MINING CLOSED SEQUENTIAL PATTERNS IN LARGE SEQUENCE DATABASES

V. Purushothama Raju[1] and G.P. Saradhi Varma[2]

[1]Research Scholar, Dept. of CSE, Acharya Nagarjuna University
Guntur, A.P., India

[2]Department of Information Technology
S.R.K.R. Engineering College, Bhimavaram, A.P., India

ABSTRACT

Sequential pattern mining is studied widely in the data mining community. Finding sequential patterns is a basic data mining method with broad applications. Closed sequential pattern mining is an important technique among the different types of sequential pattern mining, since it preserves the details of the full pattern set and it is more compact than sequential pattern mining. In this paper, we propose an efficient algorithm CSpan for mining closed sequential patterns. CSpan uses a new pruning method called occurrence checking that allows the early detection of closed sequential patterns during the mining process. Our extensive performance study on various real and synthetic datasets shows that the proposed algorithm CSpan outperforms the CloSpan and a recently proposed algorithm ClaSP by an order of magnitude.

KEYWORDS

Data mining, sequential pattern mining, closed sequential pattern mining, sequence database

1. INTRODUCTION

Sequential pattern mining is an important and active research topic in data mining. Sequential pattern mining was first introduced by Agrawal and Srikant in [1]. Since then several efficient sequential pattern mining algorithms have been proposed to find out sequential patterns.

Sequential pattern mining can be applied to several business and scientific applications including biological sequence analysis, market analysis, discovering web access patterns, mining customer shopping sequences, web click streams, XML query access patterns for caching, block correlations in storage systems, sequences of file block references in operating systems, target marketing, customer retention, feature selection for sequence classification, user behavior analysis, web recommender systems, network intrusion detection, personalization systems and the study of scientific or medical processes.

There is an increasing trend of utilizing sequential pattern mining in source code mining and software specification mining. These tools convert the source code into a sequence database representation, and find the sequential patterns in order to retrieve different information such as copy-pasted code segments, API usage, programming rules and etc.

Consider a book store such as Amazon can find that the customers who have first bought a book ‘‘Introduction to Data Mining” will often purchase another book ‘‘Data Science for Business” in

DOI : 10.5121/ijdms.2015.7103

a later transaction. Store managers can use this information to do recommendation when a user browses the data mining book. Another useful application is to identify block access patterns of disk systems. The block access patterns are useful to predict the blocks that are accessed next, and these blocks can be prefetched into cache to reduce the disk I/O latency.

Sequential pattern mining produces an exponential number of patterns when the database contains long sequences which are expensive in both time and space. The same problem also occurs in itemset and graph mining when the patterns are long. For example, assume the database contains a long sequence $\{(x_1)(x_2)\ldots\ldots(x_{100})\}$ it will generate 2^{100} - 1 frequent subsequences. Some subsequences have the support which is equivalent to the support of the long sequence, which are basically redundant patterns. Therefore, instead of mining the complete set of sequential patterns, it is better to mine closed sequential patterns only. A closed sequential pattern is a sequential pattern which has no supersequence with the same support.

Closed sequential pattern mining extensively reduces the number of patterns produced and it can be utilized to obtain the complete set of sequential patterns. In addition, closed sequential pattern mining algorithms make use of search space pruning techniques and outperform sequential pattern mining algorithms. Closed sequential pattern mining helps users to find more interesting patterns and reduces the burden of users to explore too many patterns.

As an example, consider a sequence database which has 31,000 sequences, when min_sup is 40%, there are 16,204 sequential patterns, where as only 609 patterns are closed sequential patterns. To find the complete set of sequential patterns, an efficient sequential pattern mining algorithm will take about 400 seconds on a 2 GHz Intel dual core PC. Where as a fastest closed sequential pattern mining algorithm takes only 53 seconds to mine all closed sequential patterns on the same machine. Therefore, it is better to mine only closed sequential patterns.

There are two popular algorithms for closed sequential pattern mining: CloSpan[2] and BIDE[3]. CloSpan performs the mining in two stages. In the first stage it produces a closed sequential pattern candidate set and stores it in a prefix sequence lattice. In the second stage it performs post pruning to eliminate non closed sequential patterns. CloSpan works under candidate maintenance-and-test paradigm, hence it is not scalable because a large number of closed sequential pattern candidates will occupy more memory space and lead to huge search space for checking the closure of new patterns, particularly for low support threshold values or long patterns. BIDE mines closed sequential patterns without candidate maintenance. It prunes the search space more deeply and performs closure checking of patterns in an efficient manner. BIDE follows a strict depth-first search order to produce the closed sequential patterns.

In this paper, we propose an efficient algorithm, called CSpan, which makes use of the occurrence checking method that allows early detection of closed sequential patterns during the mining process. Our extensive performance study on various real and synthetic datasets shows that the proposed algorithm CSpan outperforms the CloSpan and a recently proposed algorithm ClaSP[4] by an order of magnitude.

The remaining of this paper is organized as follows. Section 2 discusses the related work. In Section 3, we present the problem definition. In Section 4, we discuss the proposed method for mining closed sequential patterns. Section 5 reports the performance evaluation. Finally, we conclude the work in Section 6.

2. RELATED WORK

Agrawal and Srikant [1] proposed the sequential pattern mining problem. Later a number of efficient algorithms have been developed for sequential pattern mining. These algorithms are categorized into three groups such as Apriori-based methods, pattern-growth methods and vertical format based methods.

Apriori-based methods implement a candidate-generation-and-test strategy. The representative algorithm of these methods is GSP [5], which is an enhancement of AprioriAll algorithm developed by the same authors. Algorithms implementing a candidate-generation-and-test strategy produce a large number of candidate sequences. Counting and pruning such a large set is more expensive. In addition, Apriori-based methods require more number of database scans when there are long patterns.

Pattern-growth methods generally start with a frequent pattern, and grow the frequent pattern when traversing the pattern search space using depth-first search. FreeSpan[6], PrefixSpan [7], CloSpan, BIDE and etc belonging to this type. PrefixSpan finds the frequent-1 sequences in the database and builds projected databases for each sequence. It then searches the projected database to uncover locally frequent items and recursively locates the frequent sequences that contain the item as a prefix. This process is repeated until all frequent sequences are discovered. It uses a pseudo-projection technique for constructing projected databases to improve the performance. PrefixSpan is faster than GSP, because in each iteration it scans only the sequences in a projected database and a projected database is much smaller than the original database.

Vertical format based methods represent a sequence database using vertical format which allows faster computation of support counting. SPADE [8] and SPAM [9] are based on vertical format. SPADE makes use of a vertical id-list format and a join operation between two id-lists to discover frequent sequential patterns. It consumes more memory, since id-lists are several times bigger than the initial database. SPAM constructs a vertical bitmap of the database for efficient candidate generation and support counting. SPAM traverses the lexicographic sequence tree in a depth-first manner. SPAM is faster than SPADE due to the fast bitmap computation and consumes more memory space than SPADE.

Closed itemset mining is used to find closed itemsets in a transaction database. Several algorithms were developed for closed itemset mining. A-Close[10] is the first algorithm developed for mining closed itemsets. It first finds level-wise frequent itemsets using Apriori strategy, and mines all minimal generators. In the second step, it computes the closure of all minimal generators. The performance of A-Close degrades due to the huge cost of the off-line closure calculation.

Closet [11] and Closet+ [12] algorithms use compact FP-Tree data structure. In the first scan of the dataset, frequent single items are identified and in the second scan the pruned transactions are added to the FP-Tree. Closet finds closed itemsets by closure climbing and grows frequent closed itemsets with a depth first browsing of the FP-Tree and recursive conditional FP-Tree projections. It uses subset checking to detect the duplicates. Closet+ is an improvement of Closet and it has an adaptive behavior in order to fit both sparse and dense datasets. Closet+ uses a new upward checking technique to detect duplicates for sparse datasets.

Charm[13] constructs a prefix tree using frequent itemsets and searches the prefix tree using bottom-up depth-first approach. Once a frequent itemset is produced, its tid-list is checked with other itemsets having the same parent. If one tid-list contains another tid-list, the related nodes are

combined because both the itemsets belong to the same equivalence class. Charm uses diff-set technique for sorting itemset tid-lists in each node of the tree.

It is not feasible to adopt the techniques used in closed itemset mining for closed sequential pattern mining, because subsequence testing needs order matching and the search space of sequences is bigger than that of itemsets. Also, sequential pattern mining is generally less efficient than closed sequential pattern mining, particularly in mining long patterns and with low support threshold.

X.Yan et al. developed the CloSpan algorithm for mining closed sequential patterns. CloSpan produces less number of discovered sequences than the traditional methods while preserving the same expressive power. CloSpan can mine long sequences and runs faster than PrefixSpan. CloSpan divides the mining process into two phases. In first phase it generates a candidate set for closed sequential patterns and stores it in a prefix sequence lattice. In second phase it conducts post pruning to eliminate non closed sequences.

CloSpan prunes the nonclosed sequential patterns by using an efficient search space pruning method, known as equivalence of projected databases. Unfortunately, a closed sequential pattern mining algorithm under candidate maintenance-and-test paradigm has rather poor scalability because a large number of closed sequential patterns candidates require more memory and lead to huge search space for the closure checking of new patterns, which is usually the case when the support threshold is low or the patterns are long.

Wang et al. developed the BIDE algorithm. It mines closed sequential patterns without candidate maintenance. It prunes the search space more deeply and performs closure checking of patterns in an efficient manner. BIDE follows a strict depth-first search order to produce the closed sequential patterns. It uses a novel closure checking scheme called BI-Directional Extension. The forward directional extension is used to grow the prefix patterns and also for closure checking of prefix patterns, whereas the backward directional extension is used for both closure checking of a prefix pattern and pruning the search space. It prunes the search space by using the BackScan pruning method. BIDE first applies the BackScan pruning method to check if a prefix sequence can be pruned, if not, computes the number of backward-extension items. Later it finds the number of forward extension items, if there is no backward-extension item or forward-extension item then it outputs closed sequential patterns. Although BIDE does not keep track of any historical closed sequential patterns candidates for a new pattern's closure checking, it is a computational consuming approach since it needs multiple database scans for the bi-direction closure checking and the BackScan pruning.

The COBRA algorithm was developed by Huang et al. [14]. It adopts a bi-phase reduction method for closed sequential pattern mining. It finds closed itemsets for first phase reduction and then finds closed sequential patterns for second phase reduction. COBRA first conducts item extension and then do sequence extension, which eliminates some drawbacks in pattern growth methods. It performs the mining in three stages: (I) Mining Closed Frequent Itemset (II) Database Encoding and (III) Mining Closed Sequential Pattern. COBRA uses closed itemsets to eliminate duplicate item enumeration and to decrease the matching cost for mining locally frequent items. It uses both vertical and horizontal database formats to reduce the search time in the mining process. It uses two pruning methods LayerPruning and ExtPruning. LayerPruning eliminates non closed branches and reduces the costs in pattern checking. ExtPruning checks closure of patterns to eliminate non closed sequential patterns. COBRA requires large memory space than BIDE.

ClaSP mines closed sequential patterns in temporal transaction data. It is inspired on the Spade algorithm that uses vertical database format strategy for generating sequential patterns. ClaSP has

two main phases: The first phase produces Frequent Closed Candidates (FCC) that is kept in main memory; and the second phase performs a post-pruning to remove all non-closed sequences from FCC to finally get exactly frequent closed sequences. It first discovers all frequent 1-sequences in the database and then the method DFS-Pruning is called recursively for all frequent 1-sequences to search the corresponding subtree. FCC is acquired when this process is done for all of the frequent 1-sequences and finally, the algorithm terminates by removing the non-closed sequences in FCC. ClaSP uses the method CheckAvoidable for pruning the search space and outperforms CloSpan.

3. PROBLEM DEFINITION

In this section, we first introduce some preliminary concepts, and then formalize the closed sequential pattern mining problem.

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of all items. A subset of I is called an itemset. A sequence $S=(k_1, k_2, \ldots, k_n)$ ($k_i \subseteq I$) is an ordered list of itemsets. The items in each itemset are sorted in alphabetic order. The length of the sequence is the total number of items in the sequence. A sequence $S_1 = (a_1, a_2, \ldots, a_m)$ is a subsequence of another sequence $S_2 = (b_1, b_2, \ldots, b_n)$, denoted as $S_1 \sqsubseteq S_2$, if there exists integers $1 \leq i_1 < i_2 < \ldots < i_m \leq n$ and $a_1 \subseteq b_{i1}$, $a_2 \subseteq b_{i2}$, . . . , and $a_m \subseteq b_{im}$. We call S_2 as a super-sequence of S_1 and S_2 contains S_1.

A sequence database, $SD=\{S_1, S_2, \ldots, S_n\}$, is a set of sequences and each sequence has an id. The size, $|SD|$, of the sequence database SD is the total number of sequences in the SD. The support of a sequence X in a sequence database SD is the no of sequences in SD which contain X.

Definition 1 (Sequential pattern): Given a minimum support threshold m_sup, a sequence α is a sequential pattern on SD if support (α) is greater than m_sup.

Definition 2 (Closed sequential pattern): A sequential pattern α is a closed sequential pattern if there does not exist a sequential pattern β, such that support (α) = support (β) and $\alpha \sqsubset \beta$.

The problem of closed sequential pattern mining is to find the complete set of closed sequential patterns above a minimum support threshold m_sup for an input sequence database SD. Table 1 shows a sample sequence database. The items in each itemset are sorted in alphabetic order. If m_sup=2 , the closed sequential pattern set contains 3 sequences {**(bc) : 3, (c)(d):2, (a)(bc):2**} and the corresponding sequential pattern set contains 9 sequences. It indicates that closed sequential pattern set contains less no of sequences than sequential pattern set.

Table 1. A sample sequence database

Sid	Sequence
1	(ab)(bc)
2	(bc)(d)(e)
3	(ac)(bcd)

Given a sequence $X=(p_1, \ldots, p_n)$ and an item k, X can be concatenated with k using either itemset extention $X\Delta_i k=(p_1, \ldots, p_n \cup (k))$ or sequence extension $X\Delta_s k=(p_1, \ldots, p_n, (k))$. For example (ab) is an itemset extension of (a) and (a)(b) is a sequence extension of (a).

4. PROPOSED METHOD

In this section, we describe our proposed algorithm CSpan. It uses depth-first search for generating the closed sequential patterns. Since many previously developed algorithms proved that depth-first search based algorithm is more efficient than the breadth-first search based algorithm in mining long patterns.

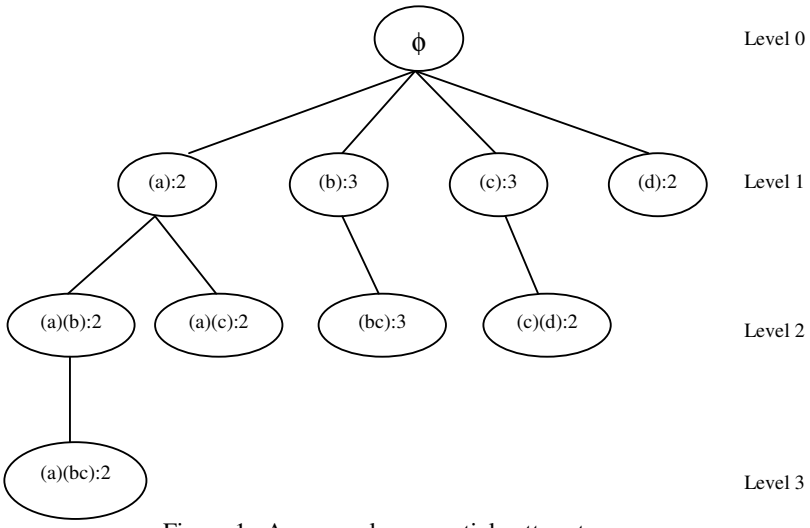

Figure 1. An example sequential pattern tree

The CSpan algorithm uses a sequential pattern tree to generate the closed sequential patterns. The sequential pattern tree is constructed as follows. The root of the tree is labeled with ϕ. Next, all frequent 1-sequences in the database are identified and added to level 1 of the sequential pattern tree. Then, a sequence k at level 1 is extended by appending a frequent item in k's projected database to get its frequent super-pattern at level 2. A sequence can be extended in two ways: sequence extension and itemset extension. In case of sequence extension, the item is added as a new itemset to the sequence. In case of itemset extension, the item is appended to the last itemset in the sequence. This process is repeated for the sequences at the level 2 and above until there are no super-sequences to generate.

Fig. 1 represents the sequential pattern tree for the sample sequence database shown in Table. 1. First the root of the sequential pattern tree is set to ϕ and then all frequent 1- sequences are added as the children of the root at level 1. The number after the sequence indicates the support of the sequence and the minimum support is chosen as 2. The children of a node are arranged lexicographically. Next, the frequent 2-sequences at level 2 are produced by concatenating the frequent 1-sequence with one of the frequent item in its projected database, and similarly the frequent 3-sequences at level 3 are produced by concatenating the frequent 2-sequence with one of the frequent item in its projected database.

A sequence in the sequential pattern tree is considered as a sub-sequence of its child. To locally mine the frequent super-sequences of a certain sequence, we construct the projected database for the sequence and grow the sequence to obtain its frequent super-sequences. We use the pseudo based projection approach of PrefixSpan to create the projected database.

4.1 Occurrence checking

We propose a new pruning method called occurrence checking that allows the early detection of closed sequential patterns.

Lemma 1. Occurrence checking: A sequential pattern X is not closed if a frequent item y exists such that (1) y appears in every sequence of X's projected database and (2) the distance between X and y is identical in every sequence of X's projected database.

Proof. If a frequent item y appears in every sequence of X's projected database and the distance between X and y is identical in every sequence of X's projected database, then we can always discover another frequent sequence containing X and y whose support is equivalent to X's support. Therefore, X cannot be closed.

We demonstrate the occurrence checking scheme as follows. Assume *a* is a frequent sequence. If we locate an item *b* after *a* in every sequence of *a*'s projected database, then we can declare that *a* is not closed since *ab* is its super-sequence with the same support.

4.2 CSpan Algorithm

The CSpan algorithm has two phases. First, it scans the database to determine all frequent 1-sequences. Second, for each frequent k-sequence, it constructs the projected database and locates all frequent items in the projected database to produce its frequent super-sequences at the next level in the sequential pattern tree, where k≥1. During the mining process, it employs occurrence checking for early detection of closed sequential patterns.

Algorithm 1: CSpan
Input: A sequence database SD and minimum support m_sup.
Output: The complete set of closed sequential patterns.
1. S1 = frequent 1-sequences in SD
2. CSP = ϕ
3. for each s in S1 do
4. PD = Projected database of s
5. CS = Generate_patterns (PD, s, m_sup)
6. CSP = CSP $\cup$ CS
7. end for

Algorithm 2: Generate_patterns(PD, s, m_sup)
Input: A projected database PD, sequence s and minimum support m_sup
Output: Closed sequence patterns with prefix s.
1. CS = ϕ
2. P1 = frequent items in PD
3. if support(s) ≥ m_sup and P1≠ ϕ then
4. if (s does not pass the occurrence checking) then
5. if (s is closed w.r.t CS) then
6. CS= CS $\cup$ s
7. end if
8. end if
9. for each p in P1 do
10. PD=Projected database of s Δ_s p
11. CS=CS $\cup$ Generate_patterns(PD, s Δ_s p, m_sup)
12. PD=Projected database of s Δ_i p

13. CS=CS ∪ Generate_patterns(PD, s Δ_i p, m_sup)
14. end for
15. end if
16. return CS

Example: Consider the sample sequence database shown in Table 1 and assume that m_sup = 2. First, we determine all frequent 1-sequences (a), (b), (c) and (d), and perform the occurrence checking on these sequences. All frequent 1- sequences pass the occurrence checking step and therefore they are not closed. Next, we determine the frequent super-sequences of frequent 1-sequence (a) by using the frequent items in its projected database. There are two frequent items in (a)'s projected database, namely, (b) and (c).

We first grow the frequent 1-sequence (a) by appending (b) to it, to get the frequent 2- sequence (a)(b). Then we check whether (a)(b) passes the occurrence checking or not. But, (a)(b) passes the occurrence checking step. Therefore, it is not a closed sequential pattern. Next we grow the sequence (a)(b) by appending a frequent item in its projected database to get the frequent 3-sequence (a)(bc), which does not pass the occurrence checking. Therefore, it is a closed sequential pattern. Since there are no frequent items in (a)(bc)'s projected database, we can't generate its super-sequences and we stop growing it further. By growing the sequential pattern tree in the manner as explained above, we can generate all closed sequential patterns. Finally, we obtain three closed sequential patterns (bc), (c)(d) and (a)(bc).

5. PERFORMANCE EVALUATION

In this section, we perform a thorough performance evaluation of our proposed CSpan algorithm on both real and synthetic data sets with various kinds of sizes and data distributions, and we compare CSpan with CloSpan and a recently proposed algorithm ClaSP.

All experiments were conducted on a 2GHz Intel Core2 Duo processor PC with 1GB main memory running Microsoft Windows XP. The algorithms were implemented in Java and were executed using different support values.

In our experiments we used a real world dataset BMS WebView of KDD CUP 2000[15] and two synthetic datasets. BMS WebView is a click stream data from an e-commerce web store named Gazelle and it has been used widely to assess the performance of frequent pattern mining. This dataset contains sequences of 59601 customers with a total of 146000 purchases in 497 distinct product categories. The average length of sequences is 2.42 items with a standard deviation of 3.22. In this dataset, there are some long sequences. For example, 318 sequences contain more than 20 items. The characteristics of the BMS WebView dataset are given in Table 2. We generated the synthetic datasets using SPMF[16] framework. The characteristics of the two synthetic datasets are given in Table 3.

Table 2. Characteristics of the BMS WebView dataset

S. No.	Characteristic	Value
1	No of sequences	59601
2	No of distinct items	497
3	Average length of sequences	2.42

Table 3. Characteristics of the Synthetic datasets

S. No.	Characteristic	Dataset1 Value	Dataset2 Value
1	No of sequences	20000	15000
2	No of distinct items	15	20
3	No of items per itemset	3	3
4	No of itemsets per sequence	5	6

Three sets of experiments were conducted to evaluate the performance of the CSpan algorithm. The first set compares the runtime performance of CSpan with CloSpan and ClaSP using real world dataset BMS WebView for different support values. The second and third sets compare the runtime performance of CSpan with CloSpan and ClaSP using synthetic datasets for different support values.

Fig. 2 shows the results of runtime performance using the real world dataset BMS WebView. The X-axis is the minimum support, while the Y-axis is the algorithms runtime. The support values are set from 0.01 to 0.06. Our proposed algorithm CSpan runs faster than CloSpan and ClaSP, because of the use occurrence checking technique.

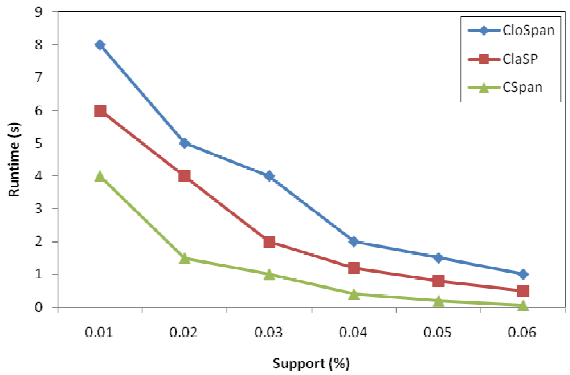

Figure 2. Performance comparison using BMS WebView dataset.

Fig. 3 and Fig. 4 show the results of runtime performance using the synthetic datasets. The X-axis is the minimum support, while the Y-axis is the algorithms runtime. The support values are set from 0.1 to 0.6. Our proposed algorithm CSpan outperforms CloSpan and ClaSP on both the synthetic datasets.

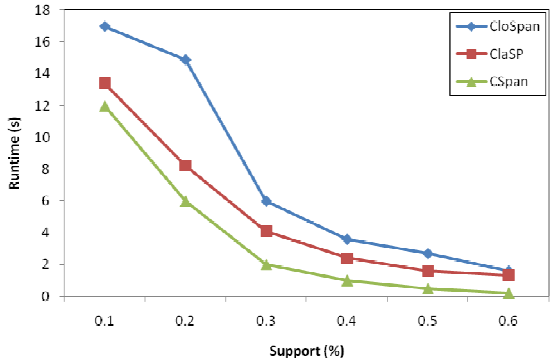

Figure 3. Performance comparison using synthetic dataset1

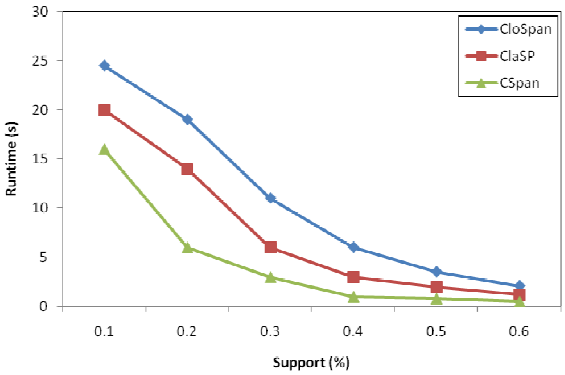

Figure 4. Performance comparison using synthetic dataset2

All the above experiments confirm that the proposed algorithm CSpan is efficient and takes less run time comparing to other algorithms. Because CSpan uses occurrence checking for early detection of closed sequential patterns and stores a result set that contains only closed sequential patterns, where as CloSpan and ClaSP keep a larger number of closed sequential pattern candidates and remove the non-closed ones at the end of the mining process.

6. CONCLUSION

Several researchers focused on the sequential pattern mining problem and many algorithms were developed to mine sequential patterns. Closed sequential pattern mining is a variant of sequential pattern mining and attains broad attention in the recent years because it has the same expression ability of the sequential pattern mining and more compact than the sequential pattern mining.
In this paper, we propose an efficient algorithm CSpan which makes use of a new pruning method called occurrence checking that allows the early detection of closed sequential patterns during the mining process. Our extensive performance study on various real and synthetic datasets shows that the proposed algorithm CSpan outperforms the CloSpan and a recently proposed algorithm ClaSP by an order of magnitude.

In future we will extend CSpan to incorporate user specified constraints. Other interesting research problems that can be pursued include parallel mining of closed sequential patterns [17] and mining of structured patterns.

REFERENCES

[1] R. Agrawal and R. Srikant, "Mining sequential patterns," Proc. Int'l Conf. Data Engineering (ICDE '95), pp. 3-14, Mar. 1995.

[2] X. Yan, J. Han, and R. Afshar, "CloSpan: Mining closed sequential patterns in large databases," Proc. SIAM Int'l Conf. Data Mining (SDM '03), pp. 166-177, May 2003.

[3] J. Wang, J. Han, and Chun Li, "Frequent closed sequence mining without candidate maintenance," IEEE Trans. Knowledge and Data Eng., vol. 19, no. 8, pp. 1042-1056, Aug. 2007.

[4] Antonio Gomariz, Manuel Campos, Roque Marin, and Bart Goethals, "ClaSP: An efficient algorithm for mining frequent closed sequences," PAKDD 2013, LNAI 7818, Part I, pp. 50–61, 2013.

[5] R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements," Proc. Int'l Conf. Extending Database Technology (EDBT '96), pp. 3-17, Mar. 1996.

[6] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.C. Hsu, "FreeSpan: Frequent pattern-projected sequential pattern mining," Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (SIGKDD'00), pp. 355-359, Aug. 2000.

[7] J . Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu, "PrefixSpan : Mining sequential patterns efficiently by prefix-projected pattern growth," Proc. Int'l Conf. Data Engineering (ICDE '01), pp. 215-224, Apr. 2001.

[8] M. Zaki, "SPADE: An efficient algorithm for mining frequent sequences," Machine Learning, vol. 42, pp. 31-60, 2001.

[9] J. Ayres, J. Gehrke, T. Yiu, and J. Flannick, "Sequential pattern mining using a bitmap representation," Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (SIGKDD' 02), pp. 429-435, July 2002.

[10] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lot Lakhal, "Discovering frequent closed itemsets for association rules," Proceedings of the 7th International Conference on Database Theory (ICDT '99), pp. 398-416,1999.

[11] J. Pei, J. Han, and R. Mao, "CLOSET: An efficient algorithm for mining frequent closed itemsets," Proc. ACM SIGMOD Workshop Research Issues in Data Mining and Knowledge Discovery (DMKD '00), pp. 21-30, May 2000.

[12] J. Wang, J. Han, and J. Pei, "CLOSET+: Searching for the best strategies for mining frequent closed itemsets," Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (SIGKDD '03), pp. 236-245, Aug. 2003.

[13] M. Zaki and C. Hsiao, "CHARM: An efficient algorithm for closed itemset mining," Proc. SIAM Int'l Conf. Data Mining (SDM '02), pp. 457-473, Apr. 2002.

[14] Kuo-Yu Huang, Chia-Hui Chang, Jiun-Hung Tung, and Cheng-Tao Ho, "COBRA: Closed sequential pattern mining using bi-phase reduction approach," Proceedings of 8th International Conference, DaWaK, Springer LNCS 4081, pp. 280-291, 2006.

[15] Ron Kohavi, Carla E. Brodley, Brian Frasca, Llew Mason, and Zijian Zheng, "KDD-Cup 2000 organizers' report: Peeling the onion," SIGKDD Explorations, vol. 2, no. 2, pp. 86-93, Dec. 2000.

[16] Fournier-Viger P., An Open-Source Data Mining Library, http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php, 2008, Accessed 20 July 2014.

[17] S. Cong, J. Han, and D.A. Padua, "Parallel mining of closed sequential patterns," Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (SIGKDD '05), pp. 562-567, Aug. 2005.