(u, v, \text{PW}, \text{SW})$ for any given parameters of u, v, PW, SW. Then, we can construct a transaction database $\mathcal{T}_X$, in which there exists any itemset X' in the subtree rooted at X s.t. $|\mathcal{T}_{X'}| = u$ and $\mu(X') = F(u, v, \text{PW}, \text{SW})$. If we apply the upper bound $\widetilde{F}$ to the data of $\mathcal{T}_X$, we have that the utility of any itemset X' is no bigger than $\widetilde{F}(u, v, \text{PW}, \text{SW})$. However, we have an itemset X' in this subtree s.t. $\mu(X') = F(u, v, \text{PW}, \text{SW})$ and $F(u, v, \text{PW}, \text{SW}) \leq \widetilde{F}(u, v, \text{PW}, \text{SW})$, and thus, the contradiction occurs. The procedure in Table 6 in fact generates the supporting transactions, on which the pattern measure is exactly equal to this upper bound. Therefore, $\widehat{Util}(X)$ is the tightest one. □

5.3. Mining top-k high utility and frequent closed itemsets

In this section, we show the problem of mining top-k constrained frequent closed pattern mining shown in Section 2.4, where $\mathcal{PM}_1$ is the *utility* and $\mathcal{PM}_2$ is the weighted sum of relative utility and support, defined as follows.

Given a transaction database $\mathcal{T}$ and weight function $\mathcal{W}$, the *utility constraint* of an itemset X is defined as $\mu(X) \geq \beta \cdot W$, where $0 < \beta \leq 1$ and W is the sum of the weights of all the transaction-item pairs in the database. An itemset X is called a *high utility itemset* if it satisfies the utility constraint.

Definition 5 (U-Quality)**.** The *u-quality* of an itemset X is defined as $q^u(X) = \sigma(X) + \lambda \cdot \frac{\mu(X)}{W}$, where $W = \sum_{t \in \mathcal{T}, i \in \mathcal{I}} \mathcal{W}(t, i)$, $\sigma(.)$ denotes the support of X, $\frac{\mu(X)}{W}$ is the ratio of X's utility to the total utility of transactions, and the weight λ ($0 \leq \lambda < +\infty$) is a user-defined parameter to capture the relative importance of support and *relative utility*.

In this task, the most challenging part is to compute the upper bounds on utility and u-quality. Since the upper bound on utility is given in Eq. (20), here, we give only the upper bound on u-quality by the following property.

Property 7 (U-Quality Upper Bound)**.** Let X' be any superset of X,

$$q^u(X') \leq \frac{|\mathcal{T}_X|}{|\mathcal{T}|} + \lambda \cdot \frac{\widehat{Util}(X)}{W} \tag{21}$$

Proof. Note that $q^u(X') = \sigma(X') + \lambda \cdot \frac{\mu(X')}{W}$, since $\sigma(X') \leq \sigma(X) = \frac{|\mathcal{T}_X|}{|\mathcal{T}|}$ and $\frac{\mu(X')}{W} \leq \frac{\widehat{Util}(X)}{W}$; thus, the conclusion holds. □

6. Occupancy

In this section, we develop the F function for the pattern measure of occupancy.

6.1. $F(u, v, \text{PW}, \text{SW})$ for occupancy

The basic idea for computing $F(u, v, \text{PW}, \text{SW})$ is similar to that described in Section 5.1, and we also give an example to show how this upper bound is developed for the measure of occupancy. Given the data PW, SW in Table 7, we aim to find the occupancy upper bound of any itemset X' with frequency 3 ($u = 3$) and extension length 3 ($v = 3$). The procedure is as follows.

(1) Identify the transactions having a suffix itemset that contains at least three items. Thus, four transactions, t_5, t_6, t_7, and t_8 are selected, while transactions t_1, t_2, t_3, and t_4 are filtered out.
(2) In order to obtain the maximal value for the occupancy for each selected transaction, we can select the top three biggest weights from the suffix itemset. For example, for transaction t_8 the entries of 4, 3 and 3 (in bold) are selected. Thus, the maximal occupancy value in this transaction is $[(5 + 3) + (4 + 3 + 3)]/20$. We conduct this process for each selected transaction from Step 1.
(3) Finally, from the four transactions we need to identify the three with the top three biggest occupancy values. Then, the transactions t_5, t_6, and t_7 are selected. By averaging these three values, we get this upper bound, i.e., $F(3, 3, \text{PW}, \text{SW}) = (16/16 + 16/16 + 18/20)/3 \approx 0.97$.

Table 7
Steps for $F(3, 3, \mathsf{PW}, \mathsf{SW})$ on occupancy.

t_{id}	PW	SW	Step (1)	Step (2)	Step (3)
t_1	⟨ 4, 3 ⟩	⟨ 5, 2 ⟩			
t_2	⟨ 5, 6 ⟩	⟨ 5 ⟩			
t_3	⟨ 5, 3 ⟩	⟨ 5, 3 ⟩			
t_4	⟨ 6, 6 ⟩	⟨ 5 ⟩			
t_5	⟨ **3**, **4** ⟩	⟨ **5**, **2**, **4** ⟩	✓	16/16	✓
t_6	⟨ **3**, **5** ⟩	⟨ **3**, **3**, **4** ⟩	✓	16/16	✓
t_7	⟨ **6**, **3** ⟩	⟨ **3**, 2, **3**, **3** ⟩	✓	18/20	✓
t_8	⟨ **5**, **3** ⟩	⟨ 2, **3**, **4**, **3** ⟩	✓	18/20	

We propose the F function for the occupancy upper bound with the following properties. Interested readers can refer to [39] for their proofs.

Property 8. Let X' be any v-extension of X with frequency u (i.e., $|\mathcal{T}_{X'}| = u$ and $|X' - X| = v$). Then, the occupancy of X' satisfies that

$$\phi(X', \mathcal{T}_{X'}, \mathcal{W}) \leq F(u, v, \mathsf{PW}, \mathsf{SW}) \tag{22}$$

where

$$F(u, v, \mathsf{PW}, \mathsf{SW}) = \frac{1}{u} \cdot \max_{\substack{l_1, l_2, \ldots, l_u \\ \mathsf{SL}[l_i] \geq v}} \sum_{i=1}^{u} \frac{\sum_{j=1}^{\mathsf{PL}[l_i]} \mathsf{PW}[l_i][j] + \sum_{j=1}^{v} \mathsf{SW}[l_i]^{\downarrow}[j]}{\sum_{j=1}^{\mathsf{PL}[l_i]} \mathsf{PW}[l_i][j] + \sum_{j=1}^{\mathsf{SL}[l_i]} \mathsf{SW}[l_i][j]} \tag{23}$$

We can also prove that this F function monotonically decreases with respect to u, as shown in the following property.

Property 9. When F is the function defined in Eq. (23), we have $F(u+1, v, \mathsf{PW}, \mathsf{SW}) \leq F(u, v, \mathsf{PW}, \mathsf{SW})$

With Properties 8 and 9, we can also easily obtain the occupancy upper bound for all the supersets of X, denoted as $\widehat{Occu}(X)$,

$$\widehat{Occu}(X) = \max_{0 \leq v \leq \hat{v}(freq_{min})} F(freq_{min}, v, \mathsf{PW}, \mathsf{SW}) \tag{24}$$

The following example shows how this upper bound is computed for the data in Table 6. Suppose $freq_{min} = 3$; then, $\hat{v}(freq_{min}) = 3$. According to Eq. (24), the occupancy upper bound for all frequent supersets of $\{a, b\}$ is

$$\widehat{Occu}(\{a, b\}) = \max_{0 \leq v \leq 3} F(3, v, \mathsf{PW}, \mathsf{SW})$$

According to Eq. (23), we can obtain

$$F(3, 0, \mathsf{PW}, \mathsf{SW}) = 1/3 \cdot (7/14 + 11/16 + 12/17) \approx 0.63$$
$$F(3, 1, \mathsf{PW}, \mathsf{SW}) = 1/3 \cdot (12/14 + 16/16 + 17/17) \approx 0.95$$
$$F(3, 2, \mathsf{PW}, \mathsf{SW}) = 1/3 \cdot (14/14 + 16/16 + 16/18) \approx 0.96$$
$$F(3, 3, \mathsf{PW}, \mathsf{SW}) = 1/3 \cdot (18/18 + 18/18 + 18/20) \approx 0.97$$

Thus, $\widehat{O}(\{a, b\}) = \max\{0.63, 0.95, 0.96, 0.97\} = 0.97$. This means that the occupancy of all the frequent supersets of $\{a, b\}$, including $\{a, b\}$, $\{a, b, c\}$, and $\{a, b, e\}$, is less than 0.97.

In addition, we can theoretically prove that $\widehat{Occu}(X)$ is the tightest upper bound if the values of only PW and SW of X are used in the computation. Namely, we have the following property.

Property 10. Given an itemset X and the PW, SW of X, $\widehat{Occu}(X)$ is the tightest upper bound on occupancy for any superset of X.

6.2. Mining top-k high occupancy and frequent closed itemsets

In this section, we show an instance of the top-k constrained closed frequent pattern mining shown in Section 2.4, where $\mathcal{PM}_1$ is the *occupancy* and $\mathcal{PM}_2$ is the weighted sum of occupancy and support.

Given a transaction database $\mathcal{T}$ and a weight function $\mathcal{W}$, the *occupancy constraint* of an itemset X is defined as $\phi(X) \geq \beta$, where $0 < \beta \leq 1$. An itemset X is called a *high occupancy itemset* if it satisfies the occupancy constraint.

Definition 6 (O-Quality)**.** The *o-quality* of an itemset X is defined as $q^o(X) = \sigma(X) + \lambda \cdot \phi(X)$, where $\sigma(.)$ denotes the relative support of X and the weight λ $(0 \leq \lambda < +\infty)$ is a user-defined parameter to capture the relative importance of occupancy and support.

Table 8
Characteristics of datasets.

Datasets	No. of trans	No. of items	AveLen	MaxLen	Type
Chess	3196	75	37	37	Large and dense
Mushroom	8124	119	23	23	Dense
T10I4D100K	100,000	870	10.1	29	Sparse

In this task, the most challenging part is to compute the upper bounds on occupancy and o-quality. Since the upper bound on occupancy is given in Eq. (24), here, we give only the upper bound on o-quality by the following property.

Property 11 (O-Quality Upper Bound)**.** Let X' be any superset of X,

$$q^o(X') \leq \frac{|\mathcal{T}_X|}{|\mathcal{T}|} + \lambda \cdot \widehat{Occu}(X) \tag{25}$$

7. Experimental evaluation

To evaluate the effectiveness of the proposed pruning techniques, we compared the performance of the proposed top-k qualified pattern mining algorithms using the proposed pruning techniques to the baselines. Note that we report only the experimental results for mining top-k high utility and frequent closed itemsets, and that the algorithm for mining top-k high occupancy and frequent closed itemsets achieves similar efficiency results by using the proposed techniques.

7.1. Experimental setup

In this paper, we give the proposed algorithm TKQ-Miner for mining top-k high utility and frequent closed itemsets with the proposed pruning techniques and compare its running time with that of the baseline methods for our problem. The following three baselines are used in this paper.

Baseline-Naive: This baseline method first finds all frequent closed patterns and computes the utility and quality for each of them, and then, outputs the top-k qualified ones. We adopt an efficient pattern-growth-based algorithm BIDE [42] to find all the frequent closed patterns; other efficient algorithms such as CLOSET+ [43] and CHARM [48] can also be used. To make the comparison fair, TKQ-Miner is also based on the adopted baseline algorithm. A comparison of this baseline and TKQ-Miner is shown in Section 7.2.

Baseline-Loose: This baseline method uses the utility upper bound technique used in [18,27]. This pruning technique is efficient for computation, but loose for pruning space. In fact, it is a special case of the proposed framework, namely, the tradeoff technique proposed in Section 4.2.3 with $m = 1$. A comparison of this baseline and TKQ-Miner is shown in Section 7.3.

Baseline-Relax: This baseline method uses the upper bound technique of minimal area constraint presented in [37], in which the automatic relaxation technique is used to further prune the search space. In fact, minimal area constraint is a special case of high utility constraint, if the weights of items in the database are all set to 1 (binary database). In other words, our proposed techniques are sufficiently general to be directly applied in [37], but [37]'s techniques cannot be used in our problem since utility is not a primitive-based constraint. A comparison of this baseline and TKQ-Miner is shown in Section 7.4.

The four algorithms were implemented in C++ language and the experiments were performed using Microsoft Windows 7 on a laptop with and a quad core Intel i5-540M processor and 4GB of main memory.

Three databases[1] were used in our experiments with different characteristics: two real datasets and one synthetic dataset. The first real dataset is *Chess*, which is large and dense. The second dataset is *Mushroom*, which is dense. The final dataset is the generated dataset *T10I4D100K*, which is sparse. Note that these datasets do not include the item weights for each transaction. In order to obtain the corresponding weighted transactional databases, we use a similar method from a previous study [27,40] and generate the weights for items randomly ranging from 1 to 100. The characteristics of the three datasets are summarized in Table 8.

7.2. Evaluation on weighted datasets

In this subsection, we compare the performance of Baseline-Naive and TKQ-Miner. There are four parameters in our problem of mining high utility and frequent closed itemsets: the support threshold α, the relative utility threshold β, the utility weight parameter λ, and the number of top qualified itemsets k. We varied only the minimal frequency threshold α to observe the changes in efficiency, while the other parameters were fixed. According to the different degree of data density, the default parameters (β, λ, k) for *Chess/Mushroom/T10I4D100K* are (*0.4,10,5*), (*0.1,10,5*), and (*0.001,10,5*), respectively.

[1] The three datasets can be downloaded from the FIMI Repository: http://fimi.ua.ac.be/ .

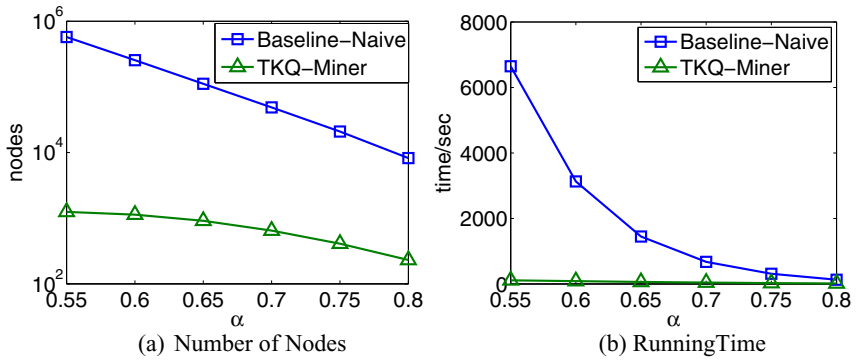


Fig. 5. Performance comparison of Baseline-Naive and TKQ-Miner on *Chess* w.r.t α.

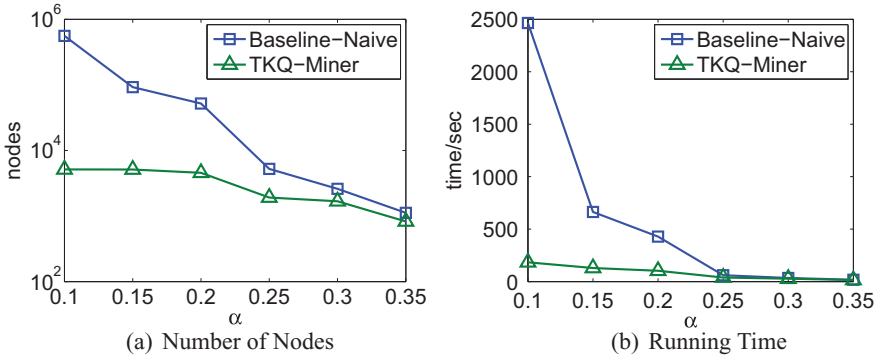


Fig. 6. Performance comparison of Baseline-Naive and TKQ-Miner on *Mushroom* w.r.t α.

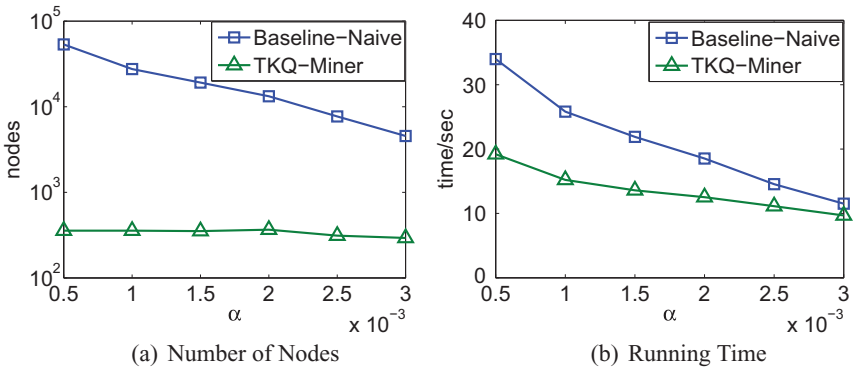
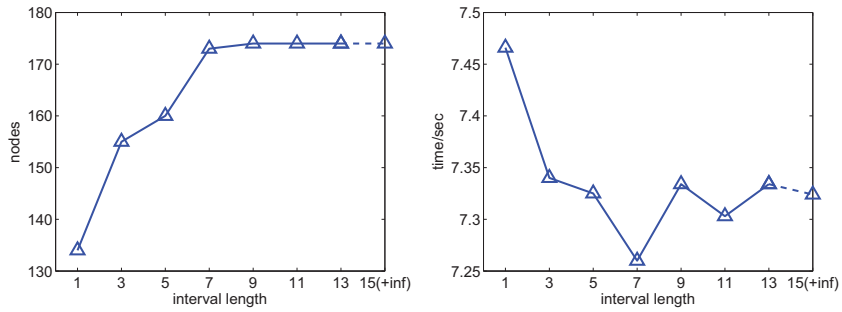


Fig. 7. Performance comparison of Baseline-Naive and TKQ-Miner on *T10I4D100K* w.r.t α.

The number of searched nodes and running time of Baseline-Naive and TKQ-Miner on the three databases are shown in Figs. 5–7, respectively. We can observe that the tendency of these figures is very consistent, that is, the greater the number of searched nodes, the longer the running time. We can also observe that our method outperforms the baseline method, in particular, when the value of α is low. For example, TKQ-Miner runs only over 60 times faster than Baseline-Naive on the larger dense dataset *Chess* with the minimal support threshold α value of 0.55, while it searches only 0.22% of nodes searched by Baseline-Naive. This difference becomes larger when the value of α is less than 0.55. Depending on the different characteristics of the datasets, the pruning effect can be different. Specifically, the effect on the performance gain decreases when the dataset becomes sparser. For example, for the sparse dataset *T10I4D100K* with the minimal support threshold α value of 0.0005, although the number of nodes searched by TKQ-Miner is much smaller than that searched by Baseline-Naive (TKQ-Miner searches only 0.67% of the nodes searched by Baseline-Naive), their running times are almost the same (TKQ-Miner runs only 1.7 times faster than Baseline-Naive). The main reason is that BIDE is very efficient in a sparse database [42] and the running time for the extra searched nodes in the baseline just compensates for the utility



(a) Number of Nodes (b) Running Time

Fig. 8. Tradeoff between bound tightness and computing efficiency for data *T10I4D100K*.

upper bound computation for searched nodes in TKQ-Miner. Briefly, our method can be effective, in particular, for dense datasets.

7.3. Evaluation on tradeoff technique

In this subsection, we compare the performance of Baseline-Loose (when interval length=+inf in Fig. 8) and TKQ-Miner (when interval length=1 in Fig. 8). In Section 4.2.3, we proposed a technique that achieves a tradeoff between bound tightness and computational efficiency. Specifically, instead of enumerating each possible value of v in the range of $[0, \mathrm{SL}^{\downarrow}(freq_{min})]$ as F does, we suggest splitting the large interval into smaller intervals $[v_0, v_1 - 1], \ldots, [v_{m-1}, v_m - 1]$, $(v_0 = 0, v_m = \mathrm{SL}^{\downarrow}(freq_{min}) + 1)$.

Fig. 8 shows the effect of this technique on the database *T10I4D100K* with $\alpha = 0.009$, $\beta = 0.003$, $\lambda = 10$, and $k = 1$. The x-axis shows the "interval length," i.e., $v_i - v_{i-1}$ (we divide the interval evenly) and the y-axis shows the number of nodes searched and the running time in the two figures, respectively. As can be seen, the smaller the interval length, the tighter the bound and the fewer the nodes searched. However, the shortest running time does not occur when the bound is tightest (134 nodes searched when interval length $= 1$), since the computation is too high. Furthermore, the shortest running time does not occur while using the most efficient method used in [18,27] (174 nodes searched when interval length=+inf), since the number of the searched nodes is large. In fact, the overall performance is the tradeoff between bound tightness and computational efficiency. In this case, the algorithm is fastest when the interval length is 7.

7.4. Evaluation on binary datasets

In this subsection, we compare the performance of Baseline-Relax and TKQ-Miner on binary datasets (i.e., the weights of items in database are all set to 1). Baseline-Relax uses the upper bound technique of minimal area constraint in [37], that is, if the frequency of the pattern (node) multiplied by the maximal transaction length in the database is smaller than a minimal threshold, then this node should be pruned. In addition, both TKQ-Miner and Baseline-Relax adopt the automatic relaxation technique in [37] to improve the minimal support threshold α to further prune the search space. Here, we varied only the minimal relative utility threshold β to observe the changes in efficiency, while the other parameters were fixed. The default parameters (α, λ, k) for each dataset were (*0.00001,10,5*).

Figs. 9–11 show the number of searched nodes and running time of Baseline-Relax and TKQ-Miner for the three databases, respectively. We can observe that our method outperforms Baseline-Relax, in particular, when β is low. For example, TKQ-Miner runs over 200 times faster than Baseline-Relax on the very dense dataset *Chess* when the minimal relative utility threshold β is less than 0.5, while it searches only 0.05% of the nodes searched by Baseline-Relax. Similarly to the results in Section 7.2, when the data becomes sparser, the difference in running time between TKQ-Miner and Baseline-Relax becomes smaller. For example, for the sparse dataset *T10I4D100K*, although TKQ-Miner searches only 4.3% of the nodes searched by Baseline-Relax, it runs only 1.2 times faster than Baseline-Relax when the minimal relative utility threshold β is less than 0.0005.

7.5. Discussion

The results of the above experiments show that TKQ-Miner using the proposed techniques outperforms other state-of-the-art techniques, in particular on very dense and large databases. The proposed techniques can give the "tightest" upper bound with certain input constraints. Although the proposed tradeoff technique between bound tightness and computational efficiency can be used to improve the overall performance, the computation of the bound is still high, in particular when the minimal support threshold is set to a low value. However, in the task of "Top-k Constrained Frequent Closed Pattern Mining," the proposed techniques of bound estimation can be used to prune the search space to a great extent. In other

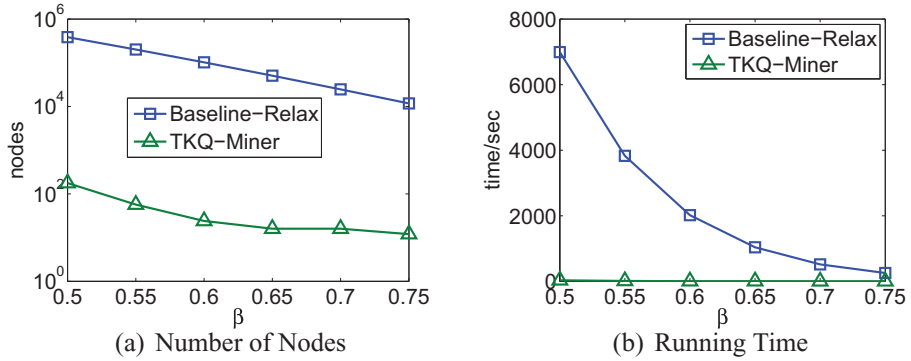


Fig. 9. Performance comparison of Baseline-Relax and TKQ-Miner on data *Chess* w.r.t β.

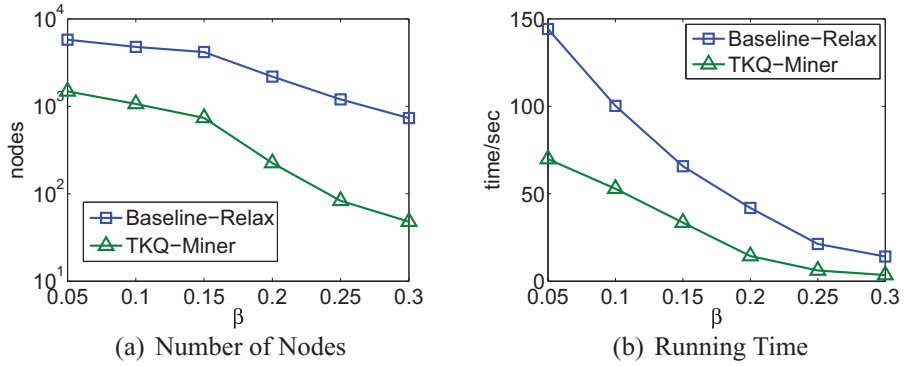


Fig. 10. Performance comparison of Baseline-Relax and TKQ-Miner on data *Mushroom* w.r.t β.

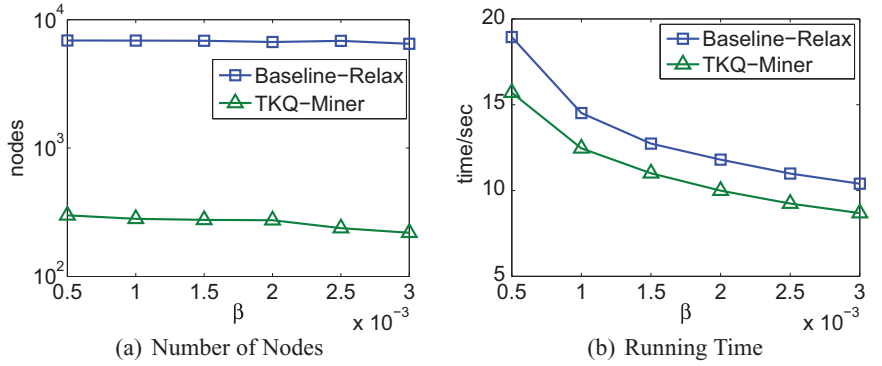


Fig. 11. Performance comparison of Baseline-Relax and TKQ-Miner on data *T10I4D100K* w.r.t β.

words, the overall time for computing the bound in this task is much smaller than that for searching the pruned nodes, and thus, the effectiveness of mining is increased.

In summary, the proposed techniques can be applied not only to pattern utility [46] and pattern occupancy [39], and other tough pattern measures, such as block-sum and block-similarity [18], but also to certain primitive-based constraints, such as minimal area constraint [37], block-size constraint [18], or tilling constraint [19]. Thus, the proposed techniques are sufficiently general and can be applied for the fast development of the new constrained pattern mining algorithms, in particular with the new tough pattern measures.

8. Related work

Mining frequent itemsets [2] is widely recognized to be fundamental for many important data mining tasks. There has been a great amount of work on designing efficient algorithms for mining frequent itemsets [1,3,24,26,31,48], and [23] presents a recent survey of this field. The frequent itemsets mining algorithms usually generate a huge number of frequent

itemsets, while the support is set to a low level. To address this problem, two general classes of methods were proposed. The first class is mining closed or maximal itemsets. For example, CHARM [48], CFP-tree [26], CLOSET+ [43], and Moment [17] are typical algorithms for mining closed itemsets, while MaxMiner [5], MAFIA [13], and GenMax [20] are typical algorithms for mining maximal itemsets.

The second class focuses on mining constrained frequent itemsets by incorporating various constraints with different properties, such as *monotonicity* [6,8,29], *succinctness* [25,29], *convertible monotonicity* [32–34], and *loose anti-monotonicity* [10]. Recently, complex constraints constructed from primitives have also been proposed [15,37,44]. In these works, the primitives are required to be monotonic or anti-monotonic or convex. Specifically, *anti-monotone* constraints can easily be pushed deeply into the frequent itemsets enumeration, since if they are not satisfiable with small itemsets, there is no hope that they will become satisfiable later with larger itemsets [6,29]. However, it is not easy for *monotone* constraints to be pushed in the computation and they proved to be less effective in pruning the search space [7,12]. However, using data reduction techniques, such as ExAnte [9] and ExAMiner [8], *anti-monotone* and *monotone* pruning strengthen each other recursively. A *succinctness* constraint can be satisfied at candidate generation time and is *pre-counting pushable*. Constraints that are both anti-monotone and succinct can be pushed completely in frequent itemsets enumeration before it starts [29]. One possible solution for certain constraints that are neither anti-monotone nor monotone is to convert them into anti-monotone or monotone ones [34], or to loosen the constraints [10], or split them into primitive-based ones [15,37,44]. If a constraint is neither anti-monotone nor monotone, cannot be converted to anti-monotone or monotone constraint, and cannot be split into primitive-based ones, we call it a *tough* constraint.

Unlike the above constraints, the general form of pattern measure constraints introduced in this paper, such as utility constraints [27,46] and occupancy constraints [39], are tough constraints, which makes the problem of efficient bound estimation of pattern measures more challenging. In previous studies, the authors usually proposed individual bound estimation methods for these constraints. In [18], Gade et al. also proposed a tough constraint named a *block constraint* and introduced a class of pruning methods with the help of the upper bounds. In [27], Liu et al. proposed the upper bound for the constraint on utility. In [39], Tang et al. proposed the upper bound for the constraint on occupancy without item weights. This study in fact unified these individual studies into a general framework. It revealed the basic idea of developing efficient bound estimation methods for all these kinds of pattern measure, presented the properties for reducing the computing complexity in this estimation, and addressed the tradeoff between the bound tightness and computing efficiency.

9. Conclusions

In this paper, we proposed a generic and unified framework for efficient and effective bound estimation of pattern measures. This framework demonstrates the basic idea for developing a bound estimation method for any pattern measure. We also presented the properties for reducing the computing complexity in this estimation, and discussed the tradeoff between the bound tightness and computational efficiency. We applied this framework to two typical pattern measures, i.e., *utility* and *occupancy*, to show its effectiveness. Additionally, we extended the traditional SQL-like pattern measures, such as *min, max, avg*, and *var* to their corresponding *relative pattern measures*, and we also explained the application of our proposed techniques to these pattern measures. Our extensive experimental results on real and large synthetic datasets demonstrate the improvements in efficiency provided by the proposed techniques. We hope these encouraging results will lead to many future studies.

Acknowledgment

Lei Zhang was supported by the Natura l Science Foundation of China (Grant no. 61502001), the Academic and Technology Leader Imported Project of Anhui University (no. J10117700050) and the Information Assurance technology Collaborative Innovation Center (no. y01008409). Ping Luo was supported by the National Natural Science Foundation of China (no. 61473274) and National High-tech R&D Program of China (863 Program) (no. 2014AA015105). Enhong Chen was supported the National Science Foundation for Distinguished Young Scholars of China (Grant No. 61325010).

References

[1] R.C. Agarwal, C.C. Aggarwal, V.V.V. Prasad, A tree projection algorithm for generation of frequent itemsets, J. Parallel Distrib. Comput. 61 (2000) 350–371.
[2] R. Agrawal, T. Imielinski, A.N. Swami, Mining association rules between sets of items in large databases, in: Proceedings of International Conference on Management of Data, 1993, pp. 207–216.
[3] R. Agrawal, R. Srikant, Fast algorithms for mining association rules in large databases, in: Proceedings of International Conference on Very Large Data Bases, 1994, pp. 487–499.
[4] K. Amphawan, P. Lenca, Mining top-k frequent-regular closed patterns, Expert Syst. Appl. 42 (2015) 7882–7894.
[5] R.J. Bayardo, Efficiently mining long patterns from databases, in: Proceedings of International Conference on Management of Data, 1998, pp. 85–93.
[6] R.J. Bayardo Jr., R.J. Bayardo, R. Agrawal, R. Agrawal, D. Gunopulos, D. Gunopulos, Constraint-based rule mining in large dense databases, in: Proceedings of International Conference on Data Engineering, 1999, pp. 188–197.
[7] F. Bonchi, F. Giannotti, A. Mazzanti, D. Pedreschi, Adaptive constraints pushing in frequent pattern mining, in: Proceedings of European Conference on Principles and Practice of Knowledge Discovery in Databases, 2003.
[8] F. Bonchi, F. Giannotti, A. Mazzanti, D. Pedreschi, Examiner: optimized level-wise frequent pattern mining with monotone constraints, in: Proceedings of IEEE International Conference on Data Mining, 2003.

[9] F. Bonchi, F. Giannotti, A. Mazzanti, D. Pedreschi, Exante: anticipated data reduction in constrained pattern mining, in: Proceedings of European Conference on Principles and Practice of Knowledge Discovery in Databases, 2003.
[10] F. Bonchi, C. Lucchese, Pushing tougher constraints in frequent pattern mining, in: Proceedings of The Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2005, pp. 114–124.
[11] F. Bonchia, F. Giannottib, C. Luccheseb, S. Orlandoc, R. Peregob, R. Trasartib, A constraint-based querying system for exploratory pattern discovery, Inf. Syst. 34 (1) (2009) 3–27.
[12] C. Bucila, J. Gehrke, D. Kifer, W. White, Dualminer: a dual-pruning algorithm for item sets with constraints, in: Proceedings of International Conference On Knowledge Discovery and Data Mining, 2003, pp. 42–51.
[13] D. Burdick, M. Calimlim, J. Gehrke, Mafia: a maximal frequent item set algorithm for transactional databases, in: Proceedings of International Conference on Data Engineering, 2001, pp. 443–452.
[14] L. Cagliero, S. Chiusano, P. Garza, G. Bruno, Pattern set mining with schema-based constraint, Knowl. Based Syst. 84 (2015) 224–238.
[15] L. Cerf, J. Besson, C. Robardet, J.-F. Boulicaut, Closed patterns meet n-ary relations, ACM Trans. Knowl. Discov. Data 3 (1) (2009).
[16] E. Chen, H. Cao, Q. Li, T. Qian, Efficient strategies for tough aggregate constraint based sequential pattern mining, Inf. Sci. 178 (2008) 1498–1518.
[17] Y. Chi, H. Wang, P.S. Yu, R.R. Muntz, Moment: maintaining closed frequent itemsets over a stream sliding window, in: Proceedings of IEEE International Conference on Data Mining, 2004, pp. 59–66.
[18] K. Gade, J. Wang, G. Karypis, Efficient closed pattern mining in the presence of tough block constraints, in: Proceedings of International Conference On Knowledge Discovery and Data Mining, 2004, pp. 138–147.
[19] F. Geerts, B. Goethals, T. Mieliainen, Tilling databases, Discov. Sci. 3245 (2004) 278–289.
[20] K. Gouda, M.J. Zaki, Genmax: an efficient algorithm for mining maximal frequent itemsets, Data Min. Knowl. Discov. 11 (3) (2005) 223–242.
[21] T. Guns, S. Nijssen, L.D. Raedt, k-pattern set mining under constraints, IEEE Trans. Knowl. Data Eng. 25 (2) (2013) 402–418.
[22] F. Hadzica, M. Heckera, A. Tagarellic, Ordered subtree mining via transactional mapping using a structure-preserving tree database schema, Inf. Sci. 310 (2015) 97–117.
[23] J. Han, H. Cheng, D. Xin, X. Yan, Frequent pattern mining: current status and future directions, Data Min. Knowl. Discov. 15 (1) (2007) 55–86.
[24] J. Han, P. Jian, Y. Yin, Mining frequent patterns without candidate generation, in: Proceedings of International Conference on Management of Data, 2000, pp. 1–12.
[25] C.K.-S. Leung, L.V.S. Lakshmanan, R.T. Ng, Exploiting succinct constraints using fp-trees, SIGKDD Explor. Newsl. 4 (1) (2002) 40–49.
[26] G. Liu, H. Lu, W. Lou, J.X. Yu, On computing storing and querying frequent patterns, in: Proceedings of International Conference On Knowledge Discovery and Data Mining, 2003, pp. 607–612.
[27] M. Liu, J. Qu, Mining high utility itemsets without candidate generation, in: Proceedings of International Conference on Information and Knowledge Management, ACM, 2012, pp. 55–64.
[28] Y. Liu, Y. Zhao, L. Chen, J. Pei, J. Han, Mining frequent trajectory patterns for activity monitoring using radio frequency tag arrays, IEEE Trans. Parallel Distrib. Syst. 23 (11) (2012).
[29] R.T. Ng, L.V.S. Lakshmanan, J. Han, A. Pang, Exploratory mining and pruning optimizations of constrained associations rules, in: Proceedings of International Conference on Management of Data, 1998, pp. 13–24.
[30] D. Nguyen, L.T. Nguyen, B. Vo, T.-P. Hong, A novel method for constrained class association rule mining, Inf. Sci. 320 (2015) 107–125.
[31] J.S. Park, M.-S. Chen, P.S. Yu, An effective hash-based algorithm for mining association rules, in: Proceedings of International Conference on Management of Data, 1995, pp. 175–186.
[32] J. Pei, J. Han, Can we push more constraints into frequent pattern mining, in: Proceedings of International Conference On Knowledge Discovery and Data Mining, 2000, pp. 350–354.
[33] J. Pei, J. Han, Constrained frequent pattern mining: a pattern-growth view, SIGKDD Explor. Newsl. 4 (1) (2002) 31–39.
[34] J. Pei, J. Han, L. Lakshmanan, Mining frequent itemsets with convertible constraints, in: Proceedings of International Conference On Knowledge Discovery and Data Mining, 2001, pp. 433–442.
[35] L.D. Raedt, A. Zimmermann, Constraint-based pattern set mining, in: Proceedings of SIAM International Conference on Data Mining, 2007.
[36] A. Silva, C. Antunes, Constrained pattern mining in the new era, Data Min. Knowl. Discov. (2015) 1–28.
[37] A. Soulet, B. Crémilleux, Mining constraint-based patterns using automatic relaxation, Intell. Data Anal. 13 (1) (2009).
[38] E. Spyropoulou, T.D. Bie, M. Boley, Interesting pattern mining in multi-relational data, Data Min. Knowl. Discov. J. 28 (2014) 808–849.
[39] L. Tang, L. Zhang, P. Luo, M. Wang, Incorporating occupancy into frequent pattern mining for high quality pattern recommendation, in: Proceedings of International Conference on Information and Knowledge Management, 2012, pp. 75–84.
[40] V.S. Tseng, B.-E. Shie, C.-W. Wu, P.S. Yu, Efficient algorithms for mining high utility itemsets from transactional databases, IEEE Trans. Knowl. Data Eng. 25 (8) (2012) 1772–1786.
[41] R. Vimieiro, P. Moscato, Disclosed: an efficient depth-first, top-down algorithm for mining disjunctive closed itemsets in high-dimensional data, Inf. Sci. 280 (0) (2014) 171–187.
[42] J. Wang, J. Han, Bide: efficient mining of frequent closed sequences, in: Proceedings of International Conference on Data Engineering, 2004, pp. 79–90.
[43] J. Wang, J. Han, J. Pei, Closet+: searching for the best strategies for mining frequent closed itemsets, in: Proceedings of International Conference On Knowledge Discovery and Data Mining, 2003.
[44] K. Wang, Y. Jiang, J.X. Yu, G. Dong, J. Han, Divide-and-approximate: a novel constraint push strategy for iceberg cube mining, IEEE Trans. Knowl. Data Eng. 17 (3) (2005) 354–368.
[45] H. Xiong, P.-N. Tan, V. Kumar, Hyperclique pattern discovery, Data Min. Knowl. Discov. J. 13 (2006) 219–242.
[46] H. Yao, H.J. Hamilton, C.J. Butz, A foundational approach to mining itemset utilities from databases, in: Proceedings of SIAM International Conference on Data Mining, 2004, pp. 482–486.
[47] U. Yun, Efficient mining of weighted interesting patterns with a strong weight and/or support affinity, Inf. Sci. 177 (17) (2007).
[48] M.J. Zaki, C. jui Hsiao, Charm: an efficient algorithm for closed itemset mining, in: Proceedings of SIAM International Conference on Data Mining, 2002, pp. 457–473.
[49] F. Zhu, X. Yan, J. Han, P.S. Yu, gprune: a constraint pushing framework for graph pattern mining, in: Proceedings of The Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2007, pp. 388–400.
[50] F. Zhu, Z. Zhang, Q. Qu, A direct mining approach to efficient constrained graph pattern discovery, in: Proceedings of International Conference on Management of Data, 2013, pp. 821–832.
[51] M. Zihayat, A. An, Mining top-k high utility patterns over data streams, Inf. Sci. 285 (2014) 138–161.