J Intell Inf Syst
DOI 10.1007/s10844-014-0314-7

Efficient discovery of correlated patterns using multiple minimum all-confidence thresholds

Uday Kiran Rage · Masaru Kitsuregawa

Received: 30 July 2013 / Revised: 3 December 2013 / Accepted: 17 February 2014

Abstract Correlated patterns are an important class of regularities that exist in a database. Although there exists no universally acceptable best measure to judge the interestingness of a pattern, *all-confidence* is emerging as a popular measure to discover the patterns. It is because the measure satisfies both the anti-monotonic and null-invariance properties. The former property makes the pattern mining practicable in real-world applications. The latter property facilitates the user to discover the patterns involving both frequent and rare items without generating the huge number of patterns. In this paper, we show that though the measure satisfies the null-invariance property, mining the patterns containing both frequent and rare items with a single minimum all-confidence ($minAllConf$) threshold leads to the dilemma known as "rare item problem." At a high $minAllConf$, the discovered correlated patterns involving rare items have very short length. At a low $minAllConf$, combinatorial explosion can occur, producing too many patterns. To confront the problem, the paper introduces an alternative model based on the concept of multiple $minAllConf$ thresholds. The proposed model generalizes the existing model of correlated patterns and facilitates the user to specify a different $minAllConf$ for each pattern depending upon its items' frequencies. A pattern-growth algorithm, called GCoMine, has also been proposed to discover the patterns. Experiment results show that GCoMine is efficient, and the proposed model can address the problem effectively.

Keywords Data mining · Knowledge discovery in databases · Correlated patterns · Rare item problem

U. K. Rage (✉)
Institute of Industrial Science, University of Tokyo, Tokyo, Japan
e-mail: uday.rage@gmail.com

M. Kitsuregawa
Institute of Industrial Science, University of Tokyo, Tokyo, Japan
e-mail: kitsure@tkl.iis.u-tokyo.ac.jp

M. Kitsuregawa
National Institute of Informatics, Tokyo, Japan

1 Introduction

Finding frequent patterns (or itemsets) from transactional databases has been widely studied in data mining (Agrawal et al. 1993; Agrawal and Srikanth 1994; Han et al. 2004, 2007). A major obstacle encountered while mining the patterns in real-world non-uniform databases is as follows: *the pattern mining involves exponential mining space and often generates a huge number of patterns*. To confront the problem, researchers have introduced correlated pattern mining (Brin et al. 1997). It is because the users can be interested in not only the frequent occurrences of sets of items, but also their possible strong correlations implied by such co-occurrences. The usefulness of correlated patterns was demonstrated in many different application domains such as climate studies (Storch and Zwiers 2002), public health (Cohen et al. 2002) and bioinformatics (Kuo et al. 2002; Xiong et al. 2005, 2006).

Numerous measures have been introduced in the literature to assess the interestingness of a pattern. Examples include *all-confidence*, *any-confidence*, *lift*, χ^2, *bond*, *h-confidence* and *relative support* (Brin et al. 1997; Xiong et al. 2006; Omiecinski 2003; Yun et al. 2003). Each measure has a selection bias that justifies the significance of a knowledge pattern. As a result, there exists no universally acceptable best measure to judge the interestingness of a pattern for any given database or application. Researchers are making efforts to suggest a right measure depending upon the user and/or application requirements (Tan et al. 2002; Wu et al. 2010; Surana et al. 2010).

Recently, *all-confidence* is emerging as a popular measure to discover the patterns (Lee et al. 2003; Kim et al. 2004, 2011; Zhou et al. 2006a, b). It is because the measure satisfies both the *anti-monotonic* (see Definition 1) and *null-invariance* (see Definition 2) properties. The former property says that "all non-empty subsets of a correlated pattern must also be correlated." This property plays a key role in reducing the search space, which in turn decreases the computational cost of mining the patterns. In other words, this property makes the pattern mining practicable in real-world applications. The latter property discloses genuine correlation relationships without being influenced by the object co-absence in a database. In other words, this property facilitates the user to discover interesting patterns involving both frequent and rare items without generating a huge number of uninteresting patterns.

Definition 1 (**Anti-monotonic property** (Agrawal and Srikanth 1994)). A measure C is anti-monotone if and only if whenever a pattern (or an itemset) X violates C, so does any superset of X.

Definition 2 (**Null-invariance property** (Tan et al. 2002)). Let us consider a 2×2 contingency table (shown in Table 1) as a contingency matrix, $M = [f_{11} f_{10}; f_{01} f_{00}]$. Let an interestingness measure is a matrix operator, O, that maps the matrix M into a scalar value, k, i.e., $OM = k$. A binary measure of association is null-invariant if $O(M + C) = O(M)$, where $C = [00; 0k]$ and k is a positive constant.

The model of correlated patterns is as follows (Lee et al. 2003). Let $I = \{i_1, i_2, \cdots, i_n\}$ be a set of items, and DB be a database that consists of a set of transactions. Each transaction T contains a set of items such that $T \subseteq I$. Each transaction is associated with an identifier, called TID. Let $X \subseteq I$ be a set of items, referred as an itemset or a *pattern*. A pattern that contains k items is a k-pattern. A transaction T is said to contain X if and only if $X \subseteq T$. The *support* of pattern X, denoted as $S(X)$, represents the number of transactions containing X in DB. The pattern X is said to be **frequent** if $S(X) \geq minSup$, where $minSup$ is

Table 1 A 2 × 2 contingency table for variables A and B

	B	$\overline{B}$	
A	f_{11}	f_{10}	f_{1+}
$\overline{A}$	f_{01}	f_{00}	f_{0+}
	f_{+1}	f_{+0}	N

the user-defined minimum support threshold. The *all-confidence* of a pattern X, denoted as $all\text{-}conf(X)$, can be expressed as the ratio of its support to the maximum support of an item within itself. That is, $all\text{-}conf(X) = \dfrac{S(X)}{max(S(i_j)|\forall\, i_j \in X)}$.

Definition 3 (The correlated pattern X**).** The pattern X is said to be ***all-confident*** or ***associated*** or ***correlated*** if

$$\begin{aligned} S(X) &\geq minSup \\ &and \\ all\text{-}conf(X) &\geq minAllConf \end{aligned} \tag{1}$$

where, *minAllConf* is the user-defined *minimum all-confidence* threshold value.

The *support* of a pattern can also be expressed in percentage of $|DB|$, where $|DB|$ represents the total number of transactions in DB. In this paper, we use the former definition of *support* for ease of explanation. Example 1 illustrates the model using the transactional database shown in Table 2.

Example 1 The transactional database shown in Table 2 has 20 transactions. The set of items, $I = \{a, b, c, d, e, f, g, h\}$. The set of items '$a$' and '$b$', i.e., $\{a, b\}$ is a pattern. It is a 2-pattern. For simplicity, we write this pattern as 'ab'. The pattern occurs in 8 transactions ($tids$ of 1, 2, 7, 10, 11, 13, 16 and 19). Therefore, the support of 'ab,' i.e., $S(ab) = 8$. If the user-specified $minSup = 3$, then 'ab' is a frequent pattern because $S(ab) \geq minSup$. The *all-confidence* of 'ab,' i.e., $all\text{-}conf(ab) = \frac{8}{max(11,9)} = 0.72$. If the user-specified $minAllConf = 0.63$, then 'ab' is a correlated pattern because $S(ab) \geq minSup$ and $all\text{-}conf(ab) \geq minAllConf$.

Lee et al. (2003) have proposed a pattern-growth algorithm, called CoMine, to discover the correlated patterns. In Kiran and Kitsuregawa (2012), we have proposed an improved CoMine algorithm, called CoMine++. While testing the performance of CoMine++ on different types of datasets, we have made the following key observation: *though the measure satisfies the null-invariance property, the usage of a single* $minAllConf$ *threshold confines*

Table 2 Transactional database

TID	ITEMS	TID	ITEMS	TID	ITEMS	TID	ITEMS	TID	ITEMS
1	a, b	5	c, d	9	c, d, g	13	a, b, e	17	c, d
2	a, b, e	6	a, c	10	a, b	14	b, e, f, g	18	a, c
3	c, d	7	a, b	11	a, b	15	c, d	19	a, b, h
4	e, f	8	e, f	12	a, c, f	16	a, b	20	c, d, f

the discovery of correlated patterns involving both frequent and rare items to the databases in which the frequencies of items do not vary widely. If the items' frequencies in the database vary in a great deal, then we encounter the following problems:

- At a high $minAllConf$ threshold, we discover the correlated patterns pertaining to both frequent and rare items. However, most of the discovered patterns containing rare items tend to have very short length. We have observed that most of them were singleton patterns as they have all-confidence of 1.
- In order to discover the long correlated patterns containing rare items, we have to set a low $minAllConf$ threshold. However, this may cause combinatorial explosion, producing too many patterns, because those frequent items will be associated with one another in all possible ways and many of them can be meaningless depending upon the user and/or application requirements.

We call this dilemma as the *rare item problem* (Weiss 2004). This problem is further discussed in Section 3. Please note that considering an item as either frequent or rare is a subjective issue that depends upon the user and/or application requirements.

One cannot ignore the knowledge pertaining to rare items. It is because such knowledge has been found useful in many real-world applications, such as detecting oil spills in satellite images (Kubat et al. 1998) and improving the performance of recommender systems (Gedikli and Jannach 2010). Example 2 illustrates the significance of knowledge pertaining to rare items.

Example 2 In a supermarket, the set of items 'bed' and 'pillow' are relatively less frequently purchased than the set of items 'bread' and 'jam'. However, the managers of a supermarket are more interested in the correlation relationship between the former set of items. It is because the former set of items generate more revenue per unit.

With this motivation, we make an effort to discover correlated patterns involving both frequent and rare items effectively. The contributions of this paper are as follows:

i. We theoretically analyze the pattern model and introduce the concepts "items' support intervals" and "cutoff-item-support." Next, we describe the *rare item problem* using these two concepts.
ii. To confront the problem, we extend the existing correlated pattern model to allow the user to specify multiple minimum all-confidence thresholds to reflect frequencies of items. In particular, the user can specify a different $minAllConf$-like threshold for each item known as *minimum item all-confidence* ($MIAC$). Thus, each pattern may satisfy a different $minAllConf$ threshold depending on its items' frequencies. This new model enables us to achieve our objective of producing long correlated patterns containing rare items without causing combinatorial explosion.
iii. A pattern-growth algorithm, called Generalized CoMine (GCoMine), has also been proposed to discover the patterns. The GCoMine uses the prior knowledge regarding the items' $MIAC$ values and discovers the complete set of correlated patterns with a single scan on the database.
iv. By conducting experiments on different datasets, we show that the proposed model can discover interesting patterns involving both frequent and rare items without generating a huge number of meaningless correlated patterns. In addition, we also show that the GCoMine is runtime efficient and scalable as well.

The rest of the paper is organized as follows. Section 2 describes the related work on correlated pattern mining. Section 3 describes the rare item problem while mining the patterns with a single $minAllConf$ threshold. Section 4 introduces the proposed model. Section 5 describes the GCoMine to discover the patterns. The experimental results on both synthetic and real-world datasets have been reported in Section 7. Finally, Section 6 concludes the paper with future research directions.

In Kiran and Kitsuregawa (2013), we have shown that mining correlated patterns with a single $minAllConf$ threshold leads to the rare item problem, and proposed a model to address same. In this paper, we have introduced GCoMine algorithm to discover the complete set of correlate patterns with a single scan on the database. Furthermore, we have strengthen the paper with extensive experiments on both synthetic and real-world datasets.

2 Related work

In the literature, the *rare item problem* was first identified while mining frequent patterns with a single $minSup$ threshold (Weiss 2004). It is as follows:

i. If $minSup$ is set too high, we will miss those patterns that involve rare items.
ii. In order to find the frequent patterns that involve both frequent and rare items, we have to set low $minSup$. However, this may cause combinatorial explosion, producing too many frequent patterns.

To confront the problem, researchers have made efforts to mine frequent patterns using multiple $minSups$ framework (Liu et al. 1999; Kiran and Reddy 2011). In this framework, each item in the database is specified with a $minSup$-like constraint known as *minimum item support* (MIS). The $minSup$ for a pattern is represented with the minimal MIS value among all its items. Thus. each pattern can satisfy a different $minSup$ depending upon its items' frequencies. In this paper, we have employed similar approach to address the problem in correlated pattern mining.

Brin et al. (1997) have introduced correlated pattern mining using $lift$ and χ^2 as the interestingness measures. Lee et al. (2003) have shown that correlated patterns can be effectively discovered with *all-confidence* measure as it satisfies both *null-invariance* and *anti-monotonic properties*. An FP-growth-like algorithm, called CoMine, was also proposed to discover the correlated patterns. An improved CoMine, known as CoMine++, has also been proposed to discover the patterns (Kiran and Kitsuregawa 2013). Kim et al. (2004) have used all-confidence to discover confidence-closed correlated patterns. In addition, a pattern-growth algorithm known as CCMine has been discussed to discover the patterns. Zhou et al. (2006a) have employed all-confidence to mine mutually and positively correlated patterns. All these approaches employ only a single $minAllConf$ threshold to discover the patterns. As a result, these approaches also suffer from the *rare item problem*.

Xiong et al. (2005, 2006) have introduced $h\text{-}confidence$ measure to discover "Hyper-clique patterns." These patterns are groups of objects which are strongly related to each other. Theoretically, the $h\text{-}confidence$ is same as the *all-confidence* (Xiong et al. 2006). As a result, mining these patterns with a single minimum *h-confidence* value leads to the *rare item problem*.

Since the proposed work focusses on addressing the *rare item problem* in the basic model of correlated patterns, it can be observed that our work provides a lot of scope for the data mining researchers to investigate the extensions of our work to address the problem in all of the above approaches.

Some of the null-invariance measures, such as Cosine, do not satisfy the the anti-monotonic property. This increases the search space, which in turn increases the computational cost of mining the patterns. Kim et al. (2011) have introduced two pruning techniques to reduce the computational cost of mining the patterns. An Apriori-like algorithm, called NICOMINER, has been proposed to discover the patterns. As the *all-confidence* measure satisfies the null-invariance property, NICOMINER can be employed to discover the patterns with our model. However, it is not efficient because the algorithm suffers from the performance problems, which involves the generation of huge number of candidate patterns and multiple scans on the dataset.

3 The rare item problem in correlated pattern mining

In this section, we theoretically analyze the basic model of correlated patterns and introduce the concepts "items' support intervals" and "cutoff-item-support." These two concepts facilitate us in describing the problem effectively.

3.1 Items' support intervals

Definition 4 Items' Support Intervals: It says that an item $i_j \in I$ can generate correlated patterns of higher order by combining with only those items that have support within a specific interval. From the Definition 3, it turns out that an item $i_j \in I$ can generate correlated patterns of higher-order by combining with only those items that have support within the interval

$$\left[max \begin{pmatrix} S(i_j) \times minAllConf, \\ minSup \end{pmatrix}, max \begin{pmatrix} \dfrac{S(i_j)}{minAllConf}, \\ minSup \end{pmatrix} \right].$$

The correctness is shown in Theorem 1, and is based on Properties 1 and 2 and Lemmas 1 and 2. Example 3 illustrates the concept of items' support intervals.

Example 3 The support of item 'f' in Table 2 is 5. If $minAllConf = 0.63$ and $minSup = 3$, then 'f' can generate correlated patterns by combining with only those items that have support in the range of $[3,\ 8]$ $(= [max(5 \times 0.63, 3), max(\frac{5}{0.63}, 3)])$. In other words, the item 'f' may generate correlated patterns of higher-order by combining with only the items 'e' and 'd'. It is because only these items have support within the interval $[3, 8]$ in Table 2.

Property 1 If X and Y are two patterns such that $X \subset Y$, then $S(X) \geq S(Y)$.

Property 2 The maximum support a pattern X can have is $min(S(i_j)|\forall i_j \in X)$. Therefore, the maximum all-confidence a pattern X can have is $\dfrac{min(S(i_j)|\forall i_j \in X)}{max(S(i_j)|\forall i_j \in X)}$.

Lemma 1 *Let $minAllConf$ be the user-defined minimum all-confidence threshold value. The lower limit of a support interval for an item $i_q \in I$ having support $S(i_q)$ is $S(i_q) \times minAllConf$.*

Proof Let 'i_p' and 'i_q' be the two items (or 1-patterns) having supports such that $S(i_p) < S(i_q)$. The maximum all-confidence the pattern '$i_p i_q$' (=$\{i_p, i_q\}$) can have is $\frac{S(i_p)}{S(i_q)}$ (Property 2). If $S(i_q) \times minAllConf > S(i_p)$, then $minAllConf > \frac{S(i_p)}{S(i_q)}$. If $S(i_q) \times minAllConf = S(i_p)$, then $minAllConf = \frac{S(i_p)}{S(i_q)}$ $\left(= \textit{all-conf}(i_p i_q)\right)$. Therefore, the lower limit of a support interval for an item $i_q \in I$ with support $S(i_q)$ is $S(i_q) \times minAllConf$. □

Lemma 2 *Let $minAllConf$ be the user-defined minimum all-confidence threshold value. The upper limit of a support interval for an item $i_q \in I$ having support $S(i_q)$ is* $\dfrac{S(i_q)}{minAllConf}$.

Proof Let 'i_q' and 'i_r' be the two items (or 1-patterns) having supports such that $S(i_q) < S(i_r)$. Using Property 2, the maximum all-confidence for the pattern '$i_q i_r$' (=$\{i_q, i_r\}$) can be $\frac{S(i_q)}{S(i_r)}$. If $S(i_r) > \dfrac{S(i_q)}{minAllConf}$, then $minAllConf > \dfrac{S(i_q)}{S(i_r)}$. Therefore, the upper limit of a support interval for an item $i_q \in I$ is $\dfrac{S(i_q)}{minAllConf}$. □

Theorem 1 *For the user-defined $minSup$ and $minAllConf$ thresholds, the support interval of an item $i_q \in I$ with support $S(i_q)$ is*

$$\left[max \left(\begin{array}{c} S(i_q) \times minAllConf, \\ minSup \end{array} \right), \; max \left(\begin{array}{c} \dfrac{S(i_q)}{minAllConf}, \\ minSup \end{array} \right) \right].$$

Proof The support interval for an item $i_q \in I$ for *support* measure is $[minSup, \infty)$. In other words, items having support no less than $minSup$ can combine with one another in all possible ways to generate frequent patterns of higher-order. Using Lemmas 1 and 2, the support interval of an item $i_q \in I$ for *all-confidence* measure is $\left[S(i_q) \times minAllConf, \; \dfrac{S(i_q)}{minAllConf} \right]$. Therefore, the support interval for an item $i_q \in I$ for both *support* and *all-confidence* measures is

$$\left[max \left(\begin{array}{c} S(i_q) \times minAllConf, \\ minSup \end{array} \right), \; max \left(\begin{array}{c} \dfrac{S(i_q)}{minAllConf}, \\ minSup \end{array} \right) \right].$$

□

3.2 Cutoff-item-support

Xiong et al. (2006) have introduced "the cross-support property" (see Property 3) while mining the hyper-clique patterns with the *h-confidence* measure.

Property 3 [Generalized Cross-Support Property]: Given a measure M, for any cross-support pattern P with respect to a threshold t, if there exists a monotone increasing function f such that $M(P) < f(t)$, then the measure M has the cross-support property.

We slightly modify this property, to handle the specific case of mining correlated patterns using both *support* and *all-confidence* measures. From the definition of correlated pattern (see (1)), it turns out that if $X = \{i_1, i_2, \cdots, i_k\}$, $1 \leq k \leq n$, is a correlated pattern, then

$$S(X) \geq max(max(S(i_j)|\forall i_j \in X) \times minAllConf, minSup) \tag{2}$$

$$= S(X) \geq max \begin{pmatrix} max(S(i_1) \times minAllConf,\ minSup) \\ max(S(i_2) \times minAllConf,\ minSup) \\ \vdots \\ max(S(i_k) \times minAllConf,\ minSup) \end{pmatrix} \tag{3}$$

Using (3), we introduce the definition of "cutoff-item-support". It is as follows.

Definition 5 Cutoff-item-support of an item: The *cutoff-item-support* (CIS) of an item $i_j \in I$, denoted as $CIS(i_j)$, is the minimum support a pattern $X \ni i_j$ must have to be a correlated pattern. It is equal to the maximum of $minSup$ and product between its support and $minAllConf$. That is, $CIS(i_j) = max(minSup,\ S(i_j) \times minAllConf)$.

Example 4 The support of 'a' in Table 2 is 11. If the user-specified $minSup = 3$ and $minAllConf = 0.63$, then the CIS of 'a', denoted as $CIS(a) = 7$ ($\simeq max(3,\ 0.63 \times 11)$). It means any pattern containing 'a' must have support no less than 7 to be a correlated pattern. Similarly, the CIS values of 'b', 'c', 'd', 'e', 'f', 'g' and 'h' are 6, 6, 4, 3, 3, 3 and 3, respectively.

The CIS of an item plays a key role in correlated pattern mining. If the items have their CIS values very close to their supports, then the discovered patterns involving those items tend to have very short length (due to the apriori property Agrawal et al. 1993). If the items have their CIS values very far away from their supports, then it is possible to discover long patterns involving those items. However, this may cause combinatorial explosion, producing too many patterns as the items can combine with one another in all possible ways. Thus, while specifying the $minAllConf$ and $minSup$ thresholds, we have to be careful that items' CIS values are neither too close nor too far away from their respective supports. The definition of correlated patterns using items' CIS values is as follows.

Definition 6 (Correlated pattern.) A pattern X is said to be correlated if its support is no less than the maximum CIS value of all its items. That is, if X is a correlated pattern, then $S(X) \geq max(CIS(i_j)|\forall i_j \in X)$.

Since the CIS of an item $i_j \in X$, i.e., $CIS(i_j) = max(S(i_j) \times minAllConf,\ minSup)$, the correctness of the above definition is straight forward to prove from (3).

3.3 The problem

When we use a single $minAllConf$ threshold value to mine correlated patterns, then:

- The support interval width does not remain uniform for all items. Instead, it decreases from frequent to rare items (see Property 4).

Example 5 In Table 2, the item 'a' appears more frequently than the item 'd'. If the user-defined $minSup = 3$ and $minAllConf = 0.63$, then the support intervals for 'a' and 'd' are [7, 17] and [6, 14], respectively. It can be observed that the support interval width of

'a' is 10 (=17-7), while for 'f' is only 8 (=14-6). Thus, the support interval decreases from frequent to rare items.

Thus, the usage of a single $minAllConf$ threshold facilitates only the frequent items to combine with wide range of items' supports.

- The difference (or gap) between the items' support and corresponding CIS values do not remain uniform. Instead, it decreases from frequent to rare items (see Property 5).

Example 6 Continuing with Example 4, it can be observed that the difference between the $support$ and corresponding CIS values of 'a' and 'd' are 4 and 2, respectively. Thus, the gap decreases as we move from frequent to rare items.

Property 4 Let i_j and i_k, $1 \leq j \leq n$, $1 \leq k \leq n$ and $j \neq k$, be the items such that $S(i_j) \geq S(i_k)$. For the user-defined $minAllConf$ and $minSup$ thresholds, it turns out that the support interval width of i_j will be no less than the support interval width of i_k. That is,

$$max\begin{pmatrix} S(i_j) \times minAllConf, \\ minSup \end{pmatrix} - max\begin{pmatrix} \dfrac{S(i_j)}{minAllConf}, \\ minSup \end{pmatrix} \geq$$
$$max\begin{pmatrix} S(i_k) \times minAllConf, \\ minSup \end{pmatrix} - max\begin{pmatrix} \dfrac{S(i_k)}{minAllConf}, \\ minSup \end{pmatrix} \quad (4)$$

Property 5 Let i_p and i_q be the items having supports such that $S(i_p) \geq S(i_q)$. For the user-specified $minAllConf$ and $minSup$ thresholds, it turns out that $S(i_p) - CIS(i_p) \geq S(i_q) - CIS(i_q)$.

The non-uniform width of items' support intervals and the non-uniform gap between the items' support and corresponding CIS values causes the following problems while mining correlated patterns with a single $minAllConf$ threshold:

i. At a high $minAllConf$ value, rare items will have CIS values very close (or almost equivalent) to their respective supports. In addition, the support interval width for rare items will be very small allowing only rare items to combine with one another. Since it is difficult for the rare items to combine with one another and have support which is almost equivalent to their actual support, many discovered correlated patterns containing rare items tend to have a very short length. We have observed that most of them were only singleton patterns as they always have $all\text{-}conf = 1$.
ii. To discover long correlated patterns involving rare items, we must specify a low $minAllConf$ threshold value. However, a low $minAllConf$ causes frequent items to have their CIS values far away from their respective supports causing combinatorial explosion and producing too many correlated patterns.

We call this dilemma as the *rare item problem* in correlated pattern mining. In the next section, we discuss the technique to address the problem.

4 Proposed model

To address the *rare item problem*, it is necessary for the pattern model to dynamically set high $minAllConf$ threshold for a pattern containing frequent items and low $minAllConf$

threshold for a pattern containing rare items. To do so, we introduce an alternative model of mining correlated patterns using multiple minimum all-confidence thresholds. The idea is to specify $minAllConf$ threshold for each item depending upon its frequency and specify the $minAllConf$ for a pattern accordingly. We inherit this approach from the multiple $minSups$-based frequent pattern mining, where each pattern can satisfy a different $minSup$ depending upon the items' frequencies within itself (Liu et al. 1999).

In our proposed model, the definition of correlated pattern remains the same. However, the definition of $minAllConf$ is changed to address the problem. In the proposed model, each item in the database is specified with a $minAllConf$ constraint, called *minimum item all-confidence* ($MIAC$). Next, the $minAllConf$ of pattern $X = \{i_1, i_2, \cdots, i_k\}$, $1 \leq k \leq n$, is represented as maximum $MIAC$ value among all its items. That is,

$$minAllConf(X) = max \begin{pmatrix} MIAC(i_1), & MIAC(i_2), \\ \cdots, & MIAC(i_k) \end{pmatrix} \tag{5}$$

where, $MIAC(i_j)$ is the user-specified $MIAC$ threshold for an item $i_j \in X$. Thus, the definition of correlated pattern is as follows.

Definition 7 (Correlated pattern). The pattern $X = \{i_1, i_2, \cdots, i_k\}$, $1 \leq k \leq n$, is said to be correlated if

$$\begin{array}{rcl} S(X) & \geq & minSup \\ & and & \\ all\text{-}conf(X) & \geq & max \begin{pmatrix} MIAC(i_1), & MIAC(i_2), \\ \cdots, & MIAC(i_k) \end{pmatrix} \end{array} \tag{6}$$

Please note that the correctness of *all-confidence* measure will be lost if $minAllConf$ of a pattern is represented with the *minimal* $MIAC$ of all the items within itself. In particular, if the items' $MIAC$ values are converted into corresponding CIS values, then the above definition still preserves the definition of correlated pattern given in Definition 3.

Definition 8 (Problem Definition). Given the transactional database (DB), the minimum support ($minSup$) threshold and the items' minimum item all-confidence ($MIAC$) threshold values, discover the complete set of correlated patterns that satisfy the $minSup$ and maximum $MIAC$ value of all items within itself.

The proposed model facilitates the user to specify the all-confidence thresholds such that the support interval width of items can be uniform and the gap between the items' supports and corresponding CIS values can also be uniform. In addition, it also allows the user to dynamically set a high $minAllConf$ value for the patterns containing frequent items and a low $minAllConf$ value for the patterns containing only rare items. As a result, this model can efficiently address the *rare item problem*. Moreover, the proposed model generalizes the existing model of correlated patterns. If all items are specified with a same $MIAC$ value, then the proposed model is same as the basic model of correlated patterns.

The correlated patterns discovered with the proposed model satisfy the *anti-monotonic property*. That is, all non-empty subsets of a correlated pattern must also be correlated patterns. The correctness is based on Property 6 and shown in Lemma 3. In the next section, we discuss our algorithm to discover the patterns.

Property 6 If $Y \supset X$, then $minAllConf(Y) \geq minAllConf(X)$ as $max(MIAC(i_j)| \forall i_j \in Y) \geq max(MIAC(i_j)|\forall i_j \in X)$.

Lemma 3 *The correlated patterns discovered with the proposed model satisfy the anti-monotonic property.*

Proof Let X and Y be the two patterns in a transactional database such that $X \subset Y$. From the *apriori property* (Agrawal et al. 1993), it turns out that $S(X) \geq S(Y)$ and $all\text{-}conf(X) \geq all\text{-}conf(Y)$. Further, $minAllConf(Y) \geq minAllConf(X)$ (see, Property 6). Therefore, if $all\text{-}conf(Y) \geq minAllConf(Y)$, then $all\text{-}conf(X) \geq minAllConf(X)$. Hence proved. □

5 GCoMine algorithm

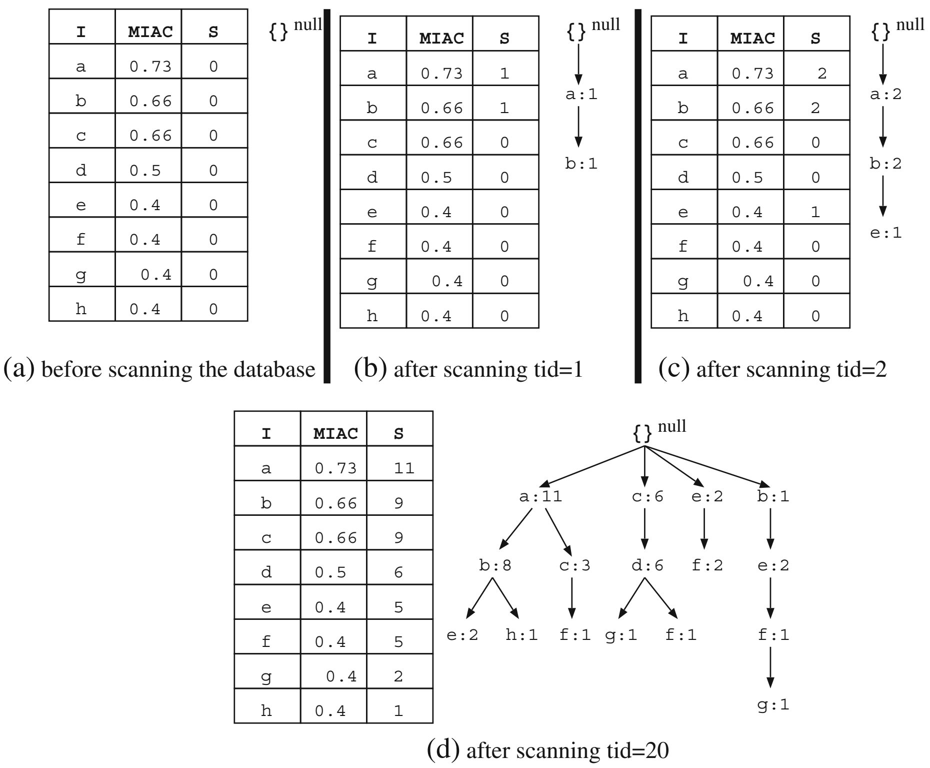
The GCoMine algorithm accepts transactional database DB, set of items I, $minSup$ and items' $MIAC$ values as the input parameters. Using the prior knowledge regarding the items' $MIAC$ values and pattern-growth technique, the algorithm discovers the complete set of correlated patterns with a single scan on the database. The steps involved in GCoMine are as follows: (i) Construction of Correlated Pattern-tree (CP-tree) and (ii) mining correlated patterns from CP-tree. We now discuss these two steps.

5.1 Construction of CP-tree

The CP-tree consists of two components: CP-list and prefix-tree. The CP-list is a list with the following fields – item-name (I), minimum item all-confidence ($MIAC$), support (S) and head of *node-links* which points to the first node in the prefix-tree carrying the item-name. The structure of prefix-tree in CP-tree is same as that of the prefix-tree in FP-tree (Han et al. 2004). That is, each node in the prefix-tree consists of three fields: item-name, count and node-link. The item-name registers which item this node represents, count registers the number of transactions represented by the portion of the path reaching this node, and node-link links to the next node in the CP-tree carrying the same item-name, or null if there is none. The key difference between these prefix-trees in CP-tree and FP-tree are as follows: the items in FP-tree are arranged in support descending order of items, while the items in CP-tree are arranged in $MIAC$ descending order of items.

The construction of CP-tree is shown in Algorithm 1. We illustrate this algorithm using the transactional database shown in Table 1. Let the user-defined $MIAC$ values for the items 'a', 'b', 'c', 'd', 'e', 'f', 'g' and 'h' be 0.73, 0.66, 0.66, 0.5, 0.4, 0.4, 0.4, 0.4, respectively. First, all items in the database are sorted in descending order of their $MIAC$ values (Step 1 in Algorithm 1). Let this sorted list of items be L. Thus, $L = \{a, b, c, d, e, f, g, h\}$. In L order, insert each item into the CP-list with corresponding $MIAC$ value and $support = 0$ (Step 2 in Algorithm 1). Next, create a root node in CP-tree and label it as "*null*." The resultant CP-list before scanning the database is shown in Fig. 1a.

A CP-tree is then updated as follows. The database DB is scanned. The items in each transaction are processed in L order, and a branch is created for each transaction as in FP-growth. Simultaneously, we increment the support values of respective items in the CP-list by 1 (Lines 4 to 8 in Algorithm 1 and Algorithm 2). For example, the scan of the first transaction, "1: a, b," which contains two items $\langle a, b$ in L order $\rangle$, leads to the construction of the first branch of the tree with nodes $\langle a : 1\rangle$ and $\langle b : 1\rangle$, where 'a' is linked as a child



(a) before scanning the database

I	MIAC	S
a	0.73	0
b	0.66	0
c	0.66	0
d	0.5	0
e	0.4	0
f	0.4	0
g	0.4	0
h	0.4	0

(b) after scanning tid=1

I	MIAC	S
a	0.73	1
b	0.66	1
c	0.66	0
d	0.5	0
e	0.4	0
f	0.4	0
g	0.4	0
h	0.4	0

(c) after scanning tid=2

I	MIAC	S
a	0.73	2
b	0.66	2
c	0.66	0
d	0.5	0
e	0.4	1
f	0.4	0
g	0.4	0
h	0.4	0

(d) after scanning tid=20

I	MIAC	S
a	0.73	11
b	0.66	9
c	0.66	9
d	0.5	6
e	0.4	5
f	0.4	5
g	0.4	2
h	0.4	1

Fig. 1 Construction of initial CP-tree

of the root and 'b' is linked as the child node of 'a'. Next, we increment support values of 'a' and 'b' in the CP-list by 1. The CP-tree generated after scanning the first transaction is shown in Fig. 1b. The second transaction containing the items 'a', 'b' and 'e' in L order will result in a branch where 'a' is linked to $root$, 'e' is linked to 'a' and 'f' is linked to 'e.' However, this branch would share a common prefix, 'a' and 'b', with the existing path for 1. Therefore, we instead increment the count of 'a' and 'b' nodes by 1, and create new node, $\langle e : 1\rangle$, where 'e' is linked to 'b'. The support of items 'a', 'b' and 'e' in CP-list are incremented by 1. The resultant CP-tree is shown in Fig. 1c. Similar process is repeated for the remaining transactions and CP-tree is updated accordingly. The resultant CP-tree after scanning every transaction in the database is shown in Fig. 1d. For the simplicity of figures, we do not show the node traversal pointers in CP-tree. However, they are maintained as in the construction process of FP-tree.

After scanning the entire database, we derive the actual supports of the items in the database. Since the correlated patterns discovered with the proposed model satisfy the anti-monotonic property, we prune infrequent items (or 1-patterns) from the CP-tree using the $minSup$ threshold (Lines 9 to 16 in Algorithm 1 and Algorithm 3). This step reduces the computational cost of mining the patterns from CP-tree. The infrequent items are the items that have support no more than $minSup$ threshold. The remaining items (or frequent items) were considered as correlated items, because, the all-confidence value of every item will be equal to 1, which is no less than their respective $MIAC$ values. Let C denote the sorted list of correlated items in CP-list in descending order of their $MIAC$ values. Figure 2 shows the

I	MIAC	S
a	0.73	11
b	0.66	9
c	0.66	9
d	0.5	6
e	0.4	5
f	0.4	5

$$\begin{tikzcd}
 & & \{\}\ \text{null} \arrow[dll] \arrow[d] \arrow[dr] \arrow[drr] & & \\
\texttt{a:11} \arrow[d] \arrow[dr] & & \texttt{c:6} \arrow[d] & \texttt{e:2} \arrow[d] & \texttt{b:1} \arrow[d] \\
\texttt{b:8} \arrow[d] & \texttt{c:3} \arrow[d] & \texttt{d:6} \arrow[d] & \texttt{f:2} & \texttt{e:2} \arrow[d] \\
\texttt{e:2} & \texttt{f:1} & \texttt{f:1} & & \texttt{f:1}
\end{tikzcd}$$

Fig. 2 Final CP-tree

final CP-tree generated pruning the infrequent items 'g' and 'h'. The constructed CP-tree preserves all the interesting correlation relationships between the sets of items in a database. The correctness of CP-tree is shown in Lemma 4.

Lemma 4 *Given a transactional database DB, the minimum support ($minSup$) and items' $MIAC$ values, the constructed CP-tree contains the complete information about all correlated patterns in DB.*

Rationale *In the CP-tree construction process, each transaction in DB is mapped to one path in the CP-tree. All correlated items information in each transaction is completely stored in the CP-tree. Since the correlated patterns satisfy the anti-monotonic property, the constructed CP-tree contains the complete information about all correlated patterns in DB.*

5.2 Mining correlated patterns from CP-tree

The procedure for mining correlated patterns from CP-tree is shown in Algorithm 4, and is as follows. Start from each correlated 1-pattern (as an initial suffix pattern), construct its conditional pattern base (a *sub-database*, which consists of the set of prefix-paths in the CP-tree co-occurring with the suffix pattern), then construct its conditional CP-tree, and perform mining recursively on such a tree. The pattern-growth is achieved by the concatenation of the suffix pattern with the correlated patterns generated from the conditional CP-tree.

Table 3 shows the correlated patterns discovered from the CP-tree shown in Fig. 2. We briefly explain mining the patterns from CP-tree. Consider the item 'f' that has lowest $MIAC$ among all items in the CP-tree. It occurs in four branches of CP-tree. The branches are $\langle a, c, f : 1\rangle$, $\langle c, d, f : 1\rangle$, $\langle e, f : 2\rangle$ and $\langle b, e, f : 1\rangle$. Consider '$f$' as a suffix pattern (or item), its conditional prefix paths are $\langle a, c : 1\rangle$, $\langle c, d : 1\rangle$, $\langle e : 2\rangle$ and $\langle b, e : 1\rangle$, which form its conditional pattern base (see Fig. 3a). The conditional CP-tree is generated with $\langle e : 3\rangle$; the items a, b, c and d are not included because the support counts are less than the $minSup$ value (see Fig. 3b). The single path generates the correlated pattern $\{e, f : 3\}$. Similar process is repeated for other remaining items in CP-tree to discover the complete set of correlated patterns. The correctness of GCoMine has been shown in Lemma 5.

Lemma 5 *Let α be a pattern in CP-tree. Let $minSup$ be the minimum support that α has to satisfy. Let B be the α-conditional pattern base, and β be an item in B. Let $sup(\beta)$ and $sup_B(\beta)$ be the support of β in the transactional database and in B, respectively. Let $MIAC(\beta)$ be the user-specified β's $MIAC$ value. If α is a correlated pattern,*

Table 3 Mining the CP-tree by creating conditional pattern bases

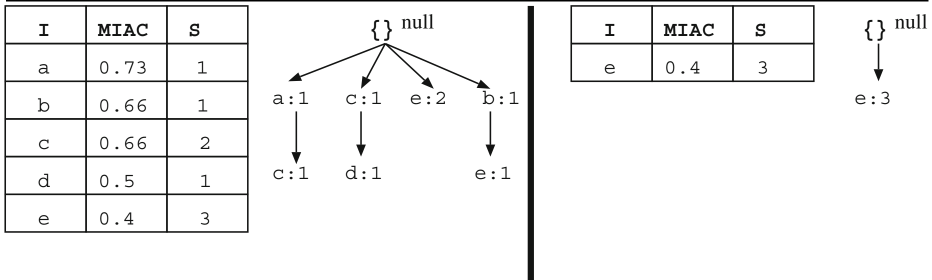
Item	Conditional pattern base	Conditional CP-tree	Correlated patterns
f	$\{\{a, c : 1\}, \{c, d : 1\}, \{e : 2\}, \{b, e : 1\}\}$	$\langle e : 3\rangle$	$\{e, f : 3\}$
e	$\{\{a, b : 2\}, \{b : 1\}\}$	–	–
d	$\{c : 6\}$	–	$\{c, d : 6\}$
c	$\{a : 3\}$	–	–
b	$\{a : 8\}$	$\langle a : 8\rangle$	$\{a, b : 8\}$

$sup_B(\beta) \geq minSup$ and $\frac{sup_B(\beta)}{sup(\beta)} \geq MIAC(\beta)$, then the pattern $< \alpha, \beta >$ is also a correlated pattern.

Proof According to the definition of conditional pattern base and compact CP-tree (or CP-tree), each subset in B occurs under the condition of the occurrence of α in the transactional database. If an item β appears in B for n times, it appears with α in n times in the database. Thus, from the definition of correlated pattern used in our model, if the $\frac{sup_B(\beta)}{sup(\beta)} \geq MIAC(\beta)$ and $sup_B(\beta) \geq minSup$, then $< \alpha, \beta >$ is a correlated pattern. Hence proved. $\square$

Algorithm 1 GCoMine (DB: transactional database, I: itemset containing n items, $minSup$: minimum support and $MIAC$: items' $MIAC$ values)

```
1: Let $L$ denote the sorted list of items in descending order of their $MIAC$ values.
2: In $L$ order, insert each item into the CP-list with their corresponding $MIAC$ values and
   $support = 0$.
3: Create the $root$ of a CP-tree, and label it as "$null$."
4: {Construction of CP-tree}
5: for each transaction $t \in DB$ do
6:     Sort all the items in $t$ in $L$ order.
7:     Let the sorted items in $t$ be $[p|P]$, where $p$ is the first element and $P$ is the
   remaining list. Call InsertTree($[p|P]$, $T$).
8: end for
9: Let $Tree$ be the CP-tree generated after scanning every transaction in the database.
10: {Pruning infrequent items from CP-tree.}
11: for $(j = n - 1; j \geq 0; - - j)$ do
12:     if $(S[i_j] < minSup)$ then
13:         Call $TreePruning(Tree, i_j)$.
14:         Delete the item $i_j$ in PF-list.
15:     end if
16: end for
17: The CP-tree is mined by calling GCoMine_patternGrowth(CP-tree, $null$).
```



I	MIAC	S
a	0.73	1
b	0.66	1
c	0.66	2
d	0.5	1
e	0.4	3

I	MIAC	S
e	0.4	3

(a) Conditional pattern base of item ‘f’ (b) Conditional CP-tree of item ‘f’

Fig. 3 The conditional pattern base of ‘f’

6 Experimental results

In this section, we first evaluate the basic and proposed models of correlated patterns. We show that proposed model allows us to find correlated patterns involving rare items without generating a huge number of meaningless correlated patterns with frequent items. Next, we evaluate the performance of GCoMine algorithm.

Algorithm 2 InsertTree($[p|P]$, T)

```
1: Increment the support of an item $p$ in CP-list by 1.
2: **if** $T$ has a child node $N$ such that p.item-name=N.item-name **then**
3:     Increment $N$'s count by 1.
4: **else**
5:     Create a new node $N$, and let is count be 1.
6:     Let its parent link be linked to $T$.
7:     Let its node-link be linked to the nodes with the same item-name via the node-link structure.
8: **end if**
9: **if** $P$ is non-empty **then**
10:    Call InsertTree($P$, $N$);
11: **end if**
```

Algorithm 3 TreePruning($Tree$, i_j)

```
1: **for** each node in the node-link of $i_j$ in $Tree$ **do**
2:     **if** the node is a leaf **then**
3:         Remove the node directly.
4:     **else**
5:         Remove the node and then link its parent node with the corresponding child node(s).
6:     **end if**
7: **end for**
```

Algorithm 4 GCoMine_patternGrowth($Tree$, α)

```
1:  for each $a_i$ in the header of $Tree$ do
2:      Generate pattern $\beta = \alpha \cup a_i$ with $all\text{-}confidence = \frac{S(\beta)}{max(S(i_j)|\forall i_j \in \beta)}$;
        {$S(\beta) = S(a_i)$ in $\alpha$-projected database.}
3:      Get a set $I_\beta$ of items to be included in $\beta$-projected database.
4:      for each $b_j \in I\beta$ do
5:          if $S(\beta \cup b_j) < minSup$ and $all\text{-}conf(\beta \cup b_j) < MIAC(b_j)$ then
6:              Delete $b_j$ from $I_\beta$. {Since CP-tree is constructed in $MIAC$ descending order
                of items, $MIAC(b_j)$ will be the maximum $MIAC$ value among all items in
                $\beta \cup b_j$.}
7:          end if
8:      end for
9:      Construct $\beta$-conditional CP-tree with items in $I_\beta$, $Tree_\beta$.
10: end for
11: if $Tree_\beta \neq \emptyset$ then
12:     Call GCoMine_patternGrowth($Tree_\beta$, $\beta$).
13: end if
```

6.1 Experimental setup

The $GCoMine$ and $CoMine$ algorithms are written in GNU C++ and run with Ubuntu 10.04 operating system on a 2.66 GHz machine with 1GB memory. The runtime specifies the total execution time, i.e., CPU and I/Os. We pursued experiments on synthetic (T10I4D100K and T10I4D1000K) and real-world (Retail (Brijs et al. 2000), Mushroom and Kosarak) datasets. The T10I4D100K and T10I4D1000K datasets are generated synthetic dataset generator discussed in Agrawal et al. (1993). The real-world datasets were downloaded from Frequent Itemset MIning (FIMI) repository (http://fimi.ua.ac.be/data/). The details of these datasets are shown in Table 4. The last column in this table shows the $minSup$ used in each of these datasets. Since 'Mushroom' is a dense dataset, we set high $minSup$ threshold, i.e., $minSup = 0.01$. In remaining datasets, we have set relatively low $minSup$, i.e., $minSup = 0.001$. For the purpose of simplicity, we do not discuss about the $minSup$ threshold used in each of these datasets. However, they are maintained as specified in Table 4.

6.2 A method to specify items' $MIAC$ values

For our experiments, we need a method to assign $MIAC$ values to items in a dataset. We use the actual frequencies (or the supports) of the items in the data as the basis to specify $MIAC$ values. Since the non-uniform difference (or gap) between the items' support and CIS values is the cause of *rare item problem*, we employ a method that specifies items' $MIAC$ values such that there exists a uniform gap between the items' $support$ and CIS values. It is as follows.

Let δ be the representative (e.g. $mean$, $median$ and $mode$) support of all items in the database. Choosing a $minAllConf$ value, calculate the gap between the δ's support and corresponding CIS values (see (7)). We call the gap as "support difference" and is denoted as SD.

$$\begin{aligned} SD &= \delta - \delta \times minAllConf \\ &= \delta \times (1 - minAllConf) \end{aligned} \quad (7)$$

Table 4 The details of the datasets

Dataset	Type	Nature	Number of items	NT	MTS	ATS	Minimum support
T10I4D100K	Synthetic	Sparse	870	100,000	29	10	0.001
T10I4D1000K	Synthetic	Sparse	30991	983026	33	10	0.001
Retail	Real-world	Sparse	16470	88162	76	10	0.001
Mushroom	Real-world	Dense	119	8124	23	23	0.01
Kosarak	Real-world	Sparse	41270	990002	2498	8	0.001

The terms ‘NT’, ‘MTS’ and ‘ATS’ respectively denote ‘Number of Transactions’, ‘Maximum Transaction Size’ and ‘Average Transaction Size’

Next, we specify $MIAC$ values for other items such that the gap (or SD) remains constant. The methodology to specify $MIAC$ values to the items $i_j \in I$ is shown in (8).

$$MIAC(i_j) = max\left(\left(1 - \frac{SD}{S(i_j)}\right),\ LMAC\right) \tag{8}$$

Where, $LMAC$ is the user-defined lowest $MIAC$ value an item can have. It is particularly necessary as $\left(1 - \frac{SD}{S(i_j)}\right)$ can give negative $MIAC$ value for an item i_j if $SD > S(i_j)$.

Figure 4a, b and c respectively shows the $MIAC$ values specified by the proposed model in *T10I4D100K*, *Retail* and *Mushroom* datasets with $minAllConf = 0.7$, $LMAC = 0.1$ and δ representing the mean of the support of all frequent items (or 1-patterns). The X-axis represents the rank of items provided in the descending order of their support values. The Y-axis represents the $MIAC$ values of items. It can be observed that the $MIAC$ values decreases as we move from frequent to less frequent (or rare) items. Thus, the proposed method specifies high $MIAC$ values for the frequent items, and relatively low $MIAC$ values for the rare items. The low $MIAC$ values for rare items facilitate them to combine with other items and generate correlated patterns. Simultaneously, the high $MIAC$ values of frequent items will prevent them from combining with one another in all possible ways and generating huge number of uninteresting patterns.

6.3 Generation of correlated patterns

Figure 5a, b and c respectively shows the number of correlated patterns generated by the basic and proposed models in *T10I4D100K*, *Retail* and *Mushroom* datasets. The $LMAC$ value is set to 0.1 in all these datasets. The δ is set to the mean of support values of all frequent items in a database. The X-axis represents the $minAllConf$ values that are varied

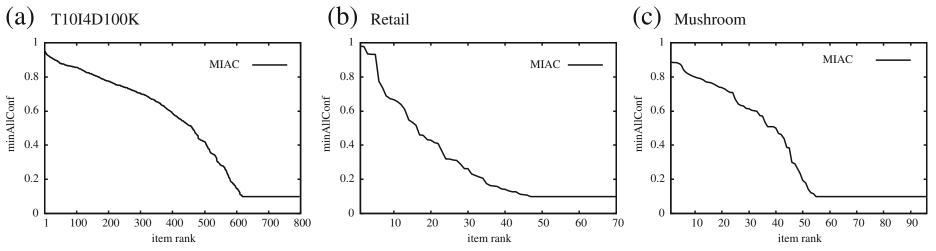


Fig. 4 The items’ $MIAC$ values in different datasets

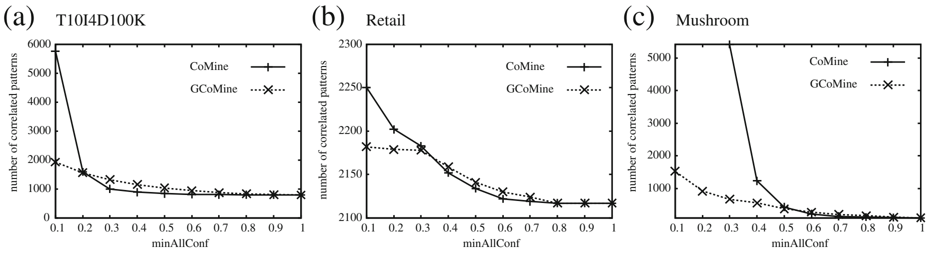


Fig. 5 The correlated patterns discovered by both the models at different *minAllConf* thresholds in different datasets

from 0.1 to 1. The Y-axis represents the number of correlated patterns discovered at different *minAllConf* values. Too many correlated patterns were discovered in both models when $minAllConf = 0$. For convenience, we are not showing the correlated patterns discovered at $minAllConf = 0$. The following observations can be drawn from these two graphs:

i. At high *minAllConf* values, the proposed model has generated more number of correlated patterns than the basic model. The reason is as follows. In the basic model, *CIS* values of rare items were very close (almost equal) to their respective supports. Since it is difficult for the rare items to combine with other (rare) items and have support as their own, very few correlated patterns pertaining to rare items have been discovered. In the proposed model, some of the rare items had their *CIS* values relatively away from their supports. This facilitated the rare items to combine with other rare items and generate correlated patterns.
ii. At low *minAllConf* values, the proposed model has generated less number of correlated patterns than the basic model. The reason is as follows. The basic model has specified low *CIS* values for the frequent items at low *minAllConf*. The low *CIS* values of the frequent items facilitated them to combine with one another in all possible ways and generate too many correlated patterns. In the proposed model, the *CIS* values for the frequent items were not very far away from their respective supports (as compared with basic model). As a result, the proposed model was able to prevent frequent items to combine with one another in all possible ways and generate too many correlated patterns.

6.4 Performance comparison of CoMine and GCoMine

Figure 6a, b and c shows the runtime taken by CoMine and GCoMine to generate correlated patterns in *T10I4D100K*, *Retail* and *Mushroom* datasets, respectively. The *LMAC* value

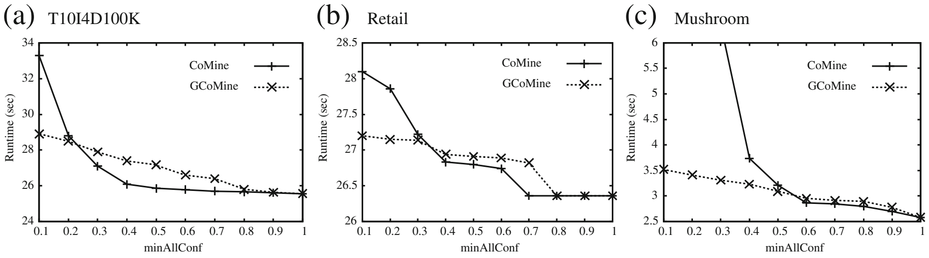


Fig. 6 The runtime comparison of GCoMine and CoMine algorithms in different datasets

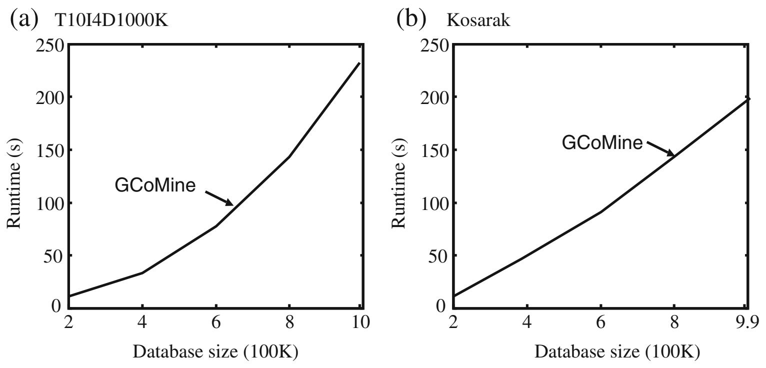


Fig. 7 Scalability test. **a** T10I4D1000k and **b** Kosarak datasets

is set to 0.1 in all these datasets. The δ is set as the average support of all those items in the database that have support no less than the $minSup$ value. The X-axis represents the $minAllConf$ values that are varied from 0.1 to 1. The Y-axis represents the runtime taken by both the algorithms to discover the correlated patterns at different $minAllConf$ values. It can be observed that the runtime of both the algorithms increases with the decrease in $minAllConf$ threshold value. It is because the runtime of both the algorithms depend upon the number of correlated patterns getting generated at a particular $minAllConf$ threshold value. More important, it is clear from this figure that GCoMine can efficiently mine the correlated patterns at large or small threshold values.

6.5 Scalability of GCoMine

We study the scalability of our GCoMine algorithm on execution time by varying the number of transactions in *T10I4D1000K* and *Kosarak* datasets. In the literature, these two datasets have been widely used to study the scalability of algorithms. The experimental setup was as follows. Each dataset was divided into five portions with 0.2 million transactions in each part. Then, we investigated the performance of GCoMine after accumulating each portion with previous parts. We fixed the $minAllConf = 0.5$, $LMAC = 0.1$, $minSup = 0.001$ and δ as the average support of all those items in the database that have support no less than the $minSup$ value.

Figure 7a and b respectively show the runtime requirements of GCoMine in *T10I4D1000K* and *Kosarak* datasets with the increase of dataset size. It is clear from the graphs that as the database size increases, overall tree construction and mining time increases. However, GCoMine shows stable performance of about linear increase of runtime consumption with respect to the database size. Therefore, it can be observed from the scalability test that GCoMine can mine the correlated patterns over large databases and distinct items with considerable amount of runtime.

7 Conclusions and future work

This paper has theoretically analyzed the basic model of correlated patterns using the concepts, “items’ support intervals” and “cutoff-item-supports”. Using these two concepts, we have shown that though the all-confidence measure satisfies the null-invariant property, mining correlated patterns with a single $minAllConf$ threshold in the databases of widely varying items' frequencies can cause dilemma known as the *rare item problem.* To confront the problem, we have proposed a generalized model of mining the patterns using the concept of multiple $minAllConf$ threshold values. A pattern-growth algorithm, called GCoMine, has been proposed to discover the patterns. Using the prior knowledge regarding the items' $MIAC$ values and pattern-growth technique, the proposed algorithm discovers the complete set of correlated patterns with a single scan on the database. The effectiveness of the new model and proposed algorithm are shown experimentally and practically.

As a part of future work, we would like to extend our work to mine correlated patterns in graph databases, sequential databases and multi-dimensional databases. We are also investigating the methodologies to specify items $MIAC$ values. It is also interesting to investigate efficient mining of closed and maximal correlated patterns with the proposed model.

References

Agrawal, R., Imieliński, T., Swami, A. (1993). Mining association rules between sets of items in large databases. In *SIGMOD* (pp. 207–216).

Agrawal, R., & Srikanth, R. (1994). Fast algorithms for mining association rules in large databases. In *VLDB* (pp. 487–499).

Brin, S., Motwani, R., Silverstein, C. (1997). Beyond market baskets: generalizing association rules to correlations. *SIGMOD Rec*, *26*, 265–276.

Brijs, T., Goethals, B., Swinnen, G., Vanhoof, K., Wets, G. (2000). A data mining framework for optimal product selection in retail supermarket data: the generalized PROFSET model. In *KDD* (pp. 300–304).

Cohen, H., West, S.G., Cohen, P., Aiken, L. (2002). *Applied multiple regression correlation analysis for the behavioral sciences*, 3rd edn. Lawrence Erlbaum Assoc Inc.

Gedikli, F., & Jannach, D. (2010). Neighborhood-restricted mining and weighted application of association rules for recommenders. In *International conference on web information system engineering*, (pp. 157–165).

Han, J., Pei, J., Yin, Y., Mao, R. (2004). Mining frequent patterns without candidate generation: a frequent-pattern tree approach. *Data Mining Knowledge Discovery*, *15*(1), 55–86.

Han, J., Cheng, H., Xin, D., Yan, X. (2007). Frequent pattern mining: current status and future directions. *Data Mining Knowledge Discovery*, *15*(1), 55–86.

Kim, W.Y., Lee, Y.K., Han, J. (2004). Ccmine: efficient mining of confidence-closed correlated patterns. In *PAKDD* (pp. 569–579).

Kim, S., Barsky, M., Han, J. (2011). Efficient mining of top correlated patterns based on null invariant measures. In *ECML PKDD* (pp. 172–192).

Kiran, R.U., & Reddy, P.K. (2011). Novel techniques to reduce search space in multiple minimum supports-based frequent pattern mining algorithms. In *EDBT* (pp. 11–20).

Kiran, R.U., & Kitsuregawa, M. (2012). Efficient discovery of correlated patterns in transactional databases using items' support intervals. In *DEXA* (pp. DEXA (1) 234–248).

Kiran, R.U., & Kitsuregawa, M. (2013). Mining correlated patterns with multiple minimum all-confidence thresholds. In *PAKDD-QIMIE* (pp. 234–248).

Kubat, M., Holte, R.C., Matwin, S. (1998). Machine learning for the detection of oil spills in satellite radar images. *Machine Learning*, *30*(2), 195–215.

Kuo, P.W., Jenssen, T.K., Butte, A.J., Onno-Machado, L., Kohane, I.S. (2002). Analysis of matched mrna measurements from two different microarray technologies. *Bioinformatics*, *18*(3), 405–412.

Lee, Y.K., Kim, W.Y., Cao, D., Han, J. (2003). CoMine: efficient mining of correlated patterns. In *ICDM* (pp. 581–584).

Liu, B., Hsu, W., Ma, Y. (1999). Mining association rules with multiple minimum supports. In *KDD* (pp. 337–341).

Omiecinski, E.R. (2003). Alternative interest measures for mining associations in databases. *IEEE Transactions on Knowledge and Data Engineering*, *15*, 57–69.

Pei, J., Han, J., Lakshmanan, L.V. (2004). Pushing convertible constraints in frequent itemset mining. *Data Mining and Knowledge Discovery*, *8*, 227–251.

Storch, H.V., & Zwiers, F.W. (2002). *Statistical analysis in climate research.* Cambridge University Press.

Surana, A., Kiran, R.U., Reddy, P.K. (2010). Selecting a right interestingness measure for rare association rules. In *COMAD* (pp. 115–124).

Tan, P.N., Kumar, V., Srivasta, J. (2002). Selecting the right interestingness measure for association patterns. In *KDD* (pp. 32–41).

Weiss, G.M. (2004). Mining with rarity: a unifying framework. *ACM SIGKDD Explorations Newsletter*, *6*(1), 7–19.

Wu, T., Chen, Y., Han, J. (2010). Re-examination of interestingness measures in pattern mining: a unified framework. *Data Mining Knolwedge Discovery*, *21*, 371–397.

Xiong, H., He, X., Ding, C.H.Q., Zhang, Y., Kumar, V., Holbrook, S.R. (2005). Identification of functional modules in protein complexes via hyperclique pattern discovery. In *Pacific symposium on biocomputing*.

Xiong, H., Tan, P.N., Kumar, V. (2006). Hyperclique pattern discovery. *Data Mining Knowledge Discovery*, *13*(2), 219–242.

Yun, H., Ha, D., Hwang, B., Ryu, K.H. (2003). Mining association rules on significant rare data using relative support. *Journal of Systems and Software*, *67*(3), 181–191.

Zhou, Z., Wu, Z., Wang, C., Feng, Y. (2006). Mining both associated and correlated patterns. *Computational Science ICCS*, *3994*, 468–475.

Zhou, Z., Wu, Z., Wang, C., Feng, Y. (2006). Efficiently mining mutually and positively correlated patterns. *Advanced Data Mining and Applications*, *4093*, 118–125.

Zheng, Z., Kohavi, R., Mason, L. (2001). Real world performance of association rule algorithms. In *KDD* (pp. 401–406).