Applied Intelligence (2021) 51:2077–2093
https://doi.org/10.1007/s10489-020-01994-9

Fast Top-K association rule mining using rule generation property pruning

Xiangyu Liu[1] · Xinzheng Niu[1] · Philippe Fournier-Viger[2]

Accepted: 1 October 2020 / Published online: 26 October 2020

Abstract
Traditional association rule mining algorithms can have a long runtime, high memory consumption, and generate a huge number of rules. Browsing through numerous rules and adjusting parameters to find just enough rules is a tedious task for users, who are often only interested in finding the strongest rules. Hence, many recent studies have focused on mining the top-k most frequent association rules that have a minimum confidence so as to limit the number of rules by ranking them by frequency. Though this redefined task has many applications, the performance of current algorithms remains an issue. To address this issue, this paper presents a novel algorithm named FTARM (Fast Top-K Association Rule Miner) to efficiently find the set of top-k association rules using a novel technique called Rule Generation Property Pruning (RGPP). This technique reduces the search space by analyzing the internal relationships between items of the database to be mined and the parameters set by users. Furthermore, a novel candidate pruning property is used by this technique to speed up the mining process. FTARM's efficiency was evaluated on various public benchmark datasets. A substantial reduction of the association rule mining time and memory usage was observed, and that FTARM has good scalability, which can benefit to many applications.

Keywords Data mining · Association rule mining · Top-k rules · Rule expansion

1 Introduction

Data mining techniques [28, 29] are applied in numerous domains to discover insightful, novel, and interesting patterns, as well as to build understandable, descriptive, and predictive models from large volumes of data. It encompasses various algorithms such as principle component analysis, clustering, sequence detection, association rule mining, and classification [40]. Association rule mining (ARM) is an unsupervised data mining task that consists of extracting interesting associations and frequent patterns from sets of items in a transaction database or other data repositories [38]. ARM is often considered as an exploratory data mining technique, as it can be used to find associations between values that can help to better understand the data.

Association rules have a wide range of applications in many fields. A well-known application of association rules is in the business field [19], where discovering associations between purchased products is very useful for decision-making and devising effective marketing strategies. In addition, some classification methods based on association rules have also been developed that achieved high classification accuracy on datasets from the UCI machine learning repository [2]. For classification, association rules may not always have the best accuracy but they are rules that are generally easily interpretable by humans. Apart from the aforementioned applications, association rule mining is also used for inventory control, analyzing protein sequences, medical diagnosis, fraud detection, monitoring telecommunication networks and many other applications [39].

To consider various factors, several variants of the association rule mining problem have been studied such

✉ Xinzheng Niu
xinzhengniu@uestc.edu.cn

Xiangyu Liu
ericliu_uestc@163.com

Philippe Fournier-Viger
philfv@hit.edu.cn

1 School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China

2 School of Humanities and Social Sciences, Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China

as fuzzy association rule mining, relational association rule mining, weighted association rule mining, temporal association rule mining and also several measures have been proposed to select interesting rules. The task of traditional association rule mining is to enumerate all rules satisfying some predefined minimum support and confidence thresholds in a given database [30]. The rule mining process is usually divided into two parts. Part one is to mine frequent patterns such as itemsets and subsequences that have a frequency that is no less than a given threshold in large transactions or relational datasets. Then, part two consists of deriving all association rules from the frequent patterns under the minimum confidence constraint.

In general, mining association rules requires that users set the corresponding minimum thresholds in advance. But, it is not intuitive for users to find appropriate values for these parameters, which have a direct impact on the number of generated association rules. The reason is that these parameters are dataset dependent. In many cases, ARM generates a large number of association rules, which cannot be understood or verified by the end users, thus limiting the usefulness of data mining results [31]. To address this problem, the TopKRules algorithm was proposed for mining the top-k association rules [14], where *k* is a parameter set by users to indicate the number of association rules to be found. Different from the traditional two-step mining of association rules, TopKRules uses rule expansions to find the top-k association rules that meet the user requirements. Although TopKRules accomplishes its mission, it explores a huge search space. To improve the performance of top-k association rule mining, the ETARM (Efficient Top-K Association Rule Miner) algorithm was proposed [34], which uses two pruning properties to reduce the search space. Experiments have shown that ETARM outperforms TopKRules. However, performance still remains an issue.

Top-k association rule mining is a more difficult problem than traditional ARM since the user does not have to set a minimum support threshold. As a consequence, the search for the top-k rules must be done starting with a minimum support threshold value of zero to ensure that all the desired rules can be found. Besides, the traditional two-step approach is not efficient for mining the top-k rules, thus it is necessary to design new pruning strategies to reduce the search space for mining rules.

In this paper, to further improve the performance of top-k association rule mining, we propose an algorithm named FTARM (Fast Top-K Association Rule Miner). It extends the ETARM algorithm with a novel search space reduction technique called rule generation property pruning (RGPP) to improve the performance. A significant advantage of FTARM is that it can raise the minimum support threshold more quickly, causing many low frequency rules to be ignored from the beginning, thus greatly reducing the search space.

The rest of the paper is organized as follows. Section 2 reviews relevant studies on mining the top-k association rules. Section 3 presents the basic definitions. Section 4 describes the proposed FTARM algorithm for mining the top-k association rules. Section 5 presents results from detailed experiments to compare the performance of the FTARM, ETARM and TopKRules algorithms. Lastly, Section 6 draws a conclusion and discusses future work.

2 Related work

Agrawal et al. introduced the concept of ARM in 1993 to identify interesting relationship between dataset entities [1]. Then, numerous researchers have joined the research on ARM to improve the performance of mining algorithms from different angles. Some people have focused on improving the performance using optimization techniques. Son et al. [38] proposed a method based on animal migration optimization to reduce the number of association rules by ignoring rules that are weakly supported by the data. In [47], Wen et al. presented a hybrid temporal ARM method with genetic algorithm to mine temporal association rules. Moslehi et al. [32] proposed a hybrid framework based on genetic particle swarm optimization algorithms to discover rules in continuous numeric datasets. Zhang et al. [49] introduced a method to mine association rules by integrating a multi-dimension and multi-objective double group discrete firefly algorithm (MODGDFA) with Pareto rules. Besides, some efficient data structures have also been used to improve the performance of ARM. For example, Anand et al. [3] proposed a mining algorithm which applies a priority model to find interesting relations in a database using the treap data structure. Hashem et al. [18] introduced a new memory efficient data structure called dynamic superset bit-vector to represent the relationship between frequent closed itemsets in a lattice, and Mai et al. [30] proposed a lattice-based algorithm for mining high utility association rules. ARM is an active research field and several other improvements have also been proposed [5, 10, 22, 25, 33, 44].

Frequent itemset mining (FIM) is a sub-problem of ARM. Its purpose is to extract any frequent itemset, that is a set of values that has an occurrence frequency that is no less than a given threshold. Since FIM was introduced [1], many approaches have been designed using different optimization strategies to improve the performance of this task. Aryabarzan et al. [6] proposed an efficient algorithm to quickly mine frequent itemsets using a data structure called NegNodeset. An efficient algorithm using hashing and lexicographic order for FIM was introduced in [7]. Djenouri et al. [13] combined a heuristic with bio-inspired algorithms for solving the FIM problem. To better handle large-scale data, Han et al. [17] introduced a novel precomputation-based FIM method to quickly identify frequent itemsets, and Zhang et al. [50] proposed an algorithm for mining high quality approximate frequent itemsets in very large databases. In addition, some parallel methods have also been proposed. Raj et al. [35] presented an efficient apriori-based frequent itemset mining algorithm on Spark, and Chon et al. [8] proposed a fast GPU-based FIM method for analyzing large-scale data. Other related works can be seen in [12, 15, 16, 48].

However, mining frequent itemsets still requires much memory and time, and setting the non intuitive minimum support parameter is an important limitation of FIM algorithms. As a solution, the task of FIM was redefined as top-k FIM [9, 43]. In recent years, the concept of top-k pattern mining has been extended to many other pattern types [23, 24, 26, 36]. To quickly complete this top-k mining task, many researchers have improved the performance using various data structures and pruning strategies. For example, Deng et al. [11] proposed a fast algorithm for mining the top-rank-k frequent itemsets using the Node-list structure. In [20], an efficient and effective algorithm using the N-list structure was proposed for mining the top-rank-k frequent patterns. Vo et al. [42] used the Weighted N-list (WN-list) structure and an early pruning strategy to mine the top-rank-k frequent weighted itemsets. And Le et al. [27] introduced a method for mining the top-k frequent patterns with effective threshold raising strategies.

The idea of mining the top-k association rules is analogous to the idea of mining the top-k itemsets, as both are applied to meet the needs of users who are only interested in the strongest rules or itemsets. As for top-k ARM, algorithms were designed to discover k-optimal rules [46] and filtered top-k association rules [45]. These algorithms were found to be very efficient, but they are unable to mine rules with more than one item in the consequent. To address this issue, the TopKRules algorithm was proposed [14]. The traditional $minsup$ parameter is replaced by a k parameter, which is more intuitive to set for users. TopKRules does not restrict the number of items contained in the antecedent or consequent of association rules. The algorithm finds the top-k most frequent association rules satisfying a minimum confidence in a transaction database using a process called rule expansion. Although TopKRules offers a substantial performance improvement compared with the traditional two-step mining algorithms, the number of candidate rules generated by rule expansions can still be huge and runtimes remain long. To solve this problem, the ETARM algorithm was recently proposed [34]. Two novel search space pruning strategies are used to reduce the number of candidate rules, which can considerably decrease the running time and memory usage.

TopKRules and ETARM have been widely used in recent years, and they each play an indispensable role in different data mining tasks and practical applications [4, 21, 33, 37, 41]. But in many cases, ETARM and TopKRules still have long runtimes. Hence, it remains an important research problem to design faster algorithms. For top-k association rule mining algorithms, how quickly the minimum support threshold is increased directly influences performance. Based on this observation, this paper presents a novel algorithm named FTARM, which extends ETARM with novel techniques. FTARM prunes the search space before and during rule expansions to speed up the mining process. A key to the success of FTARM is that it can raise the minimum support threshold to a higher level before even starting the rule expansion process.

3 Problem definition

This section is divided in two subsections, which describe the traditional problem of ARM and that of top-k ARM, respectively.

3.1 Traditional association rule mining

Let $I = \{i_1, i_2, i_3, ..., i_m\}$ be a set of items or features, and $DB = \{T_1, T_2, T_3, ..., T_n\}$ be a set of transactions, where each transaction T_j $(1 \leq j \leq n)$ comprises a subset of items, and has a transaction identifier (tid) j. An association rule r is an implication having the form $X \rightarrow Y$ where X and Y are sets of items and $X \cap Y = \emptyset$. X and Y are called the antecedent and consequent of the rule, respectively. The meaning of r is that if the antecedent X appears in a transaction T_j, that is $X \subset T_j$, then the consequent Y is likely to appear in the same transaction T_j. Two important criteria in association rule mining are the support and confidence as they can reveal reliable rules for specific applications [14].

Definition 1 Tidset (Transaction identifier set)

The tidset of an itemset X can be expressed as $tids(X)$ and defined as $tids(X) = \{j | T_j \in DB \wedge X \subseteq T_j\}$. For a rule $r : X \rightarrow Y$, its tidset is denoted as $tids(r)$ and is equivalent to $tids(X \cup Y) = tids(X) \cap tids(Y)$.

Example 1 The database DB_1 of Table 1 is used to provide examples illustrating definitions throughout this section. According to Definition 1, $tids(b) = \{1, 2, 3, 5\}$, $tids(\{a, b, d\}) = \{1, 2, 5\}$.

Table 1 An example database DB_1

Tid	Items
1	a,b,c,d
2	a,b,c,d,e
3	a,b,c,e
4	a,c,d,e
5	a,b,d

Definition 2 Support

The support $sup(r)$ of an association rule $r : X \rightarrow Y$ is defined as the proportion of transactions satisfied by r in DB, where T_j is said to be satisfied by r if it contains all the items of r. The support $sup(X)$ of an itemset X in a database is similarly defined.

$$sup(X) = \frac{|tids(X)|}{|DB|} \tag{1}$$

$$sup(r) = \frac{|tids(X \cup Y)|}{|DB|} \tag{2}$$

Definition 3 Confidence

Rule confidence is defined as the probability that the consequent Y of a rule is satisfied when its antecedent X is satisfied. This quality measure thus assesses how dependent the set of items Y is on the set of items X.

$$conf(r) = \frac{|tids(X \cup Y)|}{|tids(X)|} \tag{3}$$

Both $minsup$ and $minconf$ are parameters that users need to set before starting the association rule mining process, and their values are in the [0,1] interval.

Example 2 Table 2 lists some association rules found in the DB_1 database and their corresponding support and confidence values.

Rules that both have a support no less than $minsup$ and a confidence no less than $minconf$ are called strong rules or valid rules, and mining them in a given database is the task of ARM.

Table 2 Example association rules

Rule	Support	Confidence
{a}→{c}	0.80	0.80
{d}→{a,e}	0.40	0.50
{b,c}→{a}	0.60	1.00
{a,d}→{c}	0.60	0.75
{a,c,e}→{b}	0.40	0.67

3.2 Top-k association rule mining

To address the problem that setting the $minsup$ parameter is unintuitive in traditional ARM, it was proposed to mine the top-k association rules.

Given a transaction database DB, an integer k and the $minconf$ threshold, the problem of mining the top-k association rules is to discover a set L with k rules such that for each rule $r \in L | conf(r) \geq minconf$, and there is no rule $s \notin L | conf(s) \geq minconf \wedge sup(s) \geq sup(r)$ [14]. It can be seen that the set L contains the k most frequent association rules that satisfy the confidence threshold. In addition, it is important to note that when multiple rules have the same support, the set L can contain more than k rules. Besides, it is also possible that no set L having k rules may be found if the database is too small. In this case, a set L of less than k rules may be shown to the user.

Example 3 For the database DB_1, $k = 5$ and $minconf = 0.80$, the set L of top-k association rules is shown in Table 3 . In this case, the total number of rules exceeds k.

Definition 4 Candidate item

To explore the search space of rules, algorithms such as TopKRules and ETARM repeatedly apply a process called rule expansion, which consists of adding an item to either the left or right side of a rule to obtain a new rule. A candidate item is an item that is larger than all items in the expanded side (left or right) of a rule according to the total order and that does not appear in the other side.

Definition 5 Left expansion

A left expansion is the process of adding an item $i \in I$ to the antecedent of a rule $X \rightarrow Y$ to generate a new rule $X \cup \{i\} \rightarrow Y$.

Definition 6 Right expansion

Table 3 The top-5 association rules found in DB_1

Rule	Support	Confidence
{a}→{b}	0.80	0.80
{b}→{a}	0.80	1.00
{a}→{c}	0.80	0.80
{c}→{a}	0.80	1.00
{a}→{d}	0.80	0.80
{d}→{a}	0.80	1.00

A right expansion is the process of adding an item $i \in I$ to the consequent of a rule $X \rightarrow Y$ to generate a new rule $X \rightarrow Y \cup \{i\}$.

Property 1 For an item $i \in I$, the two association rules $r : X \rightarrow Y$ and $r' : X \cup \{i\} \rightarrow Y$ satisfy $sup(r) \geq sup(r')$.

Property 2 For an item $i \in I$, the two association rules $r : X \rightarrow Y$ and $r' : X \rightarrow Y \cup \{i\}$ satisfy $sup(r) \geq sup(r')$.

Proof According to Definition 1, the tidset of a rule r is the intersection of the tidsets of all items that it contains. Therefore, whether r is left expanded or right expanded, the tidset of the new rule r' obtained must be a subset of the tidset of r, that is, $|tids(r)| \geq |tids(r')|$. From Definition 2, $sup(r) = |tids(r)|/|DB|$, and $sup(r') = |tids(r')|/|DB|$, that is, $sup(r) \geq sup(r')$. Hence, Properties 1 and 2 are proved. □

Properties 1 and 2 indicate that applying left and/or right expansions to a rule will only produce rules having a support that is not greater than that of the original rule, which means that an infrequent rule cannot generate frequent rules through expansions. It should be noted that the above conclusions do not apply to the confidence of rules, as the next two properties state.

Property 3 If an item $i \in I$ is added to the antecedent of a rule $r : X \rightarrow Y$ to generate a new rule $r' : X \cup \{i\} \rightarrow Y$, then the confidence of r' may be greater than, equal to, or less than that of r.

Property 4 For an item $i \in I$, the two association rules $r : X \rightarrow Y$ and $r' : X \rightarrow Y \cup \{i\}$ satisfy $conf(r) \geq conf(r')$.

Proof According to Definition 3, for the rule $r : X \rightarrow Y$ and $r' : X \rightarrow Y \cup \{i\}$, $conf(r) = |tids(X \cup Y)|/|tids(X)|$ and $conf(r') = |tids(X \cup Y \cup \{i\})|/|tids(X)|$. Where $tids(X \cup Y \cup \{i\})$ is a subset of $tids(X \cup Y)$, that is $|tids(X \cup Y)| \geq |tids(X \cup Y \cup \{i\})|$. Therefore, $conf(r) \geq conf(r')$ and Property 4 is proved. □

4 The FTARM algorithm

To find the top-k association rules, the state-of-the-art ETARM and TopKRules algorithms perform rule expansions starting from rules containing two items and using an internal *minsup* threshold set to zero. Then when k rules are found, these algorithms raise the *minsup* threshold to that of the least frequent rule among the top-k rules until now, which allow to reduce the search space. Though this process guarantees finding the top-k association rules, performance is often unsatisfying and largely depends on how fast the *minsup* threshold can be raised.

The proposed FTARM algorithm uses a novel rule generation property pruning (RGPP) technique to improve mining performance, and one of its significant advantages is that it can raise the *minsup* threshold more quickly. Different from existing algorithms, FTARM prunes the search space before and during rule expansions. Before rule expansions, the internal relationships between items in the input database and parameters set by the user are first analyzed. Not only useless items are eliminated directly, but also the internal minimum support value that influences the number of rules is set based on that analysis instead of being set to zero (as in ETARM and TopKRules). Then, in the process of rule expansion, a variable used to record the largest database item is updated in real time as the *minsup* threshold is raised so as to further reduce the search space.

This section first explains the novel propositions adopted by FTARM. Then, the detailed design of FTARM is described and an example is provided to illustrate how it is applied.

4.1 Three novel propositions

The next paragraphs first present two properties that will be used to discuss the novel propositions.

Property 5 Let there be a set of items $I = \{i_1, i_2, i_3, ..., i_m\}$. The total number of rules that can be generated from these items is given by the following equation:

$$S = \sum_{j=1}^{m-1} C_m^j (2^{m-j} - 1) \tag{4}$$

By Property 5, the minimum number of items m needed to generate k rules can be derived from the user-defined parameter k. For example, when $k = 30$, the minimum number of required items m is 4.

Property 6 Let there be a database $DB = \{T_1, T_2, T_3, ..., T_n\}$ and a set of items $I = \{i_1, i_2, i_3, ..., i_m\}$ such that $T_j \subseteq I$ $(1 \leq j \leq n)$. If a set $I' = \{i_s, i_{s+1}, i_{s+2}, ..., i_e\}$ is a subset of the set I, then the support of the rules generated using the set I' should be greater than or equal to M_s, and the confidence of the rules generated by the set I' should be greater than or equal to M_c.

$$M_s = \frac{|tids(i_s \cup i_{s+1}...i_{e-1} \cup i_e)|}{|DB|} \tag{5}$$

$$M_c = \frac{|tids(i_s \cup i_{s+1}...i_{e-1} \cup i_e)|}{|Max\{tids(i_s), tids(i_{s+1}), ..., tids(i_{e-1}), tids(i_e)\}|} \tag{6}$$

Proof According to Definition 2, the support of a rule $r: X \to Y$ is $sup(r) = |tids(X \cup Y)|/|DB| = |tids(X) \cap tids(Y)|/|DB|$. That is, the support of r should be the same as the support of the least supported item among all the items it contains. Therefore, if the rule r' contains all the items in the set I', its support must be the lowest support among all the rules that I' can generate, that is, $M_s = sup(r') = |tids(i_s \cup i_{s+1}...i_{e-1} \cup i_e)|/|DB|$. Assume that $r' : X' \to Y'$ is the rule having the lowest confidence among rules that can be generated using the set I'. Then, according to Definition 3, $conf(r') = |tids(X' \cup Y')|/|tids(X')|$, the numerator part should be the smallest of all possible cases, that is $|tids(X' \cup Y')| = |Min\{tids(i_s), ..., tids(i_e)\}| = |tids(i_s \cup i_{s+1}...i_{e-1} \cup i_e)|$, and the denominator part should be the largest of all possible cases, that is $|tids(X')| = |Max\{tids(i_s), ..., tids(i_e)\}|$. Therefore, $M_c = conf(r') = |tids(i_s \cup i_{s+1}...i_{e-1} \cup i_e)|/|Max\{tids(i_s), ..., tids(i_e)\}|$ and Property 6 is proved. □

Proposition 1 *Initialization of the minsup variable*

Under the premise that the user has set the k and $minconf$ parameters, the FTARM algorithm uses Property 5 to calculate the minimum number of items m required to generate k rules, and then sorts the items in descending order of support in the given database, and preferentially select m items from the items with higher support to form a set I'. If the M_c value of set I' is greater than or equal to the $minconf$ threshold, the process is stopped and $minsup$ can be initialized to the M_s of I'.

Rationale For set I', the number of rules S that can meet the $minconf$ threshold is greater than or equal to k, and the support of all rules is not less than M_s. Therefore, the support of the top-k frequent association rules to be mined must not be less than M_s, and the value of the $minsup$ variable can be initialized to M_s.

Proposition 2 *Remove useless items*

According to the $minsup$ variable that has been initialized, each item i from the database such that $sup(i) < minsup$ can be removed from the database and ignored when performing subsequent rule expansions.

Rationale Expanding a frequent rule $r : X \to Y$ with an item i satisfying $sup(i) < minsup$ will result in a rule r' that is infrequent no matter whether the rule r is expanded to the left or right with item i.

However, in the TopKRules and ETARM algorithms, the $minsup$ variable is initialized to zero and all items in the database are preserved before starting rule expansions. As a consequence, these algorithms can generate much more candidate rules than the proposed FTARM algorithm, and they typically perform much more expansion operations as it will be shown in the experimental evaluation of this paper, and this results in consuming more time and memory.

Proposition 3 *Update the largest item in real time*

During the execution of the algorithm, when the $minsup$ value is raised, the largest item in the database is updated to the item that satisfies the $minsup$ threshold and is the largest according to the total order.

Rationale In the ETARM algorithm, if the largest item in the antecedent (consequent) of a rule according to the total order is also the largest item in the database, the rule should not be expanded by a left (right) expansion. But ETARM does not take into account the following situation: let I_c be a candidate item set for the antecedent (consequent) of a rule, if there is no item $i \in I_c | sup(i) \geq minsup$, the rule should also not be expanded by a left (right) expansion. Proposition 3 was defined based on this observation.

4.2 The proposed algorithm

The pseudocode of the proposed FTARM algorithm is shown in Algorithm 1. FTARM takes as parameter a database and the k and $minconf$ parameters. FTARM consists of two parts: expansion preparation and rule expansion.

At the beginning of the algorithm, the tidset of each item contained in the database DB is calculated and items are sorted according to a total order (such as the lexicographical order). Next, the Initialize_Remove procedure is called to initialize the $minsup$ variable and remove the useless items from DB. Then, for all rules generated using a pair of remaining items (i, j) such that $sup(i) \geq minsup$ and $sup(j) \geq minsup$, if the rule r is valid, it is added to a set L containing the top-k rules so far, and if r is frequent, it is added to a set R of rules to be considered for subsequent expansions. When the largest item of the antecedent of r is larger than or equal to $MaxItem$, its flag $expandLR$ is set to false, otherwise it is set to true.

In the rule expansion phase, the algorithm loops to find the rule r with the highest support in the set R. If its flag $expandLR$ is set to true, it will be expanded from the left and right, otherwise it will only be expanded from the right. After the above operations are completed, r is removed from R, and at this time, the rules having a support less than $minsup$ in R are also removed. When R is empty, the algorithm terminates, and the rules in L are the top-k rules.

Algorithm 1 **FTARM**(*DB*,*k*,*minconf*).

```
1: $R$:=∅. $L$:=∅.
2: Scan the database $DB$ to record the tidset of each item.
3: Call the Initialize_Remove procedure to initialize the minsup variable and remove useless items from $DB$.
4: for each pair of items $(i, j)$ such that $sup(i) \geq minsup$ and $sup(j) \geq minsup$ do
5:     if $sup(\{i\} \rightarrow \{j\}) \geq minsup$ then
6:         if $conf(\{i\} \rightarrow \{j\}) \geq minconf$ then
7:             Save($\{i\} \rightarrow \{j\}, L, k, minsup$).
8:         if $conf(\{j\} \rightarrow \{i\}) \geq minconf$ then
9:             Save($\{j\} \rightarrow \{i\}, L, k, minsup$).
10:        for each rule $\{i\} \rightarrow \{j\}$ and $\{j\} \rightarrow \{i\}$ do
11:            if the largest item in the antecedent is larger than or equal to $MaxItem$ (the largest item in $DB$ until now) then
12:                Set flag $expandLR$ to false.
13:            else Set flag $expandLR$ to true.
14:        end for
15:        $R := R \cup \{i\} \rightarrow \{j\} \cup \{j\} \rightarrow \{i\}$.
16: end for
17: while $\exists\ r \in R$ and $sup(r) \geq minsup$ do
18:     Select the rule $r$ with the highest support in $R$.
19:     if $r.expandLR$ = true then
20:         Expand_L($r, L, R, k, minsup, minconf$).
21:         Expand_R($r, L, R, k, minsup, minconf$).
22:     else Expand_R($r, L, R, k, minsup, minconf$).
23:     Remove $r$ from $R$.
24:     Remove rules in $R$ with support less than $minsup$.
25: end while
26: Return $L$.
```

Algorithm 2 **Initialize_Remove**(*DB*,*k*,*minconf*).

```
1: Calculate the minimum number of items $m$ required to generate $k$ rules based on $k$.
2: Sort items in descending order of support.
3: for each integer $v \in [m, |I|]$ do
4:     Among the top $v$ items with the highest support, let $I'$ be the $m$ items having the lowest support.
5:     if the $M_c$ of $I'$ satisfies $M_c \geq minconf$ then
6:         Break the for loop of Line 3.
7: end for
8: $minsup$ is set to the $M_s$ of $I'$.
9: for each item $i \in I$ do
10:    if $sup(i) < minsup$ then
11:        Remove item $i$ from the database $DB$.
12: end for
```

The Initialize_Remove procedure As shown in Algorithm 2, the minimum number of items m required to generate k rules is first calculated, and then under the condition that all items in the database DB are arranged in descending order of support, the m items are preferentially selected from items with higher support to form a set I'. If the M_c of I' satisfies the constraint that M_c is greater than or equal to $minconf$, then the search process is stopped and the $minsup$ variable′s value is initialized to the M_s of I'. Finally, the procedure traverses the database to remove items having a support less than $minsup$, and ignores them in the following search for the top-k rules.

The Save procedure (Algorithm 3). The main task of this procedure is to update the set L, $minsup$ and $MaxItem$ in real time during the algorithm's execution. Firstly, the rule r is added to L. Then, if L still contains at least k rules after removing the rules with a support of $minsup$, these rules are removed. After the removal operation, the procedure sets $minsup$ to the lowest support of the rules in the current L, and scans items to find the item i that meets $minsup$ and is the largest according to the total order, and finally records it in $MaxItem$.

Algorithm 3 **Save**(*r*,*L*,*k*,*minsup*).

```
1: $L := L \cup \{r\}$.
2: if $|L| > k$ then
3:     Calculate the number of rules in $L$ with support equal to $minsup$ and record it in a variable $count$.
4:     if $|L| - count \geq k$ then
5:         Remove rules with support equal to $minsup$ in $L$.
6:         Set $minsup$ to the lowest support of rules in $L$.
7:         $MaxItem$ := 0.
8:         for each $i \in I$ do
9:             if $sup(i) > minsup$ and $i$ is larger than $MaxItem$ in the total order then
10:                $MaxItem := i$.
11:        end for
```

As mentioned earlier, during the rule expansion phase, the Expand_L and Expand_R procedures perform the expansion operations on the candidate rules to obtain more valid rules and then add them to the set L.

Expand_L procedure is shown in Algorithm 4. The candidate item set I_c of the antecedent of the rule r to be expanded is first calculated, and then each item i in I_c is individually added to the left side of r to obtain a new rule r'. For the rule r', if it is frequent, r' is added to the set R. If r' is frequent and its confidence is not less than $minconf$, the save procedure is called to add r' to the set L. For the flag $expandLR$ of r', if the largest item of the antecedent of r' is larger than or equal to $MaxItem$, it is set to false, which means that r' is only considered for right expansions, otherwise it is set to true, and r' can be considered for both left and right expansions.

Algorithm 4 **Expand_L**(r,L,R,k,$minsup$,$minconf$).

```
1: Scan the transaction $tid$ ($tid \in tids(r)$) to get the candidate item set $I_c$ of the antecedent of $r$.
2: for each item $i \in I_c$ do
3:     Add $i$ to the antecedent of $r$ to get a new rule $r'$.
4:     if $sup(r') \geq minsup$ then
5:         if $conf(r') \geq minconf$ then
6:             Save($r', L, k, minsup$).
7:         if the largest item of the antecedent of $r'$ is larger than or equal to $MaxItem$ then
8:             Set flag $expandLR$ to false.
9:         else Set flag $expandLR$ to true.
10:        $R := R \cup \{r'\}$.
11: end for
```

Algorithm 5 **Expand_R**(r,L,R,k,$minsup$,$minconf$).

```
1: Scan the transaction $tid$ ($tid \in tids(r)$) to get the candidate item set $I_c$ of the consequent of $r$.
2: for each item $i \in I_c$ do
3:     Add $i$ to the consequent of $r$ to get a new rule $r'$.
4:     if $sup(r') \geq minsup$ and $conf(r') \geq minconf$ then
5:         Save($r', L, k, minsup$).
6:         if the largest item of the consequent of $r'$ is smaller than $MaxItem$ then
7:             Set flag $expandLR$ to false.
8:             $R := R \cup \{r'\}$.
9: end for
```

The Expand_R procedure is shown in Algorithm 5. It is very similar to the expand_L procedure, adding qualified candidate items to the right side of the rule r to generate a new rule r'. The main difference is that the conditions for r' to be added to the set R are more strict. Based on Property 4, r' is only added to the set R for right expansions if r' is valid and the largest item of its consequent is smaller than $MaxItem$.

The FTARM algorithm recursively performs left and right expansions starting from rules containing one item in the antecedent and one item in the consequent, which ensures that all possible rules can be generated. To avoid generating the same rule twice by different combinations of left and right expansions, FTARM never does a left expansion to a rule that was generated by a right expansion (based on the flag $expandLR$, which is set to false by Expand_R). Besides, to avoid exploring all possible rules, the algorithm does not expand a rule that has a support lower than $minsup$ (based on Properties 1 and 2) and does not do a right expansion of a rule that has confidence lower than $minconf$ (based on Property 4). Note that a rule that has a confidence lower than $minconf$ cannot be eliminated for left expansions due to Property 3. To further improve performance, this paper has introduced three propositions, which also guarantee that only rules that are not top-k rules are eliminated (by Properties 5 and 6). Hence, all the desired top-k rules can be obtained by FTARM.

4.3 An illustrative example

Consider that $k = 10$ and $minconf = 0.8$. Mining the top-k association rules in the database of Table 4 using FTARM is done as follows.

Step 1. Initialize sets R and L, traverse the database to calculate and record the tidset of each item (Table 5), and record the largest item in $MaxItem$ ($MaxItem$ = h).

Step 2. Based on Property 5, calculate the minimum number of items m required to generate k rules (m = 3).

Step 3. Sort all items in descending order of support, preferentially select m items from higher support items and combine them to obtain the set I' satisfying the constraint that M_c is no less than $minconf$, then initialize $minsup$ to the M_s of I', and remove all items whose support is less than $minsup$. In this example, I' is finally composed of items b, c and e, $minsup$ is initialized to 0.8, and the value of $MaxItem$ is modified to e after items a, d, f, g and h have been removed.

Step 4. Generate all rules having two items and add the rules that meet the corresponding conditions to the sets R (Table 6) and L, respectively.

Step 5. Select the rule r with the highest support in set R ($sup(r) \geq minsup$), and expand it according to its flag $expandLR$. If $expandLR$ is set to true, it will be expanded from the left and right, otherwise it will only be expanded from the right. Some examples of rule expansions can be seen in Fig. 1. In that example, the rule {b}→{c} has a support of 0.8 and a confidence of 1.0. Moreover, its flag $expandLR$ is set to true, indicating that this rule could be left and right expanded. By performing a left expansion of {b}→{c} with item e, the rule {be}→{c} can be obtained. This rule has the flag $expandLR$ set to false because the largest item of its antecedent is equal to $MaxItem$, and it is only considered for right expansions. By expanding

Table 4 A second example database DB_2

Tid	Items
1	a,b,c,e,f
2	b,c,d,e,g
3	a,b,c,e,h
4	c,d,e,f
5	a,b,c,d,e,g

Table 5 Tidsets of items

Item	Tidset	Item	Tidset
a	{1,3,5}	b	{1,2,3,5}
c	{1,2,3,4,5}	d	{2,4,5}
e	{1,2,3,4,5}	f	{1,4}
g	{2,5}	h	{3}

{b}→{c} with item e, the rule {b}→{ce} is obtained and its *expandLR* flag is set to null because the largest item of its consequent is not smaller than *MaxItem*. Hence, it cannot be added to the set R for further expansions. Other examples in Fig. 1 describes the expansions of other rules in a similar way. Through the process of expanding rules, the new rules obtained from the expansions are added to sets R and L respectively when the corresponding conditions are met. After that, r and rules having a support that is less than *minsup* are removed from R.

Then, step 5 is repeated until the set R is empty. The algorithm returns the set L, which contains the top-k association rules (Table 7).

5 Experiments

This section describes results from experiments to evaluate the performance of the proposed FTARM algorithm. Section 5.1 gives a detailed description of the datasets and the testing environment. Section 5.2 presents results from experiments that compare the performance of FTARM, ETARM and TopKRules for two typical scenarios, in terms of runtime and memory usage. Section 5.3 reports results from scalability experiments where the size of both sparse and dense datasets are varied to evaluate their influence on performance. Lastly, Section 5.4 discusses and summarizes the results.

5.1 Datasets and environment

The proposed FTARM algorithm has been implemented in Java and its performance was compared with that of the state-of-the-art TopKRules [14] and ETARM [34] algorithms. All the experiments were carried on a personal computer having an Intel Core I7-9700K 3.6 GHz processor and 16 GB of RAM, running Microsoft Windows 10. Each algorithm was run 20 times and the average results were calculated.

Table 6 Candidate rules with two items

Rule	Support	Confidence	ExpandLR
{b}→{c}	0.80	1.00	true
{c}→{b}	0.80	0.80	true
{b}→{e}	0.80	1.00	true
{e}→{b}	0.80	0.80	false
{c}→{e}	1.00	1.00	true
{e}→{c}	1.00	1.00	false

Standard benchmark datasets obtained from http://www.philippe-fournier-viger.com/spmf/ were used to compare the performance of the algorithms. Characteristics of datasets are shown in Table 8 . They are well known datasets for association rule mining and have different transaction counts, item counts and densities. The Chess, Mushrooms, Pumsb and Connect datasets have also been used in the TopKRules and ETARM papers to test their performance.

5.2 Performance comparison

To better compare the performance of the FTARM, ETARM and TopKRules algorithms, their performance was assessed for two scenarios: the parameter k is varied and the *minconf* parameter is changed.

Case 1: minconf was set to 0.8, and k was varied from 2000 to 16000.

Figure 2 shows the mining times of the algorithms for the eight datasets and different k values. The parameter settings in this experiment are similar to those used in the ETARM paper. It is observed that the proposed FTARM algorithm outperforms the other two algorithms. And as the value of k increases, the gap in terms of runtime of the three algorithms becomes larger. For example, for the Accidents dataset (Fig. 2e), when $k = 2000$, the mining times of the three algorithms are almost the same, but for $k = 16000$, the runtime of FTARM is 23.57% less than ETARM and 40.21% less than TopKRules. And for the PAMP dataset (Fig. 2h), when $k = 16000$, the runtime of FTARM is 32.68% less than ETARM and 45.10% less than TopKRules. On the Chess, Mushrooms, Connect and RecordLink datasets, the percentage of runtime reduction obtained by FTARM compared with ETARM for $k = 16000$ are 30.35%, 21.36%, 27.83% and 41.93%, respectively.

Figure 3 shows the maximum amounts of memory used by FTARM, ETARM and TopKRules for various k values. It can be seen that FTARM consumes less memory than the others. For the Connect dataset (Fig. 3d), when $k = 16000$, the memory consumption of ETARM reaches more than twice that of FTARM, and under the same parameter settings, the memory consumption of ETARM on the Pumsb dataset (Fig. 3c) is equivalent to three times that of FTARM. This reduction in memory consumption is also evident for larger datasets, such as PAMP (Fig. 3h), where as k is increased the memory consumption of FTARM increases slowly, while under the same experimental conditions, the memory consumption of ETARM increases to more

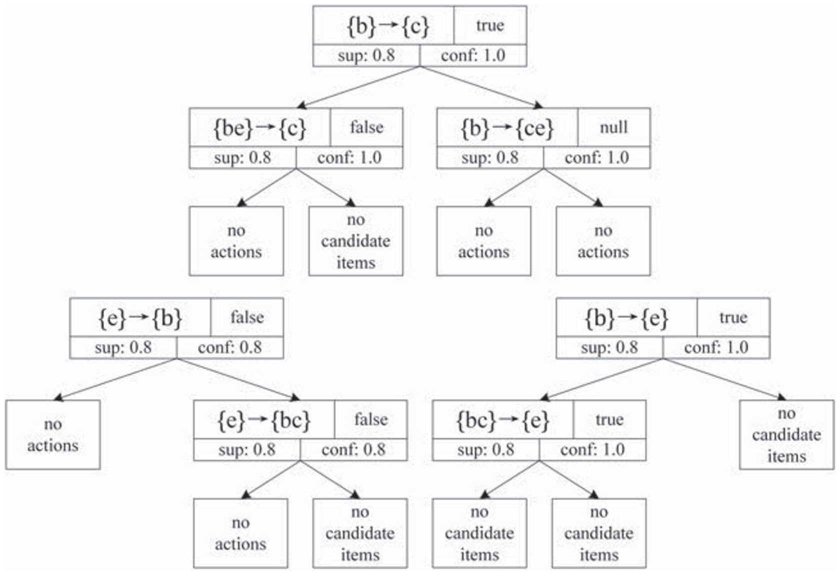


Fig. 1 Examples of rule expansions

than twice that of FTARM, and the maximum memory consumption of TopKRules even exceeds three times that of FTARM.

Case 2: k was set to 8000, and $minconf$ was varied from 0.3 to 0.8.

Figure 4 illustrates the runtimes of FTARM, ETARM and TopKRules for the different datasets and parameter values. The range of $minconf$ values in this experiment is larger than the one used in the TopKRules paper. It is observed that FTARM performs best. For instance, on the Chess dataset (Fig. 4a), when $minconf = 0.8$, the time used by FTARM is 22.37% less than ETARM and 32.13% less than TopKRules. And on the Connect dataset (Fig. 4d), when $minconf = 0.8$, the runtime of FTARM is 26.72% less than ETARM and 31.96% less than TopKRules. Similarly, FTARM still performs well on larger datasets. For the OnlineRetail dataset (Fig. 4f), the runtime of FTARM changes less when $minconf$ is varied than the other two algorithms, and it tends to be more stable. Moreover, when $minconf = 0.8$, the gap in terms of runtime between FTARM and the other two algorithms is the largest, FTARM takes 21.16% less time than ETARM and 26.89% less than TopKRules. And on the RecordLink dataset (Fig. 4g), when $minconf = 0.8$, FTARM takes 35.23% less time than ETARM and even 55.12% less than TopKRules.

In terms of memory consumption, as can be seen in Fig. 5, FTARM still has an absolute advantage just as in case 1. For instance, for the Pumsb dataset (Fig. 5c), when $minconf = 0.8$, the memory consumption of ETARM is more than twice that of FTARM, while TopKRules consumes almost three times that of FTARM. And for the Mushrooms dataset (Fig. 5b), when $minconf = 0.8$, the maximum memory consumption of ETARM also reaches more than twice that of FTARM, and under the same experimental conditions, the memory consumed by TopKRules is more than three times that of FTARM. Moreover, on larger datasets, FTARM also significantly reduces memory consumption compared to the other two algorithms, as only a small amount of memory is used to complete the same mining task. For example, on the RecordLink dataset (Fig. 5g), when $minconf = 0.8$, FTARM consumes 41.99% less memory than ETARM, and 60.91% less than TopKRules.

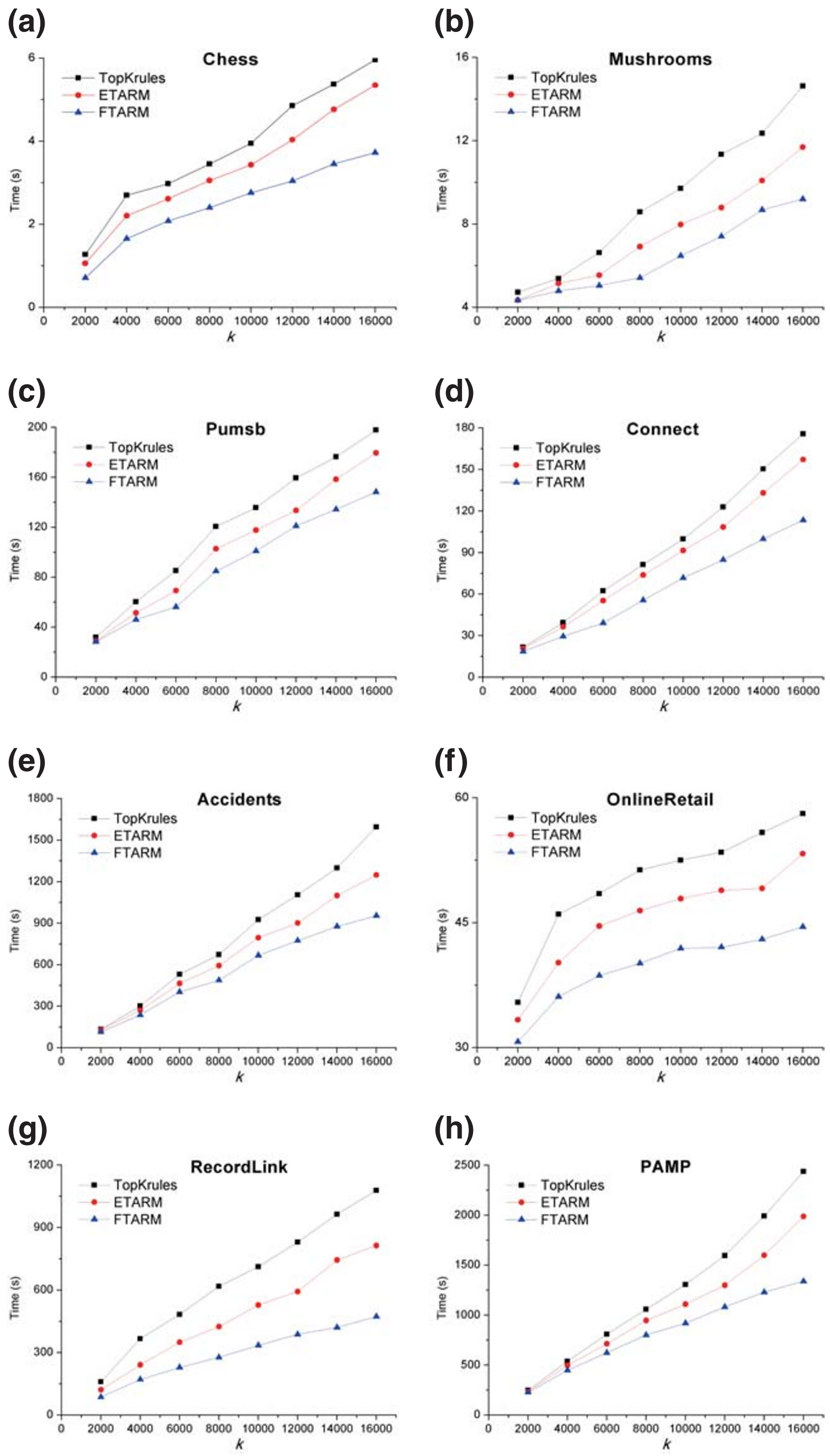
Table 7 The top-10 association rules found in DB_2

Rule	Support	Confidence	Rule	Support	Confidence
{b}→{c}	0.80	1.00	{c}→{b}	0.80	0.80
{b}→{e}	0.80	1.00	{e}→{b}	0.80	0.80
{c}→{e}	1.00	1.00	{e}→{c}	1.00	1.00
{c}→{b,e}	0.80	0.80	{c,e}→{b}	0.80	0.80
{b}→{c,e}	0.80	1.00	{b,e}→{c}	0.80	1.00
{e}→{b,c}	0.80	0.80	{b,c}→{e}	0.80	1.00

Table 8 Characteristics of the experimental datasets

Name	Transaction count	Item count	Density
Chess	3,196	75	0.493
Mushrooms	8,416	119	0.193
Pumsb	49,046	2,113	0.035
Connect	67,557	129	0.333
Accidents	340,183	468	0.072
OnlineRetail	541,909	2,603	0.002
RecordLink	574,913	29	0.345
PAMP	1,000,000	141	0.170

Fig. 2 Time analysis for different datasets in case 1



5.3 Scalability analysis

To assess the impact of database size on the performance of the FTARM, ETARM and TopKRules algorithms, scalability experiments were conducted on the Pumsb (sparse) and Connect (dense) datasets for different database sizes. The size of the database was increased by 20%, while k and $minconf$ were set to 16000 and 0.8 respectively, and the runtime and memory consumption of the three algorithms were recorded.

As can be seen in Fig. 6, as database size is increased, the runtime of the three algorithms increases, while FTARM

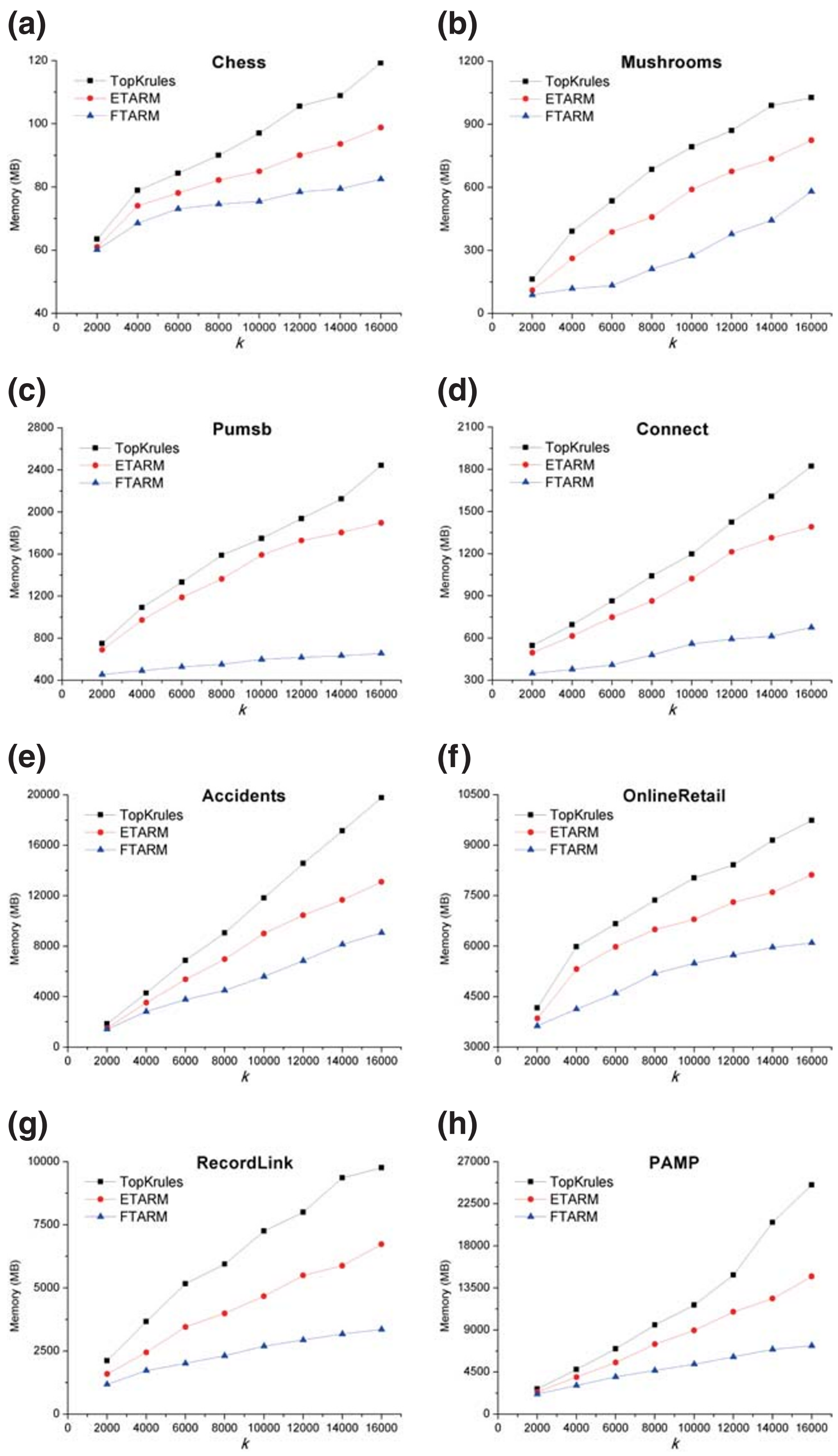


Fig. 3 Memory analysis for different datasets in case 1

always outperforms the other two algorithms. In terms of maximum memory consumption, the three algorithms also show an upward trend as database size is increased, and the performance of FTARM in terms of memory consumption is considerably better than that of the other two algorithms. On overall, it is observed that the runtime and memory usage of FTARM remains better than that of ETARM and TopKRules when processing more data. Thus, it can be concluded that the proposed FTARM algorithm has good scalability, and scales well on both sparse and dense datasets.

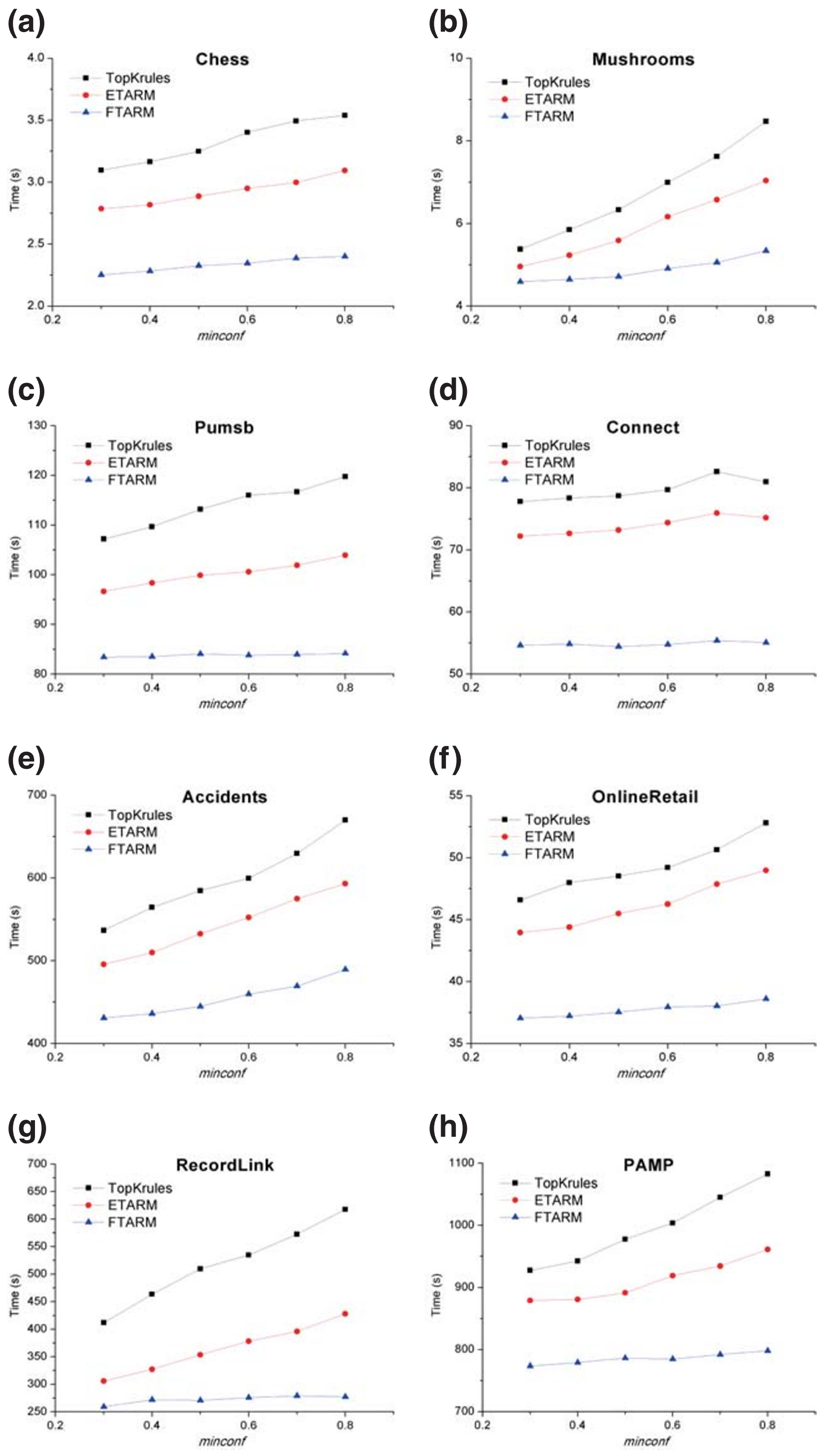
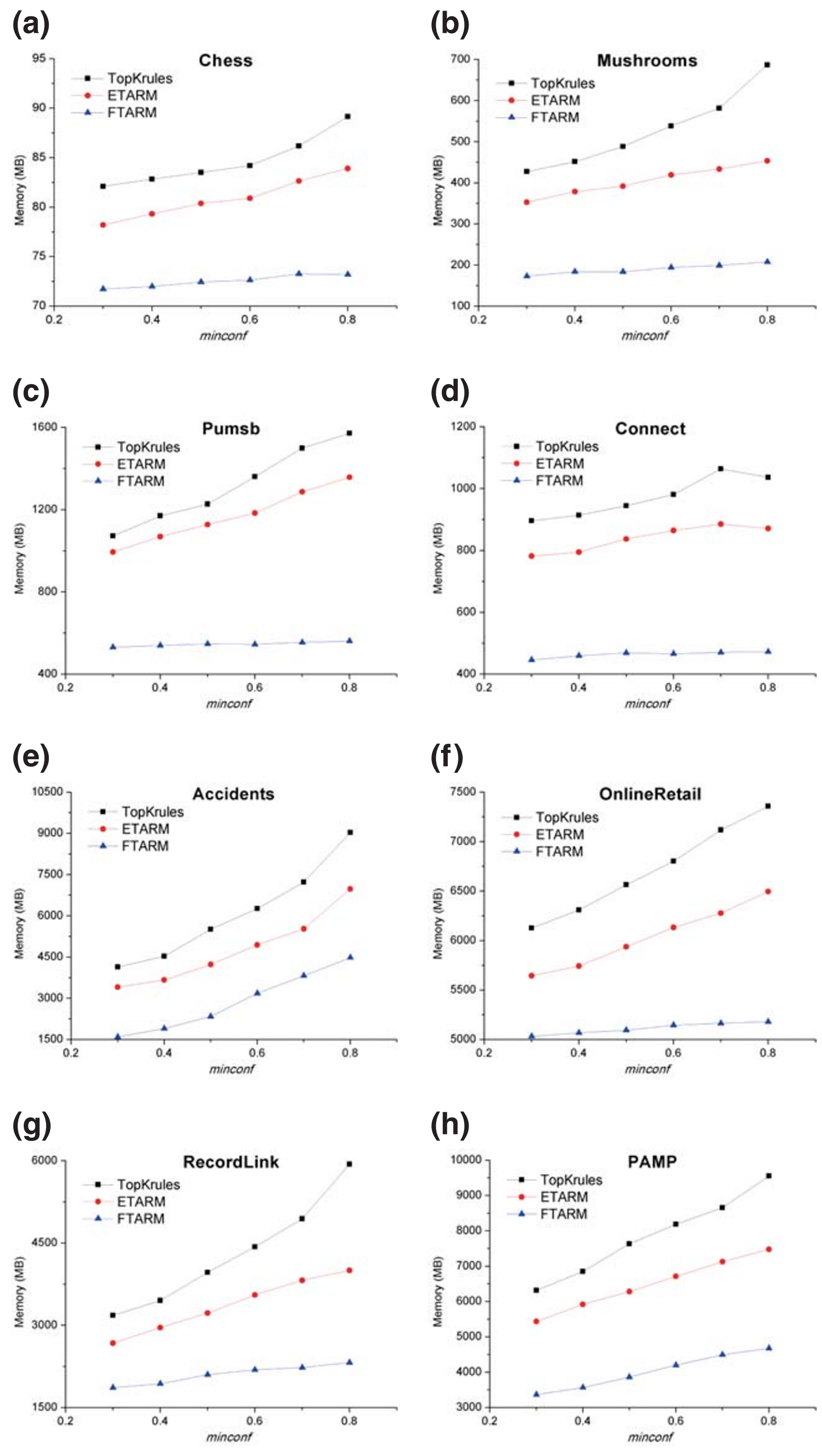


Fig. 4 Time analysis for different datasets in case 2

5.4 Discussion and summary

To comprehensively evaluate the proposed FTARM algorithm's performance, we compared its performance with the state-of-the-art ETARM and TopKRules algorithms. We conducted experiments to evaluate the influence of the k parameter, the $minconf$ threshold, and database size on performance, in terms of memory and runtime. Based on the three novel propositions proposed in this paper, FTARM raises the internal $minsup$ threshold faster, removes useless items and updates the largest item in real-time, all of which greatly reduce the search space for mining the top-k

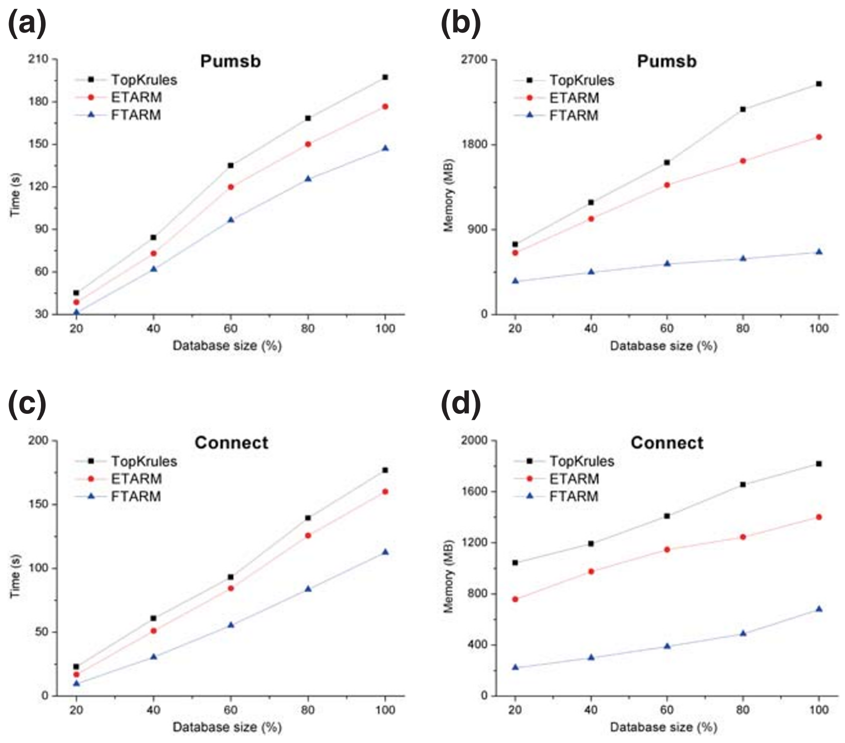
Fig. 5 Memory analysis for different datasets in case 2



rules. Compared with ETARM and TopKRules, FTARM generates much less candidate rules, which explains its substantially higher speed and lower memory consumption. Experimental results also show that the proposed algorithm can scale well on both sparse and dense datasets.

It should be noted that although the ETARM algorithm and the FTARM algorithm prune the search space for mining the top-k rules, the results of the three algorithms are exactly the same. For ETARM, if the largest item in the antecedent (consequent) of a rule is equal to $MaxItem$, it

Fig. 6 Experimental results of scalability evaluation



is and should not be considered for left (right) expansions. And if the confidence of a rule is less than $minconf$, it is and should not be considered for right expansions. FTARM extends ETARM with novel pruning strategies that also reduce the number of unnecessary rule expansions, thus further reducing the number of candidate rules. In general, both ETARM and FTARM proposed additional pruning strategies compared to TopKRules and those are based on relevant properties of association rules. These pruning strategies can eliminate some parts of the search space that do not contain the top-k rules. Hence, these strategies have no effect on the mining results.

6 Conclusion and future work

This paper presented a novel algorithm named FTARM that uses a novel rule generation property pruning (RGPP) technique to mine the top-k association rules. The advantage of FTARM lies in that it removes useless items and raises the internal $minsup$ threshold before rule expansions by analyzing the internal relationships between the parameters set by users and the items of the database to be mined, and in the process of rule expansion, a novel candidate pruning property is also used to further reduce the search space by updating the largest item in real time. Thus, the time and memory required for mining the top-k association rules are greatly reduced. Extensive experiments have shown that FTARM outperforms both ETARM and TopKRules for various datasets.

However, the proposed algorithm still requires that users set the $minconf$ parameter to mine the top-k association rules. Thus, extending FTARM to mine the top-k rules having the highest confidence without $minconf$ parameter is an interesting topic for future research. In addition, besides the support and confidence, the lift and other measures could be used to mine the top-k association rules. And to process much larger datasets, a parallel/distributed implementation of FTARM could be developed.

Acknowledgments This research is sponsored by the Science and Technology Planning Project of Sichuan Province under Grant No. 2020YFG0054, and the Scientific Research Project of State Grid Sichuan Electric Power Company Information and Communication Company under Grant No. SGSCXT00XGJS1800219.

References

1. Agrawal R, Imielinski T, Swami AN (1993) Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, pp 207–216. https://doi.org/10.1145/170035.170072
2. Alwidian J, Hammo B, Obeid N (2018) WCBA: Weighted Classification based on association rules algorithm for breast cancer disease. Appl Soft Comput 62:536–549. https://doi.org/10.1016/j.asoc.2017.11.013

3. Anand HS, Vinodchandra SS (2018) Association rule mining using treap. Int J Mach Learn Cybern 9(4):589–597. https://doi.org/10.1007/s13042-016-0546-7
4. Anwar T, Uma V (2019) CD-SPM: Cross-domain book recommendation using sequential pattern mining and rule mining. Journal of King Saud University. https://doi.org/10.1016/j.jksuci.2019.01.012
5. Aqra I, Ghani NA, Maple C, Machado JM, Safa NS (2019) Incremental algorithm for association rule mining under dynamic threshold. Appl Sci 9(24):5398. https://doi.org/10.3390/app9245398
6. Aryabarzan N, Minaeibidgoli B, Teshnehlab M (2018) negFIN: An efficient algorithm for fast mining frequent itemsets. Expert Syst Appl 105:129–143. https://doi.org/10.1016/j.eswa.2018.03.041
7. Bustiomartinez L, Letrasluna M, Cumplido R, Hernandezleon R, Feregrinouribe C, Bandeserrano JM (2019) Using hashing and lexicographic order for Frequent Itemsets Mining on data streams. J Parallel Distrib Comput 125:58–71. https://doi.org/10.1016/j.jpdc.2018.11.002
8. Chon KW, Hwang SH, Kim M (2018) GMiner: A fast GPU-based frequent itemset mining method for large-scale data. Inf Sci:19–38. https://doi.org/10.1016/j.ins.2018.01.046
9. Chuang K-T, Huang J-L, Chen M-S (2008) Mining top-k frequent patterns in the presence of the memory constraint. VLDB J 17(5):1321–1344. https://doi.org/10.1007/s00778-007-0078-6
10. Czibula G, Czibula IG, Miholca D, Crivei LM (2019) A novel concurrent relational association rule mining approach. Expert Syst Appl 125:142–156. https://doi.org/10.1016/j.eswa.2019.01.082
11. Deng Z (2014) Fast mining Top-Rank-k frequent patterns by using Node-lists. Expert Syst Appl 41(4):1763–1768. https://doi.org/10.1016/j.eswa.2013.08.075
12. Djenouri Y, Belhadi A, Fournier-Viger P (2018) Extracting useful knowledge from event logs: a frequent itemset mining approach. Knowl Based Syst 139:132–148. https://doi.org/10.1016/j.knosys.2017.10.016
13. Djenouri Y, Comuzzi M (2017) Combining Apriori heuristic and bio-inspired algorithms for solving the frequent itemsets mining problem. Inf Sci 420:1–15. https://doi.org/10.1016/j.ins.2017.08.043
14. Fournier-Viger P, Wu C, Tseng VS (2012) Mining top-k association rules. In: Proceedings of the 25th canadian conference on artificial intelligence, pp 61–73. https://doi.org/10.1007/978-3-642-30353-1_6
15. Fournier-Viger P, Zhang Y, Lin JC, Fujita H, Koh YS (2019) Mining local and peak high utility itemsets. Inf Sci 481:344–367. https://doi.org/10.1016/j.ins.2018.12.070
16. Gan W, Lin JC, Fournier-Viger P, Chao H, Hong T, Fujita H (2018) A survey of incremental high-utility itemset mining. Wiley Interdiscip Rev-Data Min Knowl Discov 8(2). https://doi.org/10.1002/widm.1242
17. Han X, Liu X, Chen J, Lai G, Gao H, Li J (2019) Efficiently mining frequent itemsets on massive data. IEEE Access 7:31409–31421. https://doi.org/10.1109/access.2019.2902602
18. Hashem T, Karim MR, Samiullah M, Ahmed CF (2017) An efficient dynamic superset bit-vector approach for mining frequent closed itemsets and their lattice structure. Expert Syst Appl 67:252–271. https://doi.org/10.1016/j.eswa.2016.09.023
19. Heydari M, Yousefli A (2017) A new optimization model for market basket analysis with allocation considerations: a genetic algorithm solution approach. Manag Market 12(1):1–11. https://doi.org/10.1515/mmcks-2017-0001
20. Huynhthile Q, Le T, Vo B, Le B (2015) An efficient and effective algorithm for mining top-rank-k frequent patterns. Expert Syst Appl 42(1):156–164. https://doi.org/10.1016/j.eswa.2014.07.045
21. Jorritsma W, Cnossen F, Dierckx R, Oudkerk M, Van Ooijen PMA (2016) Pattern mining of user interaction logs for a post-deployment usability evaluation of a radiology PACS client. Int J Med Inform 85(1):36–42. https://doi.org/10.1016/j.ijmedinf.2015.10.007
22. Khan S, Parkinson S (2018) Eliciting and utilising knowledge for security event log analysis: an association rule mining and automated planning approach. Expert Syst Appl 113:116–127. https://doi.org/10.1016/j.eswa.2018.07.006
23. Kieu T, Vo B, Le T, Deng Z, Le B (2017) Mining top-k co-occurrence items with sequential pattern. Expert Syst Appl 85:123–133. https://doi.org/10.1016/j.eswa.2017.05.021
24. Krishnamoorthy S (2019) Mining top-k high utility itemsets with effective threshold raising strategies. Expert Syst Appl 117:148–165. https://doi.org/10.1016/j.eswa.2018.09.051
25. Le T, Vo B (2016) The lattice-based approaches for mining association rules: a review. Wiley Interdiscip Rev-Data Min Knowl Discov 6(4):140–151. https://doi.org/10.1002/widm.1181
26. Le T, Vo B, Baik SW (2018) Efficient algorithms for mining top-rank-k erasable patterns using pruning strategies and the subsume concept. Eng Appl Artif Intell 68:1–9. https://doi.org/10.1016/j.engappai.2017.09.010
27. Le T, Vo B, Huynh V, Nguyen NT, Baik SW (2020) Mining top-k frequent patterns from uncertain databases. Appl Intell:1–11. https://doi.org/10.1007/s10489-019-01622-1
28. Li J, Ma X, Zhang J, Tao J, Wang P, Guan X (2017) Mining repeating pattern in packet arrivals: Metrics, models, and applications. Inf Sci 408:1–22. https://doi.org/10.1016/j.ins.2017.04.033
29. Lin JC, Gan W, Fournier-Viger P, Hong T, Tseng VS (2016) Fast algorithms for mining high-utility itemsets with various discount strategies. Adv Eng Inform 30(2):109–126. https://doi.org/10.1016/j.aei.2016.02.003
30. Mai T, Vo B, Nguyen LTT (2017) A lattice-based approach for mining high utility association rules. Inf Sci 399:81–97. https://doi.org/10.1016/j.ins.2017.02.058
31. Mlakar U, Zorman M, Fister I (2017) Modified binary cuckoo search for association rule mining. J Intell Fuzzy Syst 32(6):4319–4330. https://doi.org/10.3233/JIFS-16963
32. Moslehi F, Haeri A, Martinezalvarez F (2020) A novel hybrid GA–PSO framework for mining quantitative association rules. In: soft computing, pp 4645–4666. https://doi.org/10.1007/s00500-019-04226-6
33. Nguyen D, Luo W, Phung D, Venkatesh S (2018) LTARM: A novel temporal association rule mining method to understand toxicities in a routine cancer treatment. Knowl Based Syst 161:313–328. https://doi.org/10.1016/j.knosys.2018.07.031
34. Nguyen LTT, Vo B, Nguyen LTT, Fournier-Viger P, Selamat A (2017) ETARM: An efficient top-k association rule mining algorithm. Appl Intell 48(5):1148–1160. https://doi.org/10.1007/s10489-017-1047-4
35. Raj S, Ramesh D, Sreenu M, Sethi KK (2020) EAFIM: Efficient apriori-based frequent itemset mining algorithm on Spark for big transactional data. Knowl Inf Syst 62(9):3565–3583. https://doi.org/10.1007/s10115-020-01464-1
36. Ryang H, Yun U (2015) Top- k high utility pattern mining with effective threshold raising strategies. Knowl Based Syst 76(1):109–126. https://doi.org/10.1016/j.knosys.2014.12.010
37. Sahoo J, Das AK, Goswami A (2015) An efficient approach for mining association rules from high utility itemsets. Expert Syst Appl 42(13):5754–5778. https://doi.org/10.1016/j.eswa.2015.02.051
38. Son LH, Chiclana F, Kumar R, Mittal M, Khari M, Chatterjee JM, Baik SW (2018) ARM-AMO: An efficient association rule mining algorithm based on animal migration optimization. Knowl Based Syst 154:68–80. https://doi.org/10.1016/j.knosys.2018.04.038

39. Telikani A, Gandomi AH, Shahbahrami A (2020) A survey of evolutionary computation for association rule mining. Information Sciences. https://doi.org/10.1016/j.ins.2020.02.073
40. Thabtah F, Qabajeh I, Chiclana F (2016) Constrained dynamic rule induction learning. Expert Syst Appl 63:74–85. https://doi.org/10.1016/j.eswa.2016.06.041
41. Tseng VS, Wu C, Fournier-Viger P, Yu PS (2016) Efficient algorithms for mining Top-K high utility itemsets. IEEE Trans Knowl Data Eng 28(1):54–67. https://doi.org/10.1109/TKDE.2015.2458860
42. Vo B, Bui H, Vo T, Le T (2020) Mining top-rank-k frequent weighted itemsets using WN-list structures and an early pruning strategy. Knowl-Based Syst 201-202:106064. https://doi.org/10.1016/j.knosys.2020.106064
43. Wang J, Han J, Lu Y, Tzvetkov P (2005) TFP: An efficient algorithm for mining top-k frequent closed itemsets. IEEE Trans Knowl Data Eng 17(5):652–664. https://doi.org/10.1109/TKDE.2005.81
44. Wang L, Meng J, Xu P, Peng K (2018) Mining temporal association rules with frequent itemsets tree. Appl Soft Comput 62:817–829. https://doi.org/10.1016/j.asoc.2017.09.013
45. Webb GI (2011) Filtered-top-k association discovery. Wiley Interdiscip Revi-Data Min Knowl Discov 1(3):183–192. https://doi.org/10.1002/widm.28
46. Webb GI, Zhang S (2005) K-Optimal Rule discovery. Data Min Knowl Disc 10(1):39–79. https://doi.org/10.1007/s10618-005-0255-4
47. Wen F, Zhang G, Sun L, Wang X, Xu X (2019) A hybrid temporal association rules mining method for traffic congestion prediction. Comput Ind Eng 130:779–787. https://doi.org/10.1016/j.cie.2019.03.020
48. Xiong X, Chen F, Huang P, Tian M, Hu X, Chen B, Qin J (2018) Frequent itemsets mining with differential privacy over Large-Scale data. IEEE Access 6:28877–28889. https://doi.org/10.1109/access.2018.2839752
49. Zhang Z, Chai N, Ostrosi E, Shang Y (2019) Extraction of association rules in the schematic design of product service system based on pareto-MODGDFA. Comput Ind Eng 129:392–403. https://doi.org/10.1016/j.cie.2019.01.040
50. Zhang Z, Pedrycz W, Huang J (2017) Efficient frequent itemsets mining through sampling and information granulation. Eng Appl Artif Intell 65:119–136. https://doi.org/10.1016/j.engappai.2017.07.016