Applied Intelligence
https://doi.org/10.1007/s10489-019-01622-1

Mining top-*k* frequent patterns from uncertain databases

Tuong Le[1] · Bay Vo[2] · Van-Nam Huynh[3] · Ngoc Thanh Nguyen[4] · Sung Wook Baik[5]

Abstract
Mining uncertain frequent patterns (UFPs) from uncertain databases was recently introduced, and there are various approaches to solve this problem in the last decade. However, systems are often faced with the problem of too many UFPs being discovered by the traditional approaches to this issue, and thus will spend a lot of time and resources to rank and find the most promising patterns. Therefore, this paper introduces a task named mining top-*k* UFPs from uncertain databases. We then propose an efficient method named TUFP (mining Top-*k* UFPs) to carry this out. Effective threshold raising strategies are introduced to help the proposed algorithm reduce the number of generated candidates to enhance the performance in terms of the runtime as well as memory usage. Finally, several experiments on the number of generated candidates, mining time, memory usage and scalability of TUFP and two state-of-the-art approaches (CUFP-mine and LUNA) were conducted. The performance studies show that TUFP is efficient in terms of mining time, memory usage and scalability for mining top-*k* UFPs.

Keywords Pattern mining · Uncertain frequent pattern · Top-*k* uncertain frequent patterns

1 Introduction

The development of the Internet leads to a huge amount of data generated every day. Therefore, various methods of data analysis have thus been presented to find and use the knowledge from large information. One of these methods is data mining, which is the computational process to discover many kinds of useful knowledge in large datasets. Nowadays, data mining has been utilized comprehensively in various applications, such as the online e-commerce carried out by Amazon, Alibaba, and so on, and the analysis of shopper basket data, biomedical data, traffic data, network data, and mobile data, as well as in recommendation systems that apply association rule mining [1–3], clustering [4], classification [5–9], text mining [10, 11] and related applications [12, 13]. Mining frequent patterns (FPs) and variations [14–17] is one of the most popular problems in data mining and plays an important role in several sub-topics including association rule mining, sequential mining and classification. Mining FPs from precise databases has thus attracted much research interest from researchers around the world [1, 18–20].

In real-life applications, databases are usually collected from noisy data sources, such as acoustic, electromagnetic, optical radiation, and wireless sensors, WiFi systems, mechanical, GPS in environmental surveillance, security, chemical, and manufacturing systems. These data sources have many errors [21], due to (i) inherent measurement inaccuracies, (ii) sampling frequency of the sensors, (iii) deviations caused by rapid changes (e.g., drift and noise) to the measured property over time, (iv) wireless transmission errors, or (v) network

✉ Bay Vo
vd.bay@hutech.edu.vn

Tuong Le
tuonglecung@gmail.com

Van-Nam Huynh
huynh@jaist.ac.jp

Ngoc Thanh Nguyen
ngoc-thanh.nguyen@pwr.edu.pl

Sung Wook Baik
sbaik@sejong.ac.kr

[1] Institute of Research and Development, Duy Tan University, Da Nang 550000, Vietnam

[2] Faculty of Information Technology, Ho Chi Minh City University of Technology (HUTECH), Ho Chi Minh City, Vietnam

[3] Japan Advanced Institute of Science and Technology, Nomi, Ishikawa, Japan

[4] Faculty of Computer Science and Management, Wroclaw University of Science and Technology, Wroclaw, Poland

[5] Digital Contents Research Institute, Sejong University, Seoul, Republic of Korea

Published online: 23 January 2020

latencies. We cannot apply traditional pattern mining algorithms that are used with more precise databases to these uncertain databases, and thus it is necessary to develop methods for mining knowledge from this kind of data. Currently, there are many algorithms developed to discover useful patterns in uncertain databases, including uncertain frequent patterns (UFPs) [22, 23], uncertain weighted frequent itemsets [24], uncertain high-utility itemsets [25], uncertain sequential patterns [26, 27] and uncertain interesting patterns [28]. In 2017, Lee and Yun [22] have proposed a very efficient structure named CUP_List for rapidly mining UFPs and applied this in the LUNA algorithm. Their experimental results showed that LUNA outperforms the state-of-the art approaches.

Besides, the traditional approaches for mining FPs provide many patterns that reduces the effectiveness of intelligent systems. These systems have to find all the patterns (mining phase) and then rank these to select a limited number of patterns (ranking phase) by themselves. These two phases make the systems consume more time and resources, and they may even fail to run due to a huge memory consumption. There are currently several approaches for reducing the number of patterns, such as frequent closed patterns (FCPs), frequent maximal patterns (FMPs), top-*k* FPs, and top-rank-*k* FPs, with numerous works related to mining top-*k* and top-rank-*k* patterns [29–32] from precise databases.

In this paper, an efficient approach for mining top-*k* UFPs from uncertain databases is proposed. This approach helps the intelligent systems combine the mining and ranking phases, thus overcoming the above-mentioned issues. The main contributions of this work are as follows. [28] We apply the concept of top-*k* to the problem of mining UFPs from uncertain databases. [33] We then propose effective threshold raising strategies to enhance the mining time and reduce the memory usage. [21] Finally, TUFP, an efficient algorithm, was proposed for mining top-*k* UFPs from uncertain databases.

The remainder of the paper is organized as follows. Section 2 presents a survey of related works, including those on mining FPs, mining top-*k* patterns and mining patterns in uncertain databases. Section 3 summarizes the preliminaries and problem statement. Next, in section 4, TUFP algorithm for mining top-*k* UFPs will be proposed. Experimental results in section 5 are presented comprehensively to compare the number of generated candidates, mining time, memory usage as well as scalability in runtime among the proposed algorithm and the state-of-the-art algorithm with optimal parameter for mining top-*k* UFPs. Finally, section 6 summarizes the results and gives some future research problems.

2 Related works

Mining FPs, mining top-*k* and top-rank-*k* patterns as well as mining patterns on uncertain databases were summarized in this section. Subsection 2.1 introduces three approaches for mining FPs. Then, this section presents some approaches for reducing the number of FPs, such as FCPs and FMPs. In subsection 2.2, we review some works on mining top-*k* and top-rank-*k* patterns, including FPs, FCPs, erasable patterns (EPs) and others. Subsection 2.3 then surveys some works on mining patterns from uncertain databases, such as UFPs, uncertain high-utility patterns, and uncertain sequential patterns.

2.1 Mining frequent patterns

There are currently many algorithms which effectively mine FPs on precise databases. There are three main categories, as follows. [28] The first method uses a candidate generate-and-test strategy with some examples such as Apriori [34] and BitTableFI [19]. Firstly, they generate frequent 1-patterns and then used them to generate candidate 2-patterns, and so on. Note that only candidates which have supports greater than or equal to the user given threshold will be included in the results and the next rounds. [33] The second category adopts a divide-and-conquer strategy. Methods in this group compress the database into a tree and discover FPs from this tree by using a divide-and-conquer strategy, with FP-Growth [35] being examples of such algorithms. [21] The third category compresses the database by using vertical data formats (where each item has the set of transaction identifiers containing this item) and mine FPs by using divide-and-conquer strategy. dEclat [3], Index-BitTableFI [36], and Node-list-based method [20, 37, 38] are some examples of this kind of approach.

In reality, a large number of FPs are discovered by the traditional approaches, which makes the results of intelligent systems difficult to interpret and discover rules from, as well as needing a long time to execute. Some representations of FPs that reduces the size of the collection of patterns as well as rules are considered, including FCPs [3], FMPs [39, 40], and top-rank-*k* FPs [41–43]. FCPs are a condensed representation of FPs in which an FCP is an FP having no proper supersets that have the same support. In a similar way, FMPs are also a condensed representation of FPs. However, an FP is said to be an FMP if this pattern is frequent and no superset of this pattern is frequent. Due to the constraints of FMPs and FCPs, the number of FMPs is smaller than that of FCPs.

2.2 Mining top-*k* and top-rank-*k* patterns

Mining top-*k* FPs and top-rank-*k* FPs both adopt a different approach, combining the mining and ranking phases into one. While the results of mining top-*k* FPs is *k* FPs, those of mining top-rank-*k* FPs is *k* ranks with *l* FPs ($l \geq k$). Mining top-*k* and top-rank-*k* patterns helps intelligent systems select the top *k* rank patterns and top *k* patterns

without finding all patterns. Because this approach does not find all the patterns, mining top-*k* and top-rank-*k* patterns is more effective than traditional methods for mining patterns. There are now several methods for mining top-*k* and top-rank-*k* patterns, and these are summarized as follows.

For the problem of mining top-rank-*k* FPs, Deng [42] proposed an efficient method using a Node-list structure named NTK. The Node-list structure improves the effectiveness of the mining process of NTK. In 2015, Huynh et al. [43] proposed an extension method of NTK named the *i*NTK algorithm. *i*NTK combines the N-list structure and subsume concept to increase the effectiveness of this algorithm. Later, Dam et al. [41] proposed the BTK algorithm using B-list structure, as well as subsume concept to reduce the volume of search space and mining time. In addition, an early pruning and an effective threshold raising strategies were also used in BTK. Regarding the problem of mining top-rank-*k* FCPs, to date there has only been one work aiming to solve this problem. In this work, Nguyen et al. [44] proposed two efficient algorithms, called TRK-FCI and TRK-FCI-DBV for mining top-rank-*k* FCPs. TRK-FCI uses DCI-Plus algorithm [45] to generate candidate closed itemsets and apply some early pruning techniques to prune candidates. Then, it uses a bit vector structure for the mining process of mining top-rank-k FCIs. TRK-FCI-DBV uses a dynamic bit vector (DBV) structure to enhance mining time and memory usage compared to TRK-FCI. TRK-FCI-DBV outperforms TRK-FCI for various databases based on their experiments. Deng [46] firstly proposed the VM algorithm for the problem of mining top-rank-*k* EPs using PID_List structure which structure makes the algorithm slow and requires a lot of memory usage. For this reason, Le et al. [47] presented the dPID_List structure, pruning strategies and the subsume concept to enhance the performance. However, the above approaches only support for precise databases. Therefore, we cannot transfer these ideas for the problem of mining top-*k* frequent patterns from uncertain databases.

For mining the top-*k* high utility patterns, Ryang and Yun [31] proposed effective threshold raising strategies for mining top-*k* high utility patterns. Tseng et al. [32] then proposed two algorithms including TKU (mining Top-K Utility itemsets) and TKO (mining Top-K utility itemsets in One phase). In these algorithms, the user does not set *min_util* parameter. In 2016, Duong et al. [29] proposed the most recent algorithm for mining top-*k* high utility itemsets. In which, the novel threshold raising and pruning strategies were proposed to enhance the performance. Next, Dawar et al. [48] developed an approach for mining top-*k* high utility itemsets from data stream using the sliding window model. Besides, Petitjean et al. [30] proposed an approach for mining top-*k* sequential patterns under leverage.

2.3 Mining patterns from uncertain databases

Since the concept of uncertain databases was first proposed, there have been many works for mining various patterns from such databases ([22, 28]; Lee & Yun, 2015; [24–27]). For sequential patterns, Liu [26] proposed a method for mining uncertain sequential patterns and then Palacios et al. [27] applied the uncertain sequential pattern mining approach to aero-engine condition monitoring with uncertain health data. Next, Lin et al. proposed two methods for mining high utility itemsets [25] and a new method for mining weighted frequent itemsets [24] from uncertain databases. Bui et al. [49] then extended the work of Lin et al. [25] to mine high-utility closed itemsets from uncertain databases. Ahmed et al. [28] proposed a method for mining weighted uncertain interesting patterns from uncertain databases. In addition, in 2015, Lee et al. [23] proposed an uncertainty-based approach for mining UFPs from uncertain data with items of different importance. However, there is no special data structures like FP-Trees for frequent pattern mining which were used for uncertain databases. Therefore, Lee and Yun [22] proposed a very efficient structure named CUP_List, along with the LUNA algorithm, for mining UFPs from uncertain databases. LUNA outperforms state-of-the-art approaches for mining UFPs.

3 Preliminaries and problem statement

The preliminaries about the uncertain databases and how to calculate the expected support of a pattern are presented in subsection 3.1. Then, the problem statement of mining top-*k* UFPs is introduced in subsection 3.2. Finally, we summarize the efficient structure [22] named CUP-Lists (Conditional Uncertain Probability-List) for mining UFPs from uncertain databases in subsection 3.3.

3.1 Preliminaries

Firstly, let D be a given database that consists of N transactions and $I = \{i_1, i_2, \ldots, i_m\}$ be a set of distinct items in D. In the early days of this field of research, such databases were usually

Table 1 Example of an uncertain database

TID	A	B	C	D	E	F	G	H
1	1.0	–	0.9	0.6	–	–	–	–
2	0.9	0.9	0.7	0.6	0.4	–	–	–
3	–	0.5	0.8	0.9	–	0.2	0.4	–
4	–	–	0.9	–	0.1	0.5	–	0.8
5	0.4	0.5	0.9	0.3	–	–	0.3	0.3
6	–	–	–	0.9	0.1	0.6	–	0.3
7	0.9	0.7	0.4	0.6	–	0.9	–	–

Table 2 The top-10 of UFPs

Rank	UFP	*expSup*
1	*C*	4.6
2	*D*	3.9
3	*A*	3.2
4	*B*	2.6
5	*CA*	2.25
6	*F*	2.2
7	*CD*	2.19
8	*DA*	1.8
9	*CB*	1.76
10	*AB*	1.64

precise, such as shopper basket data, in which the contents of the databases are exactly known. However, uncertain data can be found in various real-life applications, in which the presence or absence of item i in transaction T was not be certain. The existential probability $P(i, T)$ indicates the likelihood of i being present in T which ranges from 0 (indicating that i has an insignificantly low chance to be present in T) to a value of 1 (indicating that i is definitely present in T). The existential probability values for each item can be different for each transaction. Table 1 shows an example of an uncertain database.

Definition 1 (expected support of a pattern) The expected support of a pattern X, $expSup(X)$, is defined as the sum of the expected probabilities of the presence of X in each of the transactions as follows:

$$expSup(X) = \sum_{T \in D_X} p(X, T) \quad (1)$$

where

D_X is the set of transactions in D whose potential items contain X. In which, potential items of a transaction are the set of items which has non-zero existential probability values in this transaction;

$p(X, T)$ is the existential probability of X for transaction T computed as:

$$p(X, T) = \prod_{i \in X} p(i, T) \quad (2)$$

Therefore, Eq. (1) can be rewritten as follows:

$$expSup(X) = \sum_{T \in D_X} \left(\left(\prod_{i \in X} p(i, T) \right) \right) \quad (3)$$

In real-life applications, the existential probability value of each item in a transaction is from 0 to 1. However, these values often have values closer to 1. Therefore, expSup(X) will be quite big value.

Example 1 Consider the example in Table 1, we have $expSup(A) = 1.0 + 0.9 + 0.4 + 0.9 = 3.2$ and $expSup(B) = 0.9 + 0.5 + 0.5 + 0.7 = 2.6$. Let $X = \{A, B\}$, so, $expSup(X) = (0.9*0.9) + (0.4*0.5) + (0.9*0.7) = 1.64$.

3.2 Problem statement

Definition 2 (top-*k* UFPs) An UFP X is in top-k UFPs (T^k) in D if there is not more than (k-1) UFPs whose expected support is higher than that of X. The minimum expected support to retrieve T^k will be

$$expSup_{\min}(T^k) = \min\{expSup\}(X) \mid X \in (T^k) \quad (4)$$

In other words, the problem of mining top-k UFPs from an uncertain transaction database is as follows. Given an uncertain database D and a threshold k, the problem of mining top-k UFPs is the task of finding all UFPs with k elements, and there is no

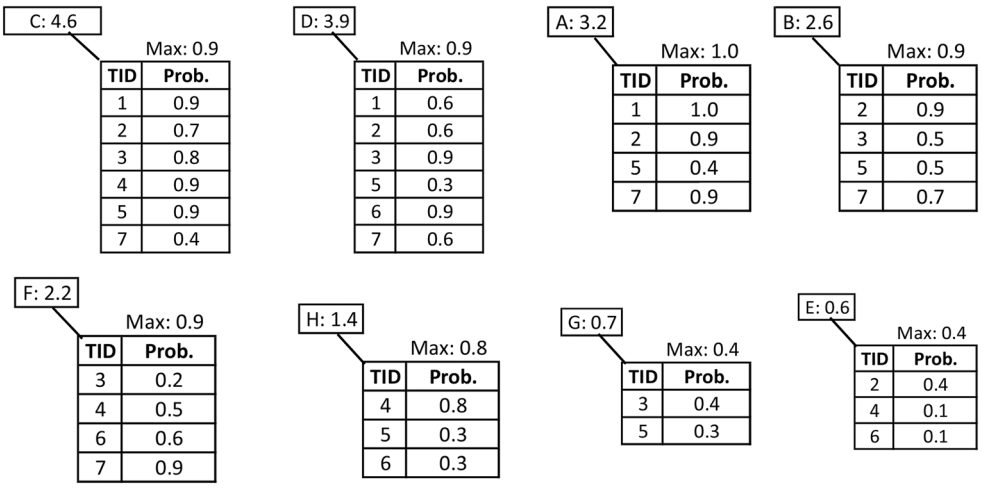

Fig. 1 CUP-Lists of all items in D

UFP which has the expected support greater than the minimal expected support in the result.

Example 2 Consider the example database (*D*) in Table 1, let *k* be 10, Table 2 shows all UFPs belonging to the top-10 UFPs from *D*.

3.3 CUP-lists

In 2017, Lee and Yun [22] proposed CUP-Lists for mining UFPs from uncertain databases. In this paper, we also use this structure for mining top-*k* UFPs. This structure is summarized as follows.

A CUP-List includes a pattern name, its expected support value (*expSup*), a set of tuples namely TEP-List. Each tuple in TEP-List consists of TID information of the transactions containing the pattern, and the corresponding existential probability. Besides, to improve the algorithm's performance, the author proposed the following strategy. Each CUP-List associated with an item has additional information called *Max* to store the maximum existential probability value in TEP-List. Let $Max(Y)$ be the maximum value among the existential probabilities that Y can have in database. Then, the overestimated *expSup* of XY is calculated as $expSup(X) * Max(Y)$. This value is the maximum *expSup* value that XY can have. Therefore, if the value is not smaller than the threshold, XY becomes a potential uncertain frequent pattern; otherwise, the algorithm can prune this pattern. This is the overestimation method for reducing the search space and redundant mining operations [22].

For the example database in Table 1, Fig. 1 shows the CUP-Lists associated with items in *D*. Given two UP-Lists, X and Y with the same prefix, a CUP-List for them is constructed as follows. First, the pattern name of the new CUP-List is $i(X) \cup i(Y)$ and its expected support value is determined by Eq. (3),

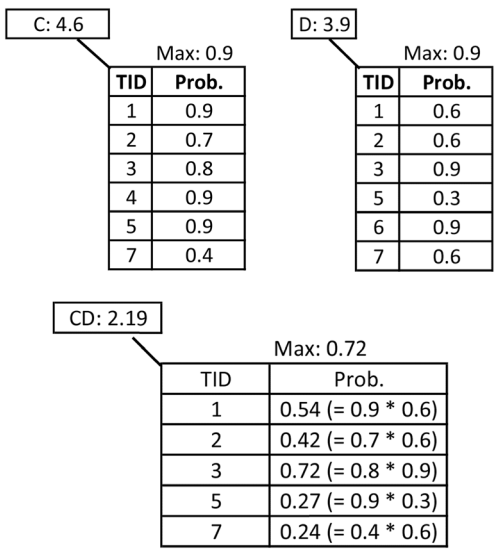

Fig. 2 Construction process of the CUP-List associated with *CD*

Table 3 The top-*6* of UFPs of the first step

No	UFP	ω
1	*C*	4.6
2	*D*	3.9
3	*A*	3.2
4	*B*	2.6
5	*F*	2.2
6	*H*	1.4

where the tuple sets stored in the new CUP-Lists are used to calculate the expected support effectively without additional database scans.

Example 3 Figure 2 shows the construction process of the CUP-List associated with itemset *CD* which was combined from *C* and *D*. First, we determine the TEP-list associated with *CD* by intersecting two TEP-lists associated with *C* and *D*, which is {1, 2, 3, 5, 7}. The algorithm will then calculate the existential probability value of each TID. For instance, the existential probability value of TID = 1 is 0.54 (= 0.9 * 0.6). The results of this process are shown in Fig. 2.

4 Mining top-*k* UFPs

4.1 The effective threshold raising strategies

In this section, two strategies to reduce the runtime will be proposed, as follows.

Strategy 1 (raising the threshold) This strategy adopts a top-*k* array T^k to maintain the current top-*k* UFPs, where UFPs are sorted by ascending order of the expected support (*expSup*). The last element in T^k is always $expSup_{\min}(T^k)$, the smallest expected support in T^k. Therefore, when T^k has *k* elements, the threshold (ξ) will be updated to $expSup_{\min}(T^k)$, the expected support of the last element in T^k. Therefore, when the algorithm inserts new UFPs to T^k and T^k has *k* elements, TUFP will remove the last element in the list and update the threshold to the expected support of the new last element in T^k.

Table 4 The top-*6* of UFPs

No	UFP	ω
1	*C*	4.6
2	*D*	3.9
3	*A*	3.2
4	*B*	2.6
5	*CA*	2.25
6	*F*	2.2

Table 5 Features of databases used in the experiments

Databases	# of Trans.	# of Items	Avg. Trans. size
BMS-POS	515,597	1657	6.5
Chainstore	1,112,949	46,086	43.0
Chess	3196	75	37
Foodmart	4141	1559	4.4
Retail	88,162	16,470	10.3
T10I4D100K	100,000	870	10

Strategy 2 (the pruning strategy by raising the threshold) The threshold variable is increased by **Strategy 1**. During the **TUFP_Search** procedure, the algorithm will not create the candidate of two UFPs C_u and C_v with $expSup(C_u) < \xi$ or $expSup(C_v) < \xi$. This strategy will help the algorithm to reduce the search space and speed up the runtime.

4.2 The proposed algorithm

The main steps of the proposed algorithm named TUFP (Alg. 1) are as follows:

1. Find all 1-patterns (U) in the database with their CUP-Lists. Then, TUFP will sort U in the expected support ascending order.
2. Insert U into the results.
3. Call the **TUFP_Search** procedure to carry out a divide-and-conquer strategy. Firstly, with k-patterns, the algorithm uses the first element to combine it with the remaining elements in this level to create the $(k+1)$-patterns. For elements, whose expected support is smaller than or equal to the threshold, the algorithm will ignore it. Otherwise, the algorithm will: (i) add them to the final results; (ii) combine them to create $(k+2)$-patterns. This strategy was used until all patterns are considered.

4.3 An example

To illustrate the TUFP algorithm, we apply it to the example database in Table 1 with $k = 6$. In the first step, the algorithm will scan this database to compute the CUP-Lists of each item denoted by U and sort U in the expected

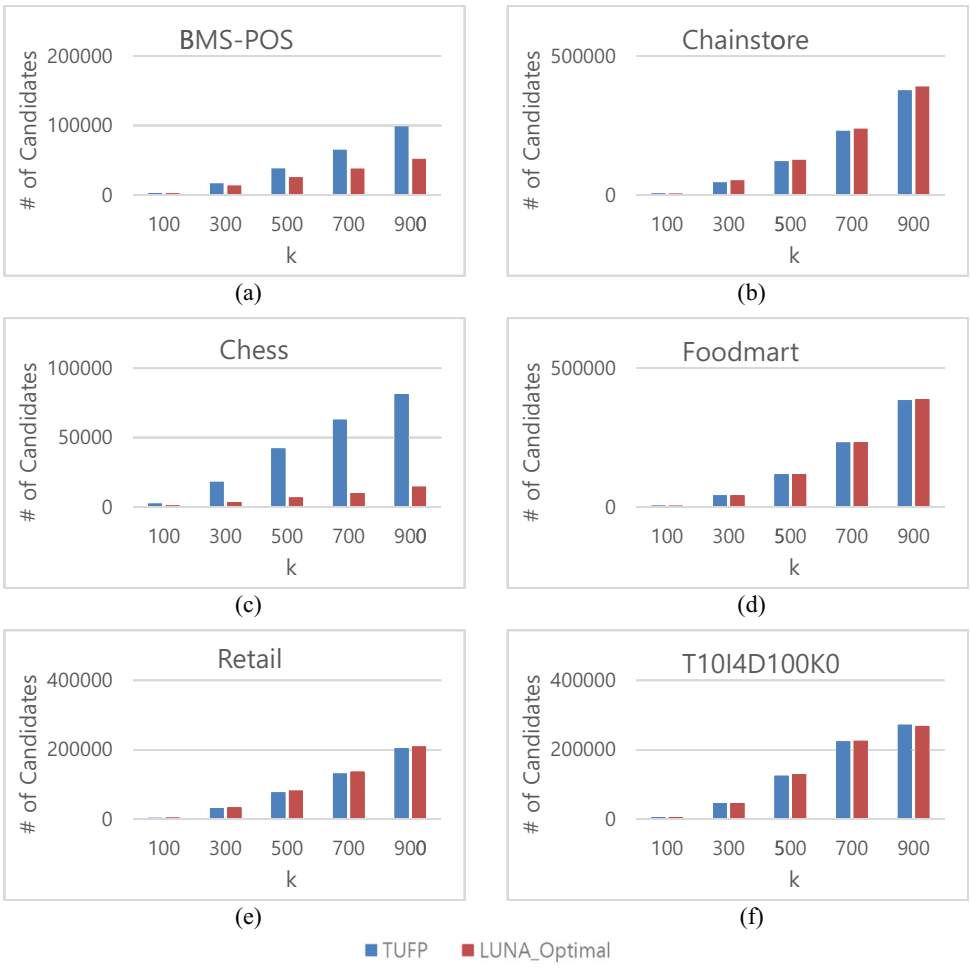

Fig. 3 The number of generated candidates with TUFP and LUNA-Optimal algorithms using the experimental databases with various thresholds k

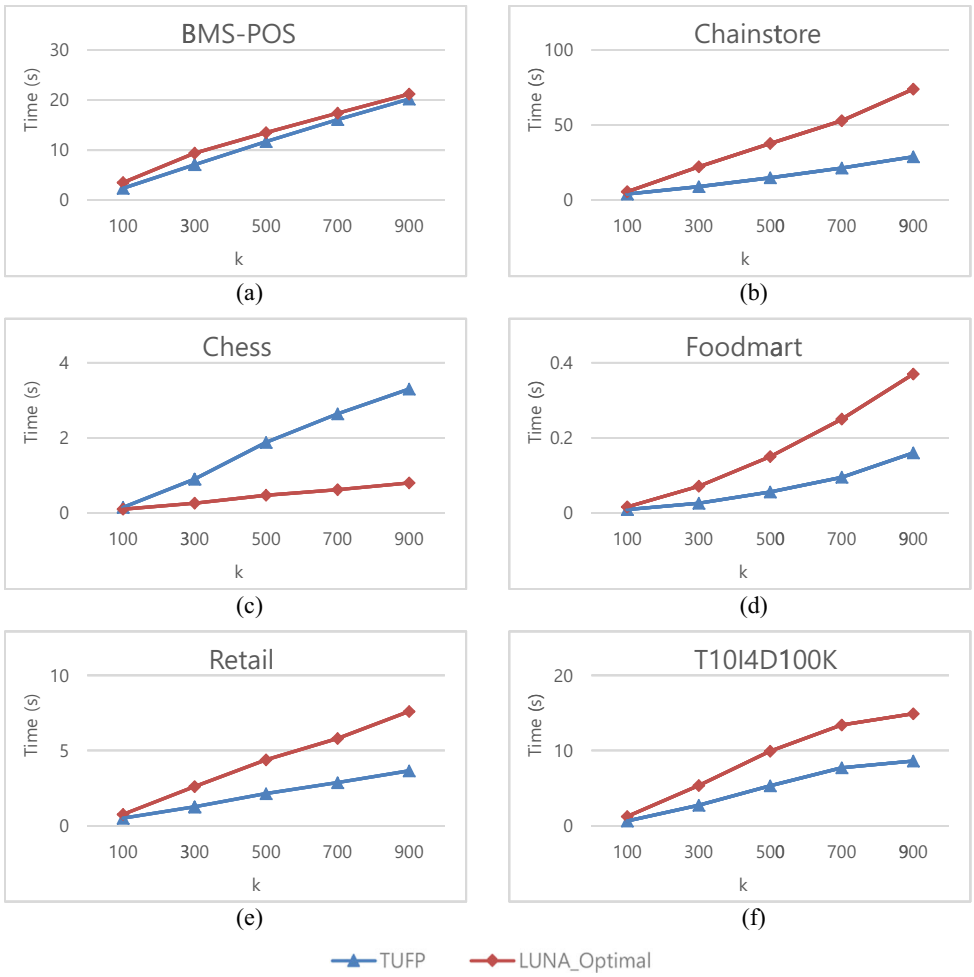

Fig. 4 The mining times of the TUFP and LUNA-Optimal algorithms on experimental databases with various thresholds k

support ascending order, as shown in Fig. 1. Next, the TUFP algorithm will insert the first six items in U into the results and C_k. The results in the first step are shown in Table 3.

In the next step, TUFP uses pattern C to combine with the remaining patterns (D, A, B, F and H) to create the next candidates with 2-itemsets. The pattern $\langle CD, 2.19\rangle$ is first created and inserted to T^k and C_{next}. In addition, H will be removed from T^k and the threshold will be set at 2.19. The algorithm will continue to create $\langle CA, 2.25\rangle$. Because the expected support of CA is larger than the threshold (2.19), then TUFP will insert CA into T^k and C_{next}. CD will then be removed from T^k and the threshold set at 2.2. Then, because the expected supports of CB, CF and CH are smaller than the threshold, TUFP will not insert them into T^k and C_{next}. C_{next} at this step has two elements, CD and CA, and thus TUFP will call **TUFP_Search** to find more patterns. However, the expected support of CDA is smaller than the threshold, and so TUFP stops the branch of C here. The results of this step are shown in Table 4.

In the next steps, the algorithm will start from D, A, B, F and H, respectively, to combine with the remaining patterns. However, there is no pattern which has the expected support larger than the threshold. Therefore, the results in Table 4 are the results of the top six UFPs for the example database.

5 Performance evaluation

All the experiments in this section were performed on a computer with an Intel Core i7–3770 3.4-GHz CPU and 16 GB of RAM. The operating system is Microsoft Windows 10. All algorithms were implemented in C# and. Net Framework (version 4.5.50709) in Microsoft Visual Studio 2012.

The experimental algorithms will perform on following databases: BMS-POS, Chainstore, Chess, Foodmart, Retail and T10I4D100K. These are well-known databases, frequently used to test performance in data mining research and randomly selected. The characteristics of the experimental databases are shown in Table 5.

Because there is no algorithm proposed for mining top-k UFP at this time, we will compare the number of generated candidates, mining time and scalability of TUFP and LUNA [22] to show the effectiveness of TUFP in relatively. LUNA is not originally developed for mining top-k UFPs and cannot be

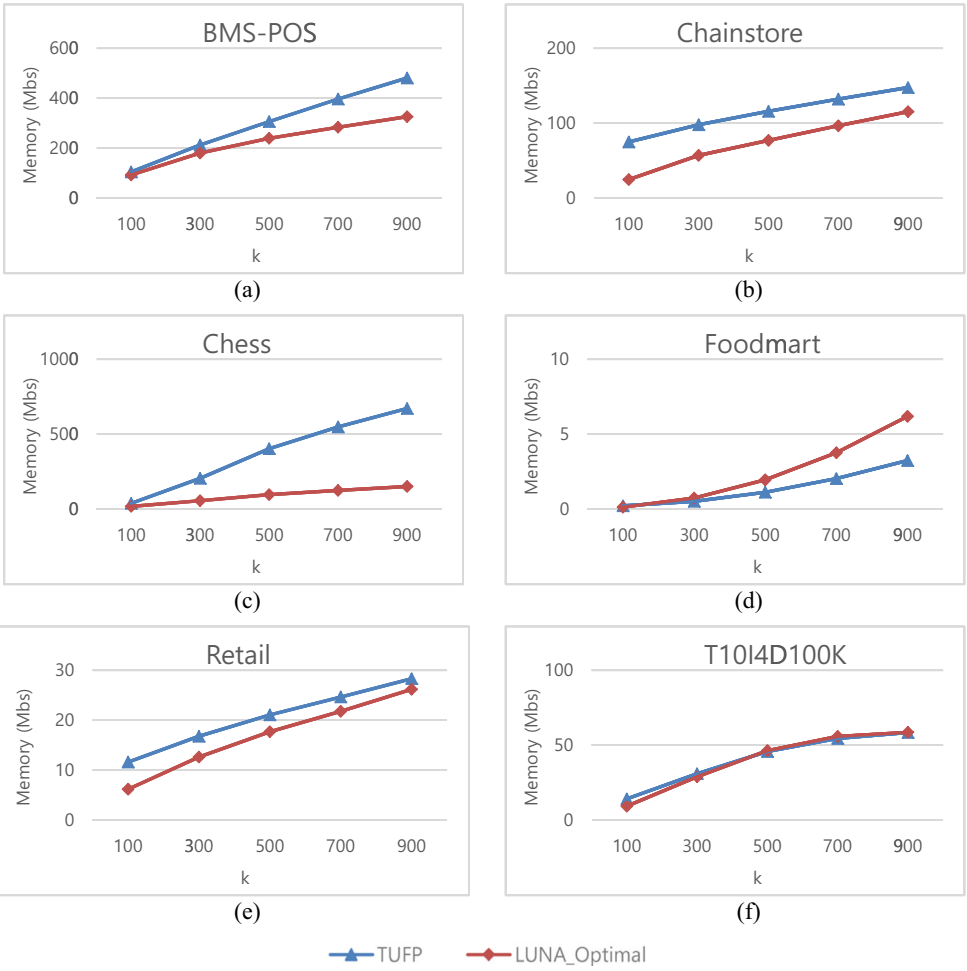

Fig. 5 Memory use of TUFP and LUNA-Optimal algorithms with the experimental databases and various thresholds k

compared on performance directly with TUFP algorithm. To compare their performance, we assume that the users can choose the optimal parameters i.e. minimum support for LUNA to give the same number of patterns as TUFP (denoted as LUNA-Optimal). In the below experiments, we do not consider the cost to obtain the optimal threshold.

5.1 The number of generated candidates

In this section, we compare the number of generated candidates between LUNA-Optimal and TUFP. For the sparse databases, such as Chainstore, Foodmart, Retail and T10I4D100K, the number of generated candidates of TUFP is nearly equal to that of LUNA-Optimal (see Fig. 3b, d, e and f), due to the effective threshold raising strategies. In more details, with $k = 900$ on the Chainstore database, the number of candidates generated by TUFP is 376,688 while that of LUNA-Optimal is 390,348. With $k = 900$ on the Retail database, the number of candidates generated by TUFP is 204,125 while that of LUNA-Optimal is 209,205.

Meanwhile, for the dense databases, such as BMS-POS and Chess, the number of generated candidates with TUFP is much larger than with LUNA-Optimal (see Fig. 3a, and c). Specifically, with $k = 900$ and the BMS-POS database, the number of generated candidates with TUFP is 98,655, while with LUNA-Optimal it is 51,930. With $k = 900$ for the Chess database, the number of generated candidates with TUFP is 81,485, while with LUNA-Optimal it is 14,527.

In general, TUFP is nearly equal to LUNA-Optimal for the sparse databases and is relatively weak for the dense databases in terms of the number of generated candidates as well as the memory usage which is shown in Subsection 5.3.

5.2 Mining time

Figure 4 shows the mining time of TUFP and LUNA-Optimal algorithms on six experimental databases with various thresholds k. For sparse databases like Chainstore, Foodmart, Retail and T10I4D100K, as shown in Figs. 4a, d, e and f, the mining time of TUFP is significantly less than that of LUNA-Optimal. Especially with large threshold k, the time gap between the mining time of TUFP and LUNA-Optimal become larger. For BMS-POS, the mining time of LUNA-Optimal and

TUFP are nearly same in Fig. 4a while LUNA-Optimal is significantly better than TUFP for Chess in Figs. 4c.

5.3 Memory usage

Figure 5 shows the memory consumption of LUNA-Optimal is little bit better than TUFP for BMS-POS (Fig. 5a), Chainstore (Fig. 5b), Foodmart (Fig. 5d), Retail (Fig. e) and T10I4D100K (Fig. 5f). However, the memory usage of LUNA-Optimal is significantly better than TUFP for Chess (Fig. 5c).

5.4 Scalability analysis

The scalability experiments on various numbers of transactions for the largest database, Chainstore, with different k values were conducted. It is very easy to realize that when k values are small in Fig. 6a, then the time differences between TUFP and LUNA-Optimal are also small. However, when the users increase k to 500, 700, and 900 in Figs. 6b, c and d, the time differences become larger. Therefore, TUFP is better than LUNA-Optimal in term of scalability on runtimes.

6 Discussion

With sparse datasets such as Chainstore, Foodmart, Retail and T10I4D100K, TUFP is more effective than LUNA-Optimal in terms of runtime, and nearly equal to LUNA-Optimal on memory usage. In contrast, with dense databases (Chess and BMS-POS databases), the runtime and memory usage of TUFP are not better than LUNA-Optimal. However, as we discussed in the beginning of Section 5, the users have to execute LUNA in several times to choose the optimal value of threshold, which can be very time-consuming. Therefore, TUFP which is little bit worse than LUNA-Optimal for dense databases may be acceptable. In addition, section 5.4 show that TUFP has the good scalability on the mining time. Generally, TUFP is efficient for mining UFPs.

7 Conclusions and future work

We proposed the TUFP method for efficiently mining the top-k UFPs from uncertain databases using effective threshold raising strategy. First, we introduced the problem of mining top-k UFPs from uncertain databases to reduce the number of UFPs by combining the mining and the ranking phases into one. Second, we proposed the effective threshold raising strategy to reduce the search space in the mining process. Next, we used this strategy in TUFP for mining top-k UFPs. Finally, several experiments were conducted on the number of generated candidates, mining time, memory usage and scalability of the TUFP, CUFP-mine and LUNA algorithms to show the effectiveness of the proposed approach.

In the future, we will study several topics related to the problem of mining FPs from uncertain databases, such as mining uncertain frequent closed patterns (UFCPs), uncertain frequent maximal patterns (UFMPs), top-k UFCPs and top-k UFMPs from uncertain databases.

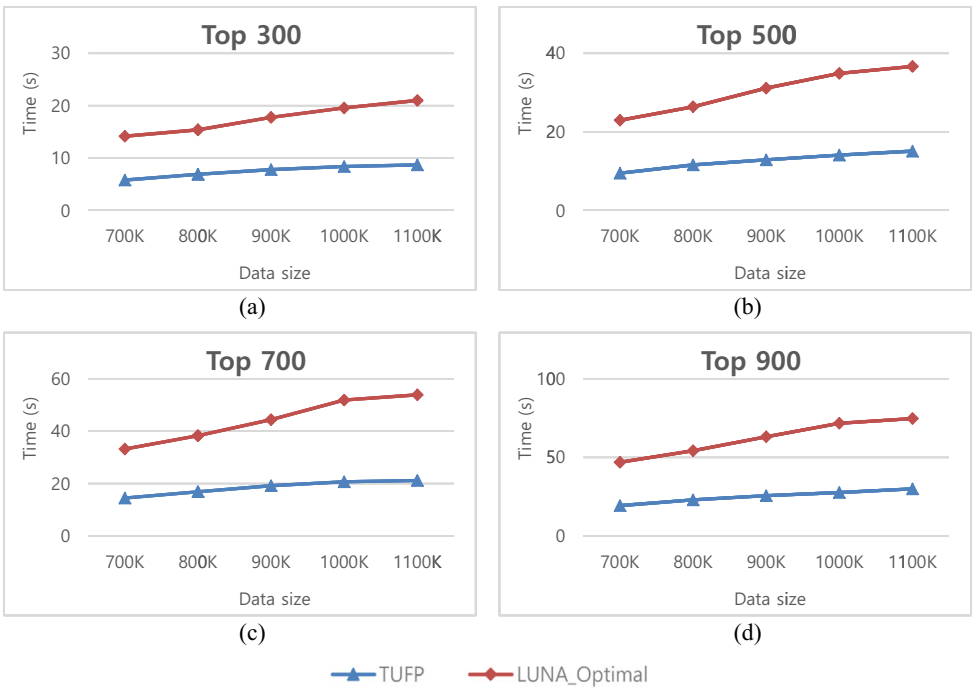

Fig. 6 Scalability on runtimes of the TUFP and LUNA-Optimal algorithms for the Chainstore database with various data sizes and k values

References

1. Agrawal R, Imielinski T, Swami A (1993) Mining association rules between sets of items in large databases. In: SIGMOD'93, pp 207–216
2. Le T, Vo B (2016) The lattice-based approaches for mining association rules: a review. WIREs Data Mining and Knowledge Discovery 6(2):140–151
3. Zaki MJ, Hsiao CJ (2005) Efficient algorithms for mining closed itemsets and their lattice structure. IEEE Trans Knowl Data Eng 17(4):462–478
4. Nanda SJ, Panda G (2015) Design of computationally efficient density-based clustering algorithms. Data Knowl Eng 95:23–38
5. Le T, Lee MY, Park JR, Baik SW (2018a) Oversampling techniques for bankruptcy prediction: novel features from a transaction dataset. Symmetry 10(4):79
6. Le T, Le HS, Vo MT, Lee MY, Baik SW (2018b) A cluster-based boosting algorithm for bankruptcy prediction in a highly imbalanced dataset. Symmetry 10(7):250
7. Le T, Vo B, Fujita H, Nguyen NT, Baik SW (2019a) A fast and accurate approach for bankruptcy forecasting using squared logistics loss with GPU-based extreme gradient boosting. Inf Sci 494:294–310
8. Le T, Vo MT, Vo B, Lee MY, Baik SW (2019b) A hybrid approach using oversampling technique and cost-sensitive learning for bankruptcy prediction. Complexity, ID 8460934
9. Le T, Baik SW (2019) A robust framework for self-care problem identification for children with disability. Symmetry 11(1):89
10. Indurkhya N (2015) Emerging directions in predictive text mining. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 5(4):155–164
11. Nassirtoussi AK, Aghabozorgi SR, The YW, Ngo DCL (2014) Text mining for market prediction: a systematic review. Expert Syst Appl 41(16):7653–7670
12. Ruiz MD, Gómez-Romero J, Molina-Solana M, Ros M, Martín-Bautista MJ (2017) Information fusion from multiple databases using meta-association rules. Int J Approx Reason 80:185–198
13. Vairavasundaram S, Varadharajan V, Vairavasundaram I, Ravi L (2015) Data mining-based tag recommendation system: an overview. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 5(3):87–112
14. Fournier-Viger P, Li Z, Lin JCW, Kiran RU, Fujita H (2019) Efficient algorithms to identify periodic patterns in multiple sequences. Inf Sci 489:205–226
15. Gan W, Lin JCW, Fournier-Viger P, Chao HC, Fujita H (2018) Extracting non-redundant correlated purchase behaviors by utility measure. Knowl-Based Syst 143:30–41
16. Gan W, Lin JCW, Chao HC, Fujita H, Yu PS (2019) Correlated utility-based pattern mining. Inf Sci 504:470–486
17. Yun U, Kim D, Yoon E, Fujita H (2018) Damped window based high average utility pattern mining over data streams. Knowl-Based Syst 144:188–205
18. 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
19. Dong J, Han M (2007) BitTableFI: an efficient mining frequent itemsets algorithm. Knowl-Based Syst 20:329–335
20. Vo B, Le T, Coenen F, Hong TP (2016) Mining frequent itemsets using the N-list and subsume concepts. Int J Mach Learn Cybern 7(2):253–265
21. Aggarwal CC, Li Y, Wang J, Wang J (2009) Frequent pattern mining with uncertain data. In: KDD, pp. 29-38
22. Lee G, Yun U (2017) A new efficient approach for mining uncertain frequent patterns using minimum data structure without false positives. Futur Gener Comput Syst 68:89–110
23. Lee G, Yun U, Ryang H (2015) An uncertainty-based approach: frequent itemset mining from uncertain data with different item importance. Knowl-Based Syst 90:239–256
24. Lin CW, Gan W, Fournier-Viger P, Hong TP, Tseng VS (2016a) Weighted frequent itemset mining over uncertain databases. Appl Intell 44(1):232–250
25. Lin JCW, Gan W, Fournier-Viger P, Hong TP, Tseng VS (2016b) Efficient algorithms for mining high-utility itemsets in uncertain databases. Knowl-Based Syst 96:171–187
26. Liu YH (2015) Mining time-interval univariate uncertain sequential patterns. Data Knowl Eng 100:54–77
27. Palacios AM, Martínez A, Sánchez L, Couso I (2015) Sequential pattern mining applied to aeroengine condition monitoring with uncertain health data. Eng Appl Artif Intell 44:10–24
28. Ahmed AU, Ahmed CF, Samiullah M, Adnan N, Leung CKS (2016) Mining interesting patterns from uncertain databases. Inf Sci 354:60–85
29. Duong QH, Liao B, Fournier-Viger P, Dam TL (2016) An efficient algorithm for mining the top-k high utility itemsets, using novel threshold raising and pruning strategies. Knowl-Based Syst 104:106–122
30. Petitjean F, Li T, Tatti N, Webb GI (2016) Skopus: mining top-k sequential patterns under leverage. Data Min Knowl Disc 30(5): 1086–1111
31. Ryang H, Yun U (2015) Top-k high utility pattern mining with effective threshold raising strategies. Knowl-Based Syst 76:109–126
32. Tseng V, 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
33. Aggarwal CC, Han J (2014) Frequent pattern mining. Springer, ISBN 978-3-319-07820-5
34. Agrawal R., Srikant R.: Fast algorithms for mining association rules. In: VLDB'94, 487–499, 1994
35. Grahne G, Zhu J (2005) Fast algorithms for frequent itemset mining using FP-trees. IEEE Trans Knowl Data Eng 17:1347–1362
36. Song W, Yang B, Xu Z (2008) Index-BitTableFI: an improved algorithm for mining frequent itemsets. Knowl-Based Syst 21:507–513
37. Deng ZH (2016) DiffNodesets: an efficient structure for fast mining frequent itemsets. Appl Soft Comput 41:214–223
38. Deng ZH, Lv SL (2015) PrePost+: an efficient N-lists-based algorithm for mining frequent itemsets via children-parent equivalence pruning. Expert Syst Appl 42(13):5424–5432
39. Fasihy H, Nadimi-Shahraki MH (2018) Incremental mining maximal frequent patterns from univariate uncertain data. Knowl-Based Syst 152:40–50
40. Vo B, Pham S, Le T, Deng ZH (2017) A novel approach for mining maximal frequent patterns. Expert Syst Appl 73:178–186
41. Dam TL, Li K, Fournier-Viger P (2016) An efficient algorithm for mining top-rank-k frequent patterns. Appl Intell 45(1):96–111
42. Deng ZH (2014) Fast mining top-rank-k frequent patterns by using node-lists. Expert Syst Appl 41(4):1763–1768
43. Huynh 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
44. Nguyen LTT, Trinh T, Nguyen NT, Vo B (2017) A method for mining top-rank-k frequent closed itemsets. J Intell Fuzzy Syst 32(2):1297–1305
45. Sahoo J, Das AK, Goswami A (2015) An effective ssociation rule mining scheme using a new generic basis. Knowl Inf Syst 43(1): 127–156
46. Deng ZH (2013) Mining top-rank-k erasable Itemsets by PID_lists. Int J Intell Syst 28(4):366–379
47. 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

48. Dawar S, Sharma V, Goyal V (2017) Mining top-k high-utility itemsets from a data stream under sliding window model. Appl Intell 47(4):1240–1255
49. Bui N, Vo B, Huynh VN, Lin CW, Nguyen LTT (2016) Mining closed high utility itemsets in uncertain databases. In: SoICT, pp. 7–14