Mining Top-K Periodic-Frequent Pattern from Transactional Databases without Support Threshold

Komate Amphawan[1,2,3], Philippe Lenca[2,3], and Athasit Surarerks[1]

[1] Chulalongkorn University, ELITE laboratory, 10330 Bangkok, Thailand
[2] Institut Telecom, Telecom Bretagne, UMR CNRS 3192 Lab-STICC, France
[3] Université européenne de Bretagne
{g48kmp,athasit}@cp.eng.chula.ac.th,
philippe.lenca@telecom-bretagne.eu

Abstract. Temporal periodicity of patterns can be regarded as an important criterion for measuring the interestingness of frequent patterns in several applications. A frequent pattern can be said periodic-frequent if it appears at a regular interval. In this paper, we introduce the problem of mining the top-k periodic frequent patterns i.e. the periodic patterns with the k highest support. An efficient single-pass algorithm using a best-first search strategy without support threshold, called *MTKPP* (Mining Top-K Periodic-frequent Patterns), is proposed. Our experiments show that our proposal is efficient.

1 Introduction

First introduced in [1], frequent pattern mining (also called frequent itemset mining) plays an essential role in many data mining tasks. There are a lot of frequent patterns mining algorithms for a large category of patterns such as association rules [1], correlations [2], sequential patterns [3], dense periodic patterns [4], frequent patterns with maximum length [5], frequent patterns with temporal dependencies [6], etc. Many works have focused on the efficiency of frequent pattern mining by using various techniques such as depth first/breath first search [7], use of trees/other data structures [8], top down/bottom up traversals [9], vertical/horizontal formats [10] and use of constraints [11][12]. Recent surveys may be found in [13] and [14].

However, two main bottlenecks exist: (i) A huge number of patterns are generated and (ii) Most of them are redundant or uninteresting. To tackle these problems, various approaches have been developed.

Frequent-closed pattern mining algorithms have been proposed to reduce redundant patterns [15] and to mine a compact set of frequent patterns which cover all frequent patterns [16]. A recent survey may be found in [17]. While the previous approaches work at the algorithmic level, another strategy is to rank patterns in a post-algorithmic phase with objective measures of interest [18]. A large number of interestingness measures have been proposed. Interesting surveys and comparisons may be found in [19][20] and [21]. At both levels,

B. Papasratorn et al. (Eds.): IAIT 2009, CCIS 55, pp. 18–29, 2009.

constraint-based patterns mining, pushing the constraints using objective measures deeply into the patterns mining process is a very interesting approach [11][12]. This approach uses efficient pruning strategies to discover interesting patterns such as optimal rule mining [22][23]. It is important to notice that most of the previous mentioned works, except mainly [22] and [23], are always subject to the dictatorship of support for the frequent pattern mining step. Avoiding the use of the support has been recognized as a major challenge, such as mining high confidence association without support pruning [24][25][26], and mining rules without support threshold [27][28].

Top-k frequent patterns mining techniques that allow the user to control the number of patterns to be discovered without any support threshold have been proposed in [29]. Recently, [30] proposed a pattern mining approach with a periodic constraint on patterns appearance and a minimum support constraint. As pointed out by the authors, there are several domains to apply periodic-frequent patterns mining: in a retail market, in web site design or web administration, in genetic data analysis, in stock market, etc. Thus the occurrence periodicity plays an important role in discovering interesting frequent patterns in such applications [4][31].

We here focus on these two bottlenecks and propose a new algorithm to discover the top-k periodic-frequent patterns without support threshold. The remainder of this paper is organized as follows: in Section 2 and 3, the problem of mining periodic-frequent patterns and the top-k periodic-frequent patterns are introduced. An efficient single-pass algorithm, named *MTKPP*, is presented in detail in Section 4. Section 5 reports the experimental study. Finally, we conclude this paper in Section 6.

2 From Periodic Patterns to Top-K Periodic Patterns

Tanbeer et al. [30] define a periodic-frequent pattern as a frequent pattern that appears in a database at a regular period (or interval). They define a new periodicity measure for a pattern using the maximum interval at which the same pattern occurs in a database. In addition, they also consider the occurrence frequency. Unfortunately, the traditional frequent patterns mining techniques fail to discover such periodic-frequent patterns because they are only concerned with the occurrence frequency. The authors further propose an efficient tree-based structure, called PF-tree (Periodic-Frequent pattern tree) that enables a pattern growth mining technique to generate the complete set of periodic-frequent patterns in a database [30]. However the user is still in charge of defining the periodicity and the support thresholds.

It is well-known that setting a minimum support threshold is a difficult task for the user. If the threshold is set too small, a large number of patterns will be found which not only consumes more time and space resources, but also burden the users with analyzing the mining results. On the contrary, if the threshold is set too large, there will be very few frequent patterns. This implies that some interesting patterns are hidden because of improper determination of support threshold. That is why many works try to avoid this task, as we mentioned in the introduction.

We here extend the work of [30] and propose a new kind of pattern, namely the top-k periodic-frequent patterns to be discovered in a transactional database. Moreover, we propose an algorithm that discovers the top-k periodic patterns with the highest values of frequency. Our approach has two major advantages. Firstly, it does not need a minimum support threshold. Secondly, our algorithm needs to read the database only once.

3 Problem Definition

We here give some definitions for top-k periodic-frequent pattern mining following the definitions by [30]. We mainly introduce a precise definition of consecutive transaction-ids. This allows us to define an unambiguous definition of the period of a pattern.

Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of $n \geq 1$ literals, called items. A set $X = \{i_j, \ldots, i_k\} \subseteq I$, $1 \leq i_j < i_k \leq n$ is called an itemset (a pattern).

A transaction $t = (tid, Y)$ is a tuple where *tid* represents a transaction-id and $Y \subseteq I$ is an itemset. A transactional database TDB over I is a set of transactions $T = \{t_1, \ldots, t_m\}$, where $m = |TDB|$ is the total number of transactions in TDB. If $X \subseteq Y$, it is said that t contains X or X occurs in t and such transaction-id is denoted as t_j^X, $j \in [1, m]$. Therefore, $T^X = \{t_j^X, \ldots, t_k^X\}$, $j, k \in [1, m]$ and $j < k$ is the set of all ordered transaction-ids (*tids list*) where X occurs in TDB.

Definition 1 (Consecutive *tids* of pattern X). *Let t_j^X and t_k^X, be two tids where X appears, $1 \leq j < k \leq m-1$ and such that there is no transaction t_i that contains X with $j < i < k$. Transactions t_j^X and t_k^X, are then defined as consecutive tids of X.*

Definition 2 (A period of pattern X). *We define a period of the pattern X, denoted as p^X, as the number of transactions between two consecutive tids t_j^X and t_k^X of X:*

$$p^X = t_k^X - t_j^X$$

For simplicity reason we will consider that the first transaction and the last transaction (say, t_f and t_l) in TDB are respectively identified as "null" (i.e., $t_f = 0$) and t_m (i.e., $t_l = t_m$) as in [30]. For instance, from Table 1 the set of transactions where pattern ab appears is $T^{ab} = \{1, 4, 6, 7, 8, 11, 12\}$. Therefore, the periods for this pattern are $1(= 1 - t_f), 3(= 4 - 1), 2(= 6 - 4), 1(= 7 - 6), 1(= 8 - 7), 3(= 11 - 8), 1(= 12 - 11)$ and $0(= t_l - 12)$, where $t_f = 0$ and $t_l = 12$. These periods are then helpful when we consider the behavior of a pattern. In particular, the largest occurrence period of a pattern provide the upper limit of its periodic occurrence characteristic.

Definition 3 (Periodicity of pattern X). *Let T^X be the set of all tids where X occurs and P^X be the set of all periods of $X(P^X = \{p_1^X, ..., p_r^X\})$, where r is the total number of periods in P^X. Then, the periodicity of X can be denoted as $Per(X) = max(p_1^X, \ldots, p_r^X)$.*

For example, in the TDB of Table 1, $P^{ab} = \{1, 3, 2, 1, 1, 3, 1, 0\}$ and $Per(ab) = 3$.

Definition 4 (Support of pattern X). *The number of transactions in a TDB that contains $X(|T^X|)$ is called the support of X and denoted $Sup(X)$.*

For example, the support of pattern ab of Table 1 is $Sup(ab) = |T^{ab}| = 7$.

Definition 5 (Periodic-frequent pattern). *A pattern X is called a periodic-frequent pattern if it satisfies both of the following constraints: (i) its periodicity is no greater than a user-given maximum periodicity: $Per(X) \leq \sigma_p \times |TDB|$ and (ii) its support is no less than a user-given minimum support: $Sup(X) \geq \sigma_s \times |TDB|$ where σ_p and σ_s are expressed in percentage of $|TDB|$.*

Therefore, the periodic-frequent pattern mining problem, given σ_p, σ_s and a TDB, is to discover the complete set of periodic-frequent patterns in TDB having periodicity no greater than $\sigma_p \times |TDB|$ and support no less than $\sigma_s \times |TDB|$.

However, as pointed out before it is quite difficult for users to set a definite support threshold if they have no special knowledge in advance. In addition, in some cases, it is natural for user to specify a simple threshold on the amount of periodic-frequent patterns, say the most 100 frequent patterns with periodicity less than 1,000 transactions. It is thus of interest to mine the most frequent k periodic patterns over transactional databases without the minimum support threshold requirement.

We thus propose the following definition of top-k periodic-frequent patterns.

Definition 6 (Top-k periodic-frequent patterns). *Let us sort the periodic patterns (i.e. patterns with periodicity no greater than $\sigma_p \times |TDB|$) by descending support values; let S be the support of the k^{th} periodic pattern in the sorted list. A pattern X is called a top-k periodic-frequent pattern if it satisfies the following constraints: (i) its periodicity is no greater than a user-given maximum periodicity : $Per(X) \leq \sigma_p \times |TDB|$ and (ii) its support is no less than the support of k^{th} pattern in the sorted list: $Sup(X) \geq S$ where σ_p is expressed in percentage of $|TDB|$.*

4 *MTKPP*(Mining Top-*K* Periodic-Frequent Patterns)

In this section, we introduce an efficient single-pass algorithm, called *MTKPP* (Mining Top-K Periodic-frequent Patterns), for mining the top-k periodic patterns with the k highest supports from transactional databases.

Our algorithm adopts a best-first search strategy to quickly find periodic patterns with the highest values of support. *MTKPP* consists of two phases: Top-k list initialization phase and Top-k mining phase. Both of them are based on the use of a top-k list as presented below.

Top-k list structure. Top-k list is a linked-list with a hash table which is used to maintain k periodic-frequent patterns with highest supports. As shown in Fig. 1, each entry in a top-k list consists of 4 fields: item or itemset name (I), total support (s^I), periodicity (p^I) and *tids* list where I occurs (T^I).

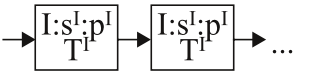

Fig. 1. Top-k list structure

Top-k list initialization phase. To create the top-k list, the database is scanned to obtain all items. At the first occurrence of each item, our algorithm creates a new entry in the top-k list and initializes the support, periodicity and *tids* list. For the other occurrences, *MTKPP* finds the existing entry in the top-k list, using a hash function for efficiency reasons. Then, the values in the entry are updated. After this step, we do not need to scan the database anymore. All items that have periodicity greater than σ_p are removed from the top-k list and the top-k list is sorted in support descending order. Finally, all items that have support less than the support of the k^{th} item in top-k list (s_k) are removed from the top-k list.

Example. Let consider the TDB presented in Table 1. The maximum periodicity threshold σ_p and the number of required results k are 4 and 5 respectively. Figure 2 illustrates the creating of the top-k list from the TDB.

Table 1. Transactional database

tid	items
1	*a b d e*
2	*c d e*
3	*b c f g*
4	*a b d f g*
5	*c e g*
6	*a b c d g*
7	*a b c d*
8	*a b c e*
9	*b c d*
10	*a c e g*
11	*a b f*
12	*a b d g*

With the scan of the first transaction $t_1 = \{a, b, d, e\}$, the entries of the top-k list for items a, b, d and e are initialized as shown in Fig. 2(a). The next transaction ($t_2 = \{c, d, e\}$) initializes a new top-k list entry for item c. It updates the values of support and periodicity for items d and e to $2:1$ and *tids* list to $\{1, 2\}$ (Fig. 2(b)). As shown in Fig. 2(c), after scanning the third transaction ($t_3 = \{b, c, f, g\}$), the periodicity p^b of b changes from 1 to 2. The top-k list after scanning all transactions is given in Fig. 2(d). Then, the item f which has the periodicity $p^f = 7$ greater than $\sigma_p = 4$ is removed from the top-k list. Finally, the top-k list is sorted by support descending order and item e is removed from

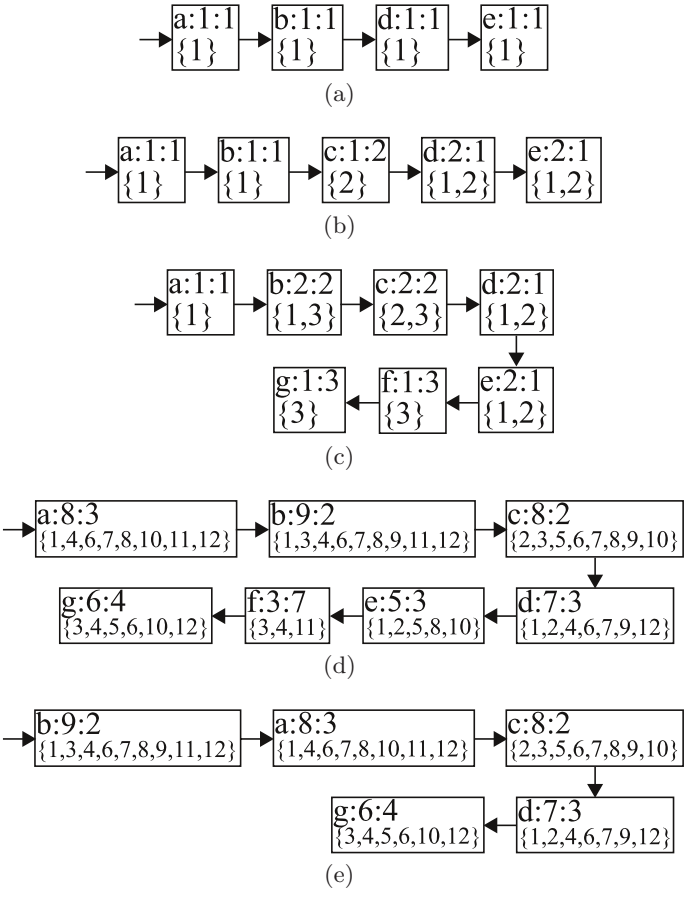

Fig. 2. Top-k list initialization

the top-k list, since the support of $e(s^e = 5)$ is less than support of $g(s^g = 6)$ which is the $k^{th}(5^{th})$ pattern in the top-k list. The top-k list after initialization phase is shown in Fig.2(e).

Top-k mining phase. To mine all top-k period-frequent patterns from the top-k list, a best-first search strategy is adopted to firstly generate the periodic patterns with the highest support. To generate a new periodic pattern, *MTKPP* starts from considering the most frequent patterns to the least frequent patterns in the top-k list. It then combines two elements in the top-k list under the following two constraints: (i) the size of the patterns of both elements must be equal; (ii) both patterns must have the same prefix (i.e. each item from both patterns is the same, except the last item). When both patterns satisfy the constraints, *MTKPP* will intersect the *tids* lists of the two elements in order to find the periodicity, the support and the *tids* list of the new generated periodic pattern. If the periodicity of the new periodic patterns is no greater than σ_p and

the support is greater than the support of the k^{th} pattern in the top-k list, then the newly generated periodic pattern is inserted into the top-k list and the k^{th} pattern will be removed from the top-k list. The details of the mining phase are described in Algorithm 1.

Algorithm 1. (*MTKPP* top-k mining)

Input: top-k list, σ_p, k
Output: top-k periodic-frequent patterns
  for each entry i in top-k list **do**
    for each entry j in top-k list $(i < j)$ **do**
      if $|I^i| = |I^j|$ **and** $|I^i_1| = |I^j_1|, |I^i_2| = |I^j_2|, ..., |I^i_{|I^i|-1}| = |I^j_{|I^j|-1}|$ **then**
        $p^{i\cup j} = 0, s^{i\cup j} = 0, T^{i\cup j} = \phi$
        for each tid_x in T^i and tid_y in T^j **do**
          if $tid_x = tid_y$ **then**
            $s^{i\cup j} = s^{i\cup j} + 1$
            $T^{i\cup j} = T^{i\cup j} \cup tid_x$
            $p = tid_x -$ *last tid* in $T^{i\cup j}$
            if $p > p^{i\cup j}$ **then**
              $p^{i\cup j} = p$
              if $p^{i\cup j} > \sigma_p$ **then**
                stop considering $I^{i\cup j}\{$pattern $i \cup j$ has periodicity $> \sigma_p\}$
        if $p^{i\cup j} \leq \sigma_p$ **and** $s^{i\cup j} \geq s_k$ **then**
          insert $I^{i\cup j}$ into top-k list
          remove k^{th} entry from top-k list

Example. *MTKPP* mine the top-k periodic-frequent patterns from the top-k list of Fig. 2(e). Since item b is the first item in the top-k list and it does not have items in the previous sequence, *MTKPP* starts by considering item a and then looks for other items with the same size and same prefix (which are in the previous sequence in the top-k list), item b. Then, b is combined with a and their *tids* lists are intersected to find the support ($s^{ba} = 7$), the periodicity ($p^{ba} = 3$) and the *tids* list ($T^{ba} = \{1, 4, 6, 7, 8, 11, 12\}$) of pattern ba. Since the periodicity of ba is less than $\sigma_p = 4$ and the support of ba is more than $s_k = 6$, ba is inserted in the top-k list and item g (the k^{th} pattern) is removed from the top-k list (Fig. 3). Next, the third element, item c, is considered. There are two elements which are in the previous sequence and have the same prefix as c: b and a. Then, c is combined with b and their *tids* lists are intersected. The *tids* list and the periodicity of cb are $\{3, 6, 7, 8, 9\}$ and 3 respectively. Because the support of cb ($s^{cb} = 5$) is less than the support of $s_k = 7$, the pattern cb is no longer considered. Next, c and a are combined and their *tids* lists are intersected. The *tids* list of ca is then $\{6, 7, 8, 10\}$. Since the periodicity of $ca(p^{ca} = 6)$ is greater than 4, ca cannot be a periodic pattern. Next, item d and itemset ba are considered in the same manner. When all patterns in the top-k list have been considered, we obtain the top-k periodic-frequent patterns. The final result is shown in Fig 3.

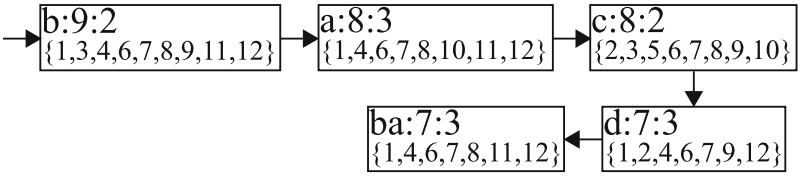

Fig. 3. Top-k frequent patterns

5 Performance Evaluation

In this section, we report our experimental studies to evaluate *MTKPP*. From the best of our knowledge, there is no other existing approach to discover top-k periodic-frequent patterns and we thus only investigate the performances of *MTKPP*.

The *MTKPP* program was implemented in C. The simulations were performed on a 1.6 GHz Intel Xeon with 4 GB main memory on a Linux platform.

5.1 Experimental Setup

We tested our algorithm on one synthetic dataset (*T10I4D100K* [1]) and two real datasets (*Retail* and *Mushroom* [32]).

While *T10I4D100K* and *Retail* are large sparse datasets with $100,000$ and $88,122$ transactions and $1,000$ and $16,469$ distinct items respectively, *Mushroom* is a dense dataset that contains $8,124$ transactions with 119 distinct items.

We conducted several experiments to evaluate the performance of our algorithm. We focused on the time and space costs of our approach, where the time cost refers to the time to initialize and mine the top-k periodic-frequent patterns from the top-k list and the space cost refers to the memory requirement of the top-k list.

We evaluate the performance of our algorithm with various values of k and σ_p: from 100 to 2000 for k, and from 2% to 30% for σ_p.

5.2 Execution Time of *MTKPP*

Figures 4(a) and 4(b) give the processing time of *MTKPP* for *T10I4D100K* and *Retail* datasets. One can observe that the computation time increases as k or σ_p increases. When the value of k increases, *MTKPP* has to find more results, therefore the computation time increases as well. When the value of σ_p increases, it causes the increasing of the number of patterns that have periodicity more than σ_p. Thus, the proposed algorithm *MTKPP* has to consider larger patterns as it cannot prune a huge number of patterns only by using the threshold σ_p.

Figure 4(c) shows the computation time of *MTKPP* on the *Mushroom* dataset. One can see that the computational time increases as k increases but it does not increase when the value of σ_p increases. Since *Mushroom* is dense, the patterns

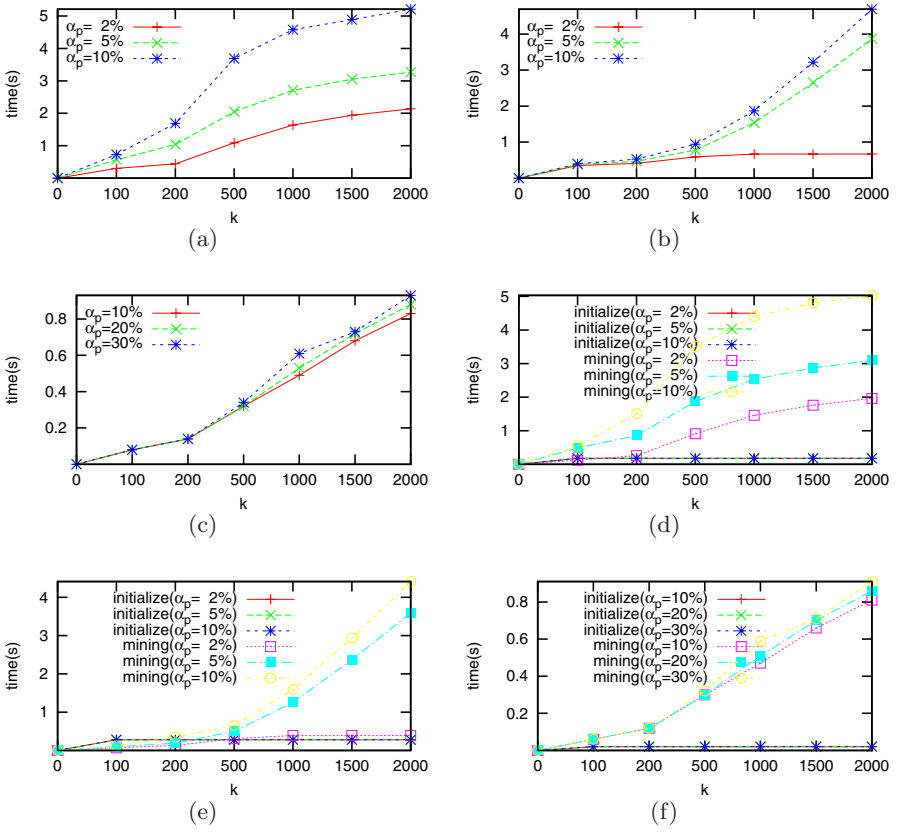

Fig. 4. Computational time of *MTKPP*

in the top-k list occur very frequently and have very low periodicity. Thus, the number of considered patterns is quite stable when the value of σ_p increases.

Figures 4(d), 4(e) and 4(f) give comparisons of the execution time of the top-k list initialization and mining phases. The time to initialize the top-k list is quite unaffected when the values of k and σ_p increase. Indeed, the number of considered transactions and the number of considered items are stable. On the other hand, the time to mine the top-k periodic-frequent patterns increases as k or σ_p increases (Figures 4(a), 4(b) and 4(c)).

5.3 Memory Consumption of *MTKPP*

The variation of memory usage of *MTKPP* with the number of periodic-frequent patterns to be mined, k, is shown in Fig. 5.

From this figure, it is obvious that the memory usage increases as k increases. In fact, the memory usage of *MTKPP* depends on the support of each pattern in the top-k list because *MTKPP* has to keep the *tids* lists of all patterns in the top-k

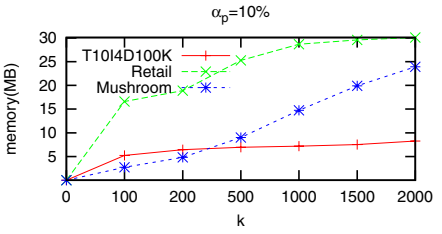

Fig. 5. Memory usage of *MTKPP*

list in order to find the support and the periodicity. For *Mushroom*, the memory usage increases linearly because the supports of patterns in the top-k list do not differ much. For *T10I4D100K* and *Retail*, the memory usage increases slightly as k increases because the supports of patterns in the top-k list are quite different.

6 Conclusion

In this paper, we introduced and studied the problem of mining the top-k periodic-frequent patterns from transactional databases.

An efficient one-pass algorithm, called *MTKPP* (Mining Top-K Periodic-frequent Patterns), is proposed. Since the minimum support to retrieve top-k periodic-frequent patterns cannot be known in advance, a new best-first search strategy is devised to efficiently retrieve the top-k periodic-frequent patterns. It firstly considers the patterns with the highest support and then combines candidates to build the top-k periodic-frequent patterns list.

Our empirical studies, on real and synthetic data, show that our algorithm is efficient and scalable for top-k periodic-frequent pattern mining.

References

1. Agrawal, R., Imielinski, T., Swami, A.N.: Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, pp. 207–216 (1993)
2. Brin, S., Motwani, R., Silverstein, C.: Beyond market baskets: generalizing association rules to correlations. In: ACM SIGMOD/PODS, pp. 265–276 (1997)
3. Agrawal, R., Srikant, R.: Mining sequential patterns. In: International Conference on Data Engineering, pp. 3–14. IEEE Computer Society, Los Alamitos (1995)
4. Engler, J.: Mining periodic patterns in manufacturing test data. In: International Conference IEEE SoutheastCon., pp. 389–395 (2008)
5. Hu, T., Sung, S.Y., Xiong, H., Fu, Q.: Discovery of maximum length frequent itemsets. Inf. Sci. 178(1), 69–87 (2008)
6. Tatavarty, G., Bhatnagar, R., Young, B.: Discovery of temporal dependencies between frequent patterns in multivariate time series. In: Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2007, part of the IEEE Symposium Series on Computational Intelligence 2007, Honolulu, Hawaii, USA, April 1-5, pp. 688–696. IEEE, Los Alamitos (2007)

7. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: VLDB 1994, Proceedings of 20th International Conference on Very Large Data Bases, Santiago de Chile, Chile, September 12-15, pp. 487–499 (1994)
8. Han, J., Pei, J., Yin, Y., Mao, R.: Mining frequent patterns without candidate generation: A frequent-pattern tree approach. Data Min. Knowl. Discov. 8(1), 53–87 (2004)
9. Grahne, G., Zhu, J.: Fast algorithms for frequent itemset mining using fp-trees. IEEE Transactions on Knowledge and Data Engineering 17(10), 1347–1362 (2005)
10. Zaki, M.J., Gouda, K.: Fast vertical mining using diffsets. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24-27, pp. 326–335 (2003)
11. Bonchi, F., Lucchese, C.: Pushing tougher constraints in frequent pattern mining. In: Ho, T.-B., Cheung, D., Liu, H. (eds.) PAKDD 2005. LNCS (LNAI), vol. 3518, pp. 114–124. Springer, Heidelberg (2005)
12. Pei, J., Han, J., Lakshmanan, L.V.S.: Mining frequent item sets with convertible constraints. In: Proceedings of the 17th International Conference on Data Engineering, Heidelberg, Germany, April 2-6, pp. 433–442 (2001)
13. Goethals, B.: Frequent set mining. In: The Data Mining and Knowledge Discovery Handbook, pp. 377–397. Springer, Heidelberg (2005)
14. Han, J., Cheng, H., Xin, D., Yan, X.: Frequent pattern mining: current status and future directions. Data Min. Knowl. Discov. 15(1), 55–86 (2007)
15. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Discovering frequent closed itemsets for association rules. In: Beeri, C., Bruneman, P. (eds.) ICDT 1999. LNCS, vol. 1540, pp. 398–416. Springer, Heidelberg (1999)
16. Pei, J., Han, J., Mao, R.: Closet: An efficient algorithm for mining frequent closed itemsets. In: ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, pp. 21–30 (2000)
17. Yahia, S.B., Hamrouni, T., Nguifo, E.M.: Frequent closed itemset base algorithms: a thorough structural and analytical survey. SIGKDD Explorations 8(1), 93–104 (2006)
18. Hilderman, R.J., Hamilton, H.J.: Applying objective interestingness measures in data mining systems. In: Zighed, D.A., Komorowski, J., Żytkow, J.M. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 432–439. Springer, Heidelberg (2000)
19. Geng, L., Hamilton, H.J.: Interestingness measures for data mining: A survey. ACM Comput. Surv. 38(3), 9 (2006)
20. Lenca, P., Meyer, P., Vaillant, B., Lallich, S.: On selecting interestingness measures for association rules: User oriented description and multiple criteria decision aid. European Journal of Operational Research 184(2), 610–626 (2008)
21. Suzuki, E.: Pitfalls for categorizations of objective interestingness measures for rule discovery. In: Statistical Implicative Analysis, Theory and Applications, vol. 127, pp. 383–395. Springer, Heidelberg (2008)
22. Li, J.: On optimal rule discovery. IEEE Transactions on Knowledge and Data Engineering 18(4), 460–471 (2006)
23. Le Bras, Y., Lenca, P., Lallich, S.: On optimal rule mining: A framework and a necessary and sufficient condition of antimonotonicity. In: Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) PAKDD 2009. LNCS, vol. 5476, pp. 705–712. Springer, Heidelberg (2009)
24. Cohen, E., Datar, M., Fujiwara, S., Gionis, A., Indyk, P., Motwani, R., Ullman, J.D., Yang, C.: Finding interesting associations without support pruning. IEEE Transactions on Knowledge and Data Engineering 13(1), 64–78 (2001)

25. Bhattacharyya, R., Bhattacharyya, B.: High confidence association mining without support pruning. In: Ghosh, A., De, R.K., Pal, S.K. (eds.) PReMI 2007. LNCS, vol. 4815, pp. 332–340. Springer, Heidelberg (2007)
26. Le Bras, Y., Lenca, P., Lallich, S.: Mining interesting rules without support requirement: A general universal existential upward closure property. Information Systems (2010)
27. Li, J., Zhang, X., Dong, G., Ramamohanarao, K., Sun, Q.: Efficient mining of high confidience association rules without support thresholds. In: Żytkow, J.M., Rauch, J. (eds.) PKDD 1999. LNCS (LNAI), vol. 1704, pp. 406–411. Springer, Heidelberg (1999)
28. Koh, Y.S.: Mining non-coincidental rules without a user defined support threshold. In: Washio, T., Suzuki, E., Ting, K.M., Inokuchi, A. (eds.) PAKDD 2008. LNCS (LNAI), vol. 5012, pp. 910–915. Springer, Heidelberg (2008)
29. Cheung, Y.L., Fu, A.W.C.: Mining frequent itemsets without support threshold: With and without item constraints. IEEE Transactions on Knowledge and Data Engineering 16(9), 1052–1069 (2004)
30. Tanbeer, S.K., Ahmed, C.F., Jeong, B.S., Lee, Y.K.: Discovering periodic-frequent patterns in transactional databases. In: Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) PAKDD 2009. LNCS, vol. 5476, pp. 242–253. Springer, Heidelberg (2009)
31. Laxman, S., Sastry, P.: A survey of temporal data mining. In: Sādhanā, Part 2, vol. 31, pp. 173–198 (2006)
32. Asuncion, A., Newman, D.: UCI machine learning repository (2007)