Indonesian Journal of Electrical Engineering and Computer Science
Vol. 4, No. 3, December 2016, pp. 655 ~ 664
DOI: 10.11591/ijeecs.v4.i3.pp655-664

Mining Top-K Click Stream Sequences Patterns

Mehdi Ali[1], Qun-Xiong Zhu*[2], Yan-Lin He[3]
[1]College of information Science & Technology, Beijing University of Chemical Technology, Beijing, China
[2,3]Engineering Research Center of Intelligent PSE, Ministry of Education of China, Beijing, China
*Corresponding author, e-mail: mehditer@hotmail.com

Abstract

Sequential pattern mining, it is not just important in data mining field, but it is the basis of many applications. However, running applications cost time and memory, especially when dealing with dense of the dataset. Setting the proper minimum support threshold is one of the factors that consume more memory and time. However, it is difficult for users to get the appropriate patterns; it may present too many sequential patterns and makes it difficult for users to comprehend the results. The problem becomes worse and worse when dealing with long click stream sequences or huge dataset. As a solution, we developed an efficient algorithm, called TopK (Top-K click stream sequence pattern mining), which employs the output as top-k patterns, K is the most important and relevant frequencies (with a high support). However, our algorithm based on pseudo-projection to avoid consuming more time and memory, and uses several efficient search space pruning methods together with BI-Directional Extension. Our extensive study and experiments on real click stream datasets show TopK significantly outperforms the previous algorithms.

Keywords*: pattern mining, click stream sequence patterns mining, sequence database, top-k, data mining*

1. Introduction

Many studies have been done on sequential pattern mining algorithms [1-7], an important problem is how the user get a useful amount of patterns by setting the proper minimum support threshold, especially when dealing with dense of database [8-9]. This problem is cost more time and memory to analyze the output patterns. However, setting high minimum support speeds up the algorithms-running time but could get a few patterns or none [9-10]. And the reverse, slow the algorithms and generate an extremely large amount of results that consume time and memory.

To address this problem, it was proposed to redefine the problem of mining click stream Sequence Patterns as the problem of mining the top-*k click stream* Sequence Patterns, where k is the number of Click Stream Sequence Patterns to be set by the user [1], [8-12]. The current best algorithms for this problem are TSP, TKS [1], [10]. However, in our experimental study, we found that TKS and TSP do not perform well on some types of datasets. Therefore, an important research question is could we develop a top-k Click Stream Sequence Patterns mining algorithm more efficient than TKS and TSP? We address this research question by proposing a novel algorithm named TopK (Top-K Click Stream Sequence Patterns mining). TopK is an efficient top-*k* algorithm for Click Stream Sequence Patterns mining.

We employ BI-Directional methods It uses the same representation and basic candidate generation procedure as BIDE [8], [13]. Moreover, TopK incorporates several efficient procedures to prune the search space and rely on a novel data structure named BESTofMS (Best of minimum support) for fasting search operations. An extensive experimental study with various real datasets shows that (1) TopK outperforms the TKS and TSP top-k Click Sstream Sequence Patterns mining algorithms in terms of both execution time and memory usage.

The rest of the article is organized as follows. In section 2, the basic concepts of sequential pattern mining are introduced and the problem of mining the top-K Click Stream Sequence Patterns without minimum support is formally defined. In addition to the related work on sequential pattern mining and top-K frequent pattern mining.

Section 3 presents the algorithm for mining top-K frequent Click Stream Sequence Patterns. A performance study is reported in Section 4, the conclusion in Section 5.

Received September 25, 2016; Revised November 10, 2016; Accepted November 26, 2016

2. Problem Definition & Related Work

Let *I* be a set of items. A set *S* = *{e*1*, e*2*,. . ., ei} ⊆ I* is called an itemset or i-itemsets if it contains i items. For simplicity, from now on we denote an itemset *I* as a concatenation of items between curly brackets. So, *I*1 = {*a,b}* and *I*2 = {*b,c*} are both two 2-itemsets. Also, without loss of generality, we assume the items in every itemset are represented in a lexicographic order [13].

A sequence *s* is a tuple *s* = ⟨*I*1,*I*2, *. . . In* > with *Ii ∈ I*, and *∀i* : 1 ≤ *i* ≤ *n*. We denote the size of a sequence |s| as the number of itemsets in that sequence. We denote the length of a sequence ($I = \sum_{i=0}^{n} |Ii|$) as the number of items in it, and every sequence with *i*- items is called a i-sequence. For instance, the sequence *α* = ⟨{*a,b*},{*b,c*}> is a 4-sequence with a size of 2 itemsets. We say *α* =⟨*Ia*1,*Ia*2, *. . . Ian* > is a subsequence of another sequence *β* = ⟨*Ib*1, *Ib*2 *. . . Ibm* > (or *β* is a super sequence of *α*), denoted as *α sup(β)*, if there exist integers 1 ≤ *j*1 < *j*2 < *. . . < jn* ≤ *m* such that *Ia*1⊆ *Ibj*1, *Ia*2 ⊆ *Ibj*2, *. . ., Ian* ⊆ *Ibjn*. For instance, ⟨{*b},{c}⟩* is a subsequence of *⟨{a,b},{b,c}⟩*, since {*b*} ⊆ {*a,b*} and {*c*} ⊆ {*b,c*} and the order in the itemsets is preserved. Furthermore, the sequence ⟨{*b*},{*c}⟩* is not a subsequence of ⟨{*a,b,c*}⟩. A sequence database *SDB* is collection of input sequences *SDB* = ⟨*s*1, *s*2, *. . . sn*⟩, incrementally ordered by the identifier of the contained sequences. In table 1.a we show a sample input database *SDB* with four input sequences.

2.1. Problem Definition

To address the difficulty of how to set the proper *minsup,* the problem of sequential pattern mining was redefined as the problem of top-k sequential pattern mining [1], [10], [12]. It is to discover a set *of k* Click Stream Sequence Patterns in a sequence database *SDB* such that for each pattern *sa* ∈ *L*, there no sequential pattern found if *sb* ∉ L | *sup*(*sb*) > *sup*(*sa*). For example, for the database of Table 1a, 1b and *k* = 10, the top-*k* Click Stream Sequence Patterns are ⟨{*b*}⟩, ⟨{*c*},{g}⟩, ⟨{*c*}⟩, ⟨{*f*}, {d}⟩, ⟨{*f*}, {*b*}⟩, ⟨{*c*}, {*d*}⟩ and⟨{*d*}⟩ with a support of 3, and ⟨{*f*}, {*g*}⟩, ⟨{*f*}⟩ and ⟨{*g*}⟩, with a support of 4. The definition of this problem is similar to the definition of other top-*k* problems in the field of pattern mining such as top-*k* frequent itemset mining [1], [11], [14], top-*k* association rule mining [10], [15] and top-*k* sequential rule mining.

Table 1a. SDB Sequence Database

SID	Sequences
1	⟨{c, f},{e},{g, *b*},{b},{d}⟩
2	⟨{c, a},{e},{f},{c, f, d, g}⟩
3	⟨{c},{f},{g},{d}⟩

Table 1b. Click Stream Patterns Support

PID	Pattern	Support
1	⟨{c},{g}⟩	3
2	⟨{c},{e}{g}⟩	2
3	⟨{f},{g, b}⟩	2
4	⟨{*b*},{d}⟩	2
5	⟨{*e*},{g}⟩	2

Definition (top-k Click Stream Sequence Pattern): A sequence s is a frequent Click Stream Sequence Pattern in a sequence database D if its support (i.e., occurrence frequency) in D is no less than minimum support. A Click Stream Sequence Pattern s is a top-k if there exist no more (k-1) Click Stream Sequence Patterns.

2.2. Related Work

The sequential pattern mining problem was first proposed by Agrawal and Srikant in [3], and developed a generalized and refined algorithm, GSP [16], based on the Apriori property [17]. Since that, many algorithms related sequential pattern mining have been proposed for performance improvements [18]. Among those algorithms for top-*k* sequential pattern mining like TKS and TSP [9], [11], [14]. Both algorithms were proposed for mining top-*k* sequential patterns, in addition to that TSP also was proposed for respectively top-*k* closed

sequential patterns. The TSP algorithm is based on PrefixSpan [9], [19]. TSP first generates frequent sequential patterns containing a single item. Then it recursively extends projecting and scanning the resulting projected database for each pattern *s* to identify items that appear greater than the value of *minsup* after *s*, and then add the items to *s*. The main benefit of this projection-based approach is that it only considers patterns appearing in the database unlike "generate-and-test" algorithms [1], [14], [20]. TKS is based on vertical database representation and basic candidate-generation on procedure of SPAM [14], [21]. Furthermore, it also includes several efficient strategies to discover top-*k* Click Stream Sequence Pattern efficiently. TKS first scans *SDB* once to construct *V*(*SDB*) and counts the support of each single item [9], [14]. Then, for each frequent item *s*, it calls the procedure "SEARCH". This procedure outputs the pattern ⟨{s}⟩ and recursively explore candidate patterns starting with the prefix ⟨{s}⟩.

However, the downside of TSP approach as projecting databases repeatedly and TKS searching approach are costly time and memory, especially when dealing with long sequences and huge items (our experimental study presented in Section 4).

3. Method Development

We propose a novel algorithm called TopK. TopK is an algorithm to modify the main procedure of the BIDE algorithm [13] to transform it as a top k algorithm.

a. The TopK Algorithm

We modified frequent-sequence procedure in BIDE algorithm, first to enumerate the 1-item frequency then sort the items as a number of frequency descending, then store the k^{th} highest items in a map and set the minimum frequency of it as minsup, then create pseudo projected database, Figure1 show the steps of TopK algorithm.

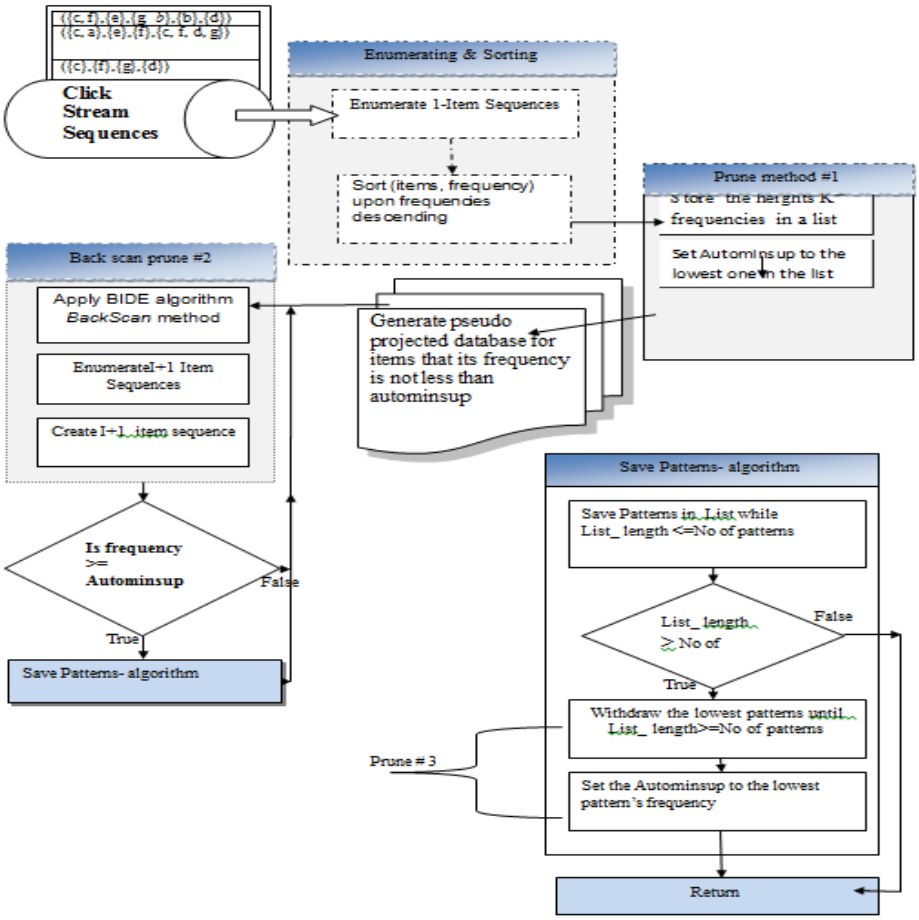

Figure 1. TopK Algorithm Steps

b. Definition (Projected sequence)

Set an input sequence S which contains a prefix i-sequence e1,e2,..ei, S is called the projected sequence when removing the first instance of the prefix i-sequence e1,e2,..ei in w.r.t. prefix e1,e2,..ei in S.

c. Definition (Projected database)

Set an input sequence database SDB, the complete set of projected sequences in SDBB, w.r.t. a prefix sequence e1,e2,..ei is called the projected database w.r.t. prefix e1,e2,..ei in SDB [13].

d. Definition pseudo-projection

Means record a set of pointers instead of physically record, each pointer point to the start position of the corresponding projected sequence.

First of all, enumerate 1-item sequences that their frequencies in the database no less than the autominsup Figure 2. Show the enumeration of Pseudo projection method to find the set of locally frequent items, and Figure 3 shows the TopK algorithm.

It scans the database once to find the frequent 1-sequences (line 2) for each frequent 1-sequences >=2, Sort items upon no of frequency (line3). Save the best of the K^{th} highest minimum support from the sorting items to a list (line 4). Select the lowest from the list and set it as minsup (line5). builds pseudo projected database for each frequent 1-sequence and not less than the minsup (line 6 and 7), treats each frequent 1-sequence as a prefix and uses *BackScan* method to check and prune if any can be pruned (line 9), if not, computes the number of *backward-extension-items* (line 10), call Savepattren((f1, *k*, *minsupq*) functions to save Click Stream Sequence Patterns(line 11), and recursively calls subroutine TopK (Sp_SDB, Sp,min_sup,BEI,FS) (line 12).

Frequent-sequence-{**enumeration (*SDB*, *FS,k*,)**
Input: an input sequence database *SDB*, K
Output: the complete set of frequent sequences, • *FS*
1. *FS* = *Ø*;
2. c.all *Frequent-s.equences(SDB, Ø, min_sup, FS.)*;
3. return *FS*;

Frequent-sequences (*Sp*._*SDB, Sp FS*)
Input: a projected sequence database *Sp*._*SDB*, a prefix sequence *Sp*.,
Output: the current set of frequent sequences, *FS*
1. Set autominsup to 2
2. if *Sp*. is non empty
3. *FS* =• *FS* U *Sp*.;
4. *LF_Sp*.= .locally frequent items (*Sp*._*SDB*, *Sp*.,• *min_sup*•);
5. if *LF_Sp*.is empty
6. Return;
7. for each locally frequent item *i*
8. *Sp*.*i* = <*Sp*.,*i*>;
9. *Sp*.*i_SDB* = p.seudo projected database (*Sp*.*i*,*Sp*._*SDB*•);
10. call *Frequent-s.equences(Sp*.*i_SDB, Sp*.*i,*• *min_sup, FS)*;

Figure 2. Subroutine to Enumerate the Sequences in Dataset

After finding Click Stream Sequence Pattern the TopK algorithm call savepattern function() to save patterns, in Figure 3 and 4. shows the function steps and detailed as follows: Save frequent sequence to list SL, If SL length less than k then, Set autominsup by call BESTofMS (Best of minimum support procedure) by selecting the next frequency in the list FL, otherwise remove sequences from SL until SL length equal K [14], and set autominsup to the lowest frequency in the list SL.

TopK (*SDB*,*K*, *F,S*) ¿**Input**: an input sequence database À*SDB*, K
Output: the complete set of frequent sequences, Â*F*À*S* 1: *FCS* = *Ø* ;
1. Set min_sup to 2;
2. *F1*=frequent 1-sequences(*SDB,autminsup*);
3. F1s=Sort(F1) descending upon the support;
4. min_supA= Select the k th support as min_sup
5. min_supQ= the lowest support of min_supA;
6. for (each 1-sequence *f1* in *F1*) do
7. *SDB*= pseudo projected database (*SDB*);
8. for (each *f1* in *F1s*) do
9. if (!*BackScan*(*f1, SDB*))
10. *BEI*=backward extension check (*f1, SDB*);
11. Savepattren((f1, *k*, *minsupq*)
12. cÎall *bide(SDB,f1,min_supQ, BEI, FS)*;
13. return *FS*;

Figure 3. TopK Algorithm

Savepattren(*FSe*, *k*, *minsup*)
1. 1.put FSe to SP
2. **IF length of FSe** *>k* **THEN**
3. **IF support of FSe** *sup*(*r*) > *minsup* **THEN**
4. Remove all patterns from SP that has the lowest supports until size of SP is equal K
5. **END IF**
6. Set *mins_upQ* to the lowest support of patterns in *L*.
7. **END IF**
   ELSE
   BESTofMS (Rise the *mins_upQ to the next one)*

Figure 4. Save founded pattern function.

4. Experimental Evaluation

This section reports the performance testing of TopK algorithm in large data sets. In particular, we compare the performance of TopK with TSP, TKS and BIDE. Because the synthetic datasets have far different characteristics from the real-world ones, in our experiments we only used some real datasets to do the tests. However, we chose some real datasets which can cover a range of distribution characteristics (sparse, a little dense, and very dense). The datasets used in this study are BMSWebView1 (Gazelle), BMSWebView2 (Gazelle), and Kosarak, real life datasets and, provided by Philippe. It can be obtained at [22]. Table 2 shows details about these datasets.

All experiments were performed on Sony Vio 2.13GHz Intel Core i3 CPU with 3.00 GB memory and Windows 7 Professional installed.

Table 2. Dataset Details

Dataset	No of sequences	No of repeated items	Length of the Largest sequence	Dataset Type
Kosarak 1	10 000 sequences	81407	608	Click-stream data of a Hungarian on-line news portal
Kosarak 2	25 000 sequences	201062	608	
BMSWebView1 (Gazelle)	59,601	149638	267	Click stream data from an e-commerce
BMSWebView2 (Gazelle)	77,512	358278	361	

Our algorithm was based on BIDE algorithm, we compared TopK algorithm with BIDE. The comparison is based on assigning the optimal minsup to BIDE so that it generates the same set of top-k Click Stream Sequence Patterns as TopK for specified values of k.

Table 3. Comparative the Running Time for the Three Algorithms

Dataset	Algorithm	Total Runtime (MS)			
		k=500	k=1500	k=2500	k=3500
Kosarak1	TopK	2398	4564	5537	5996
contains 10 000	TSP	2844	4920	5862	6524
sequences	TKS	2545	9054	12117	24458
Kosarak2	TopK	4428	7575	11093	13652
contains 25 000	TSP	4894	7923	12184	16877
sequences	TKS	4693	11192	21554	31923
BMSWebView1	TopK	4479	5435	7213	8015
contains 59,601	TSP	4674	5985	7588	8211
sequences	TKS	2501	6897	12859	18316
BMSWebView2	TopK	4350	9692	12656	17536
contains 77,512	TSP	4809	9631	12696	23255
sequences	TKS	2511	5505	12486	22063

The experiments show that TopK is more efficient than BIDE in terms of run time and memory storage (specified k as top-k sequential mining) as seen in Figure (5-8).

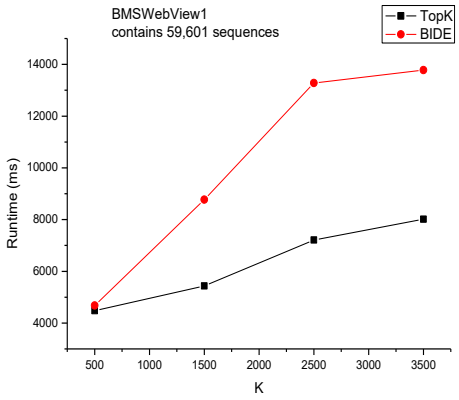

Figure 5. Execution time (ms) of BMSWebView1 dataset

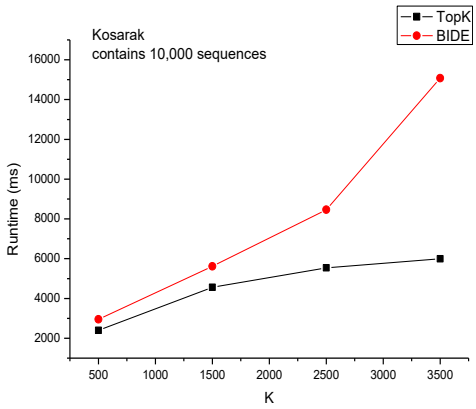

Figure 6. Execution time (ms) of Kosarak1 dataset

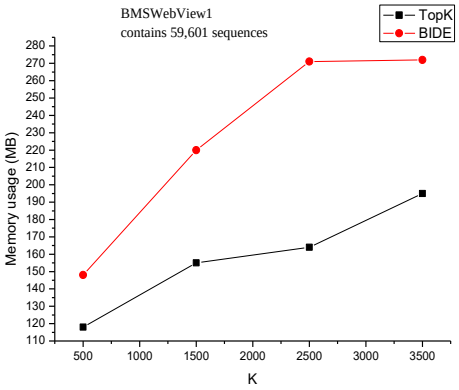

Figure 7. Memory usage (MB) of BMSWebView1 dataset

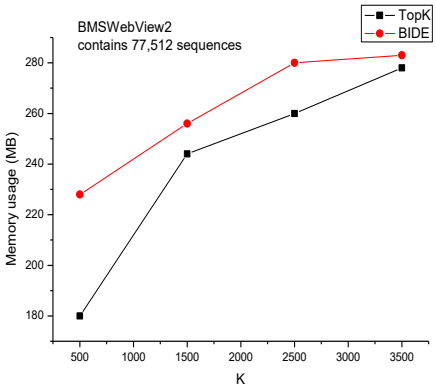

Figure 8. Memory usage (MB) of BMSWebView2 dataset

In the experiments we compared TopK with two top- k sequential mining algorithms, TSP and TKS, see Table 3. The source codes of TSP and TKS provided by TKS-corresponding-author [22].

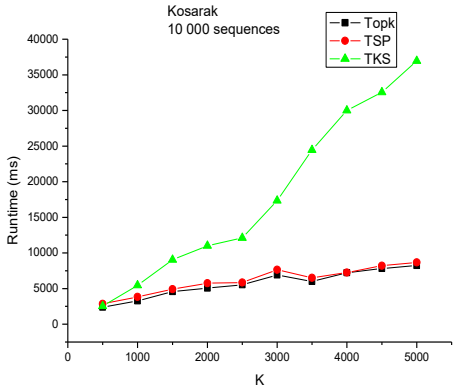

Figure 9. Execution time (ms) of Kosarak1 dataset

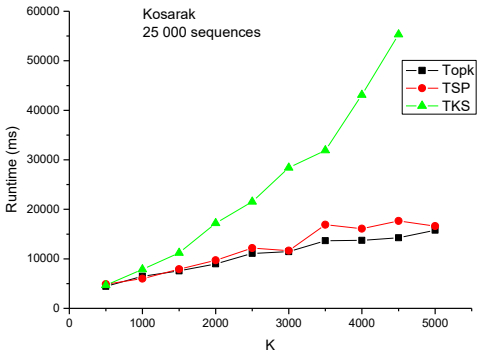

Figure 10. Execution time (ms) of Kosarak2 dataset

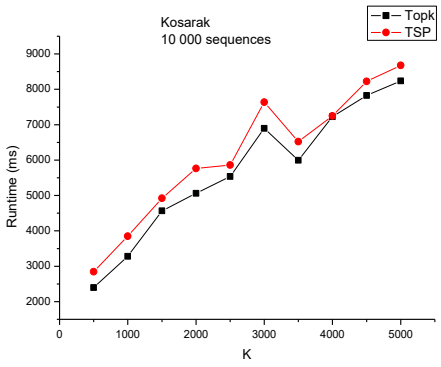

Figure 11. Execution time (ms) of Kosarak1 dataset –TopK vs. TSP

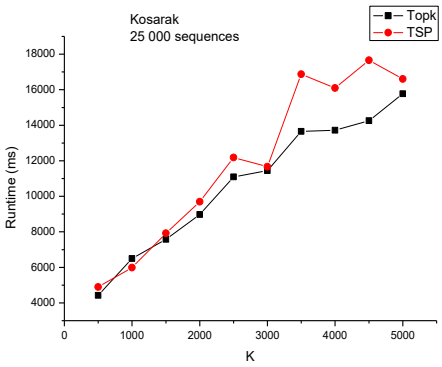

Figure 12. Execution time (ms) ofKosarak2 dataset –TopK vs. TSP

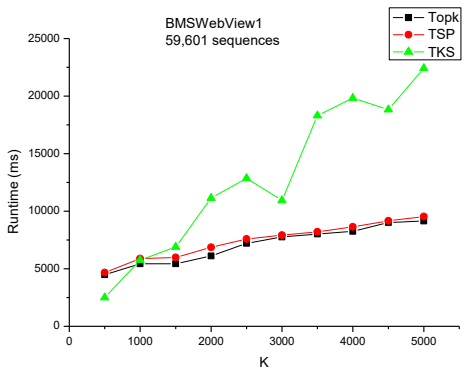

Figure 13. Execution time (ms) of BMSWebView1 dataset

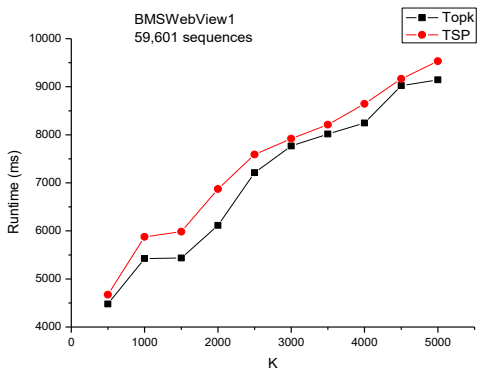

Figure 14. Execution time (ms) of BMSWebView1 dataset –TopK vs TSP

We executed the three algorithms Topk, TKS and TSP on all dataset with varying of the quantity of K,K⊆{500,1500,2500,3500} to assessment TopK algorithm comparing with the others.

Table 3. Comparative Memory Usage between Three Algorithms

Dataset	Algorithm	Maximum Memory Usage (MB)			
		k=500	k=1500	k=2500	k=3500
Kosarak1	TopK	4	120	32	78
contains 10 000	TSP	7	24	39	85
sequences	TKS	69	52	71	87
Kosarak2	TopK	41	22	29	135
contains 25 000	TSP	41	23	37	147
sequences	TKS	69	52	103	123
BMSWebView1	TopK	18	55	64	95
contains 59,601	TSP	25	71	94	29
sequences	TKS	2	22	114	23
BMSWebView2	TopK	80	44	60	78
contains 77, 512	TSP	81	48	97	28
sequences	TKS	100	11	154	94

Figures 9-10, show the comparison between TopK,TSP, and TKS to execute Kosarak dataset with two deferent sizes, we notice that TopK better than TKS and TSP, and Figures 11-14 show TopK stable for execution two types of datasets and outperforms TSP in terms of executing time.

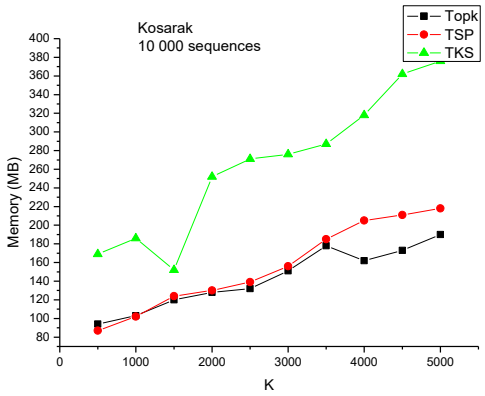

Figure 15. Memory usage (MB) of Kosarak1 dataset

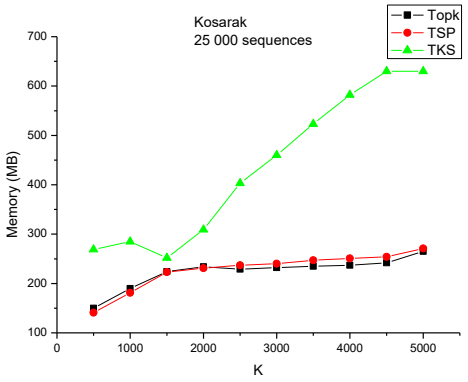

Figure 16. Memory usage (MB) of Kosarak2 dataset

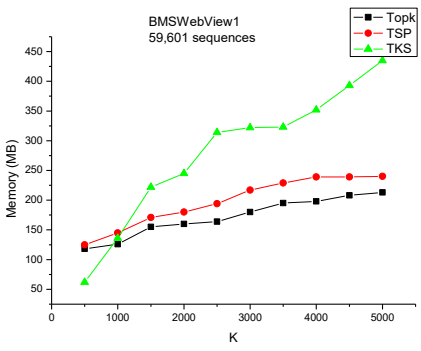

Figure 17. Memory usage (MB) of BMSWebView1 dataset

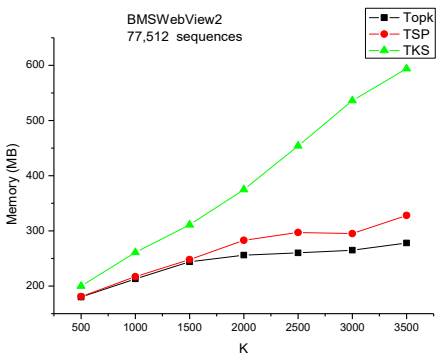

Figure 18. Memory usage (MB) of BMSWebView2 dataset

Figures 15-20 show the comparison between the three TopK, TSP, TKS for execution deferent types and sizes fo dataset in terms of memory usage, figures show that TopK is more efficient than the others, TopK uses less memory and stable for the deferent sizes of the dataset.

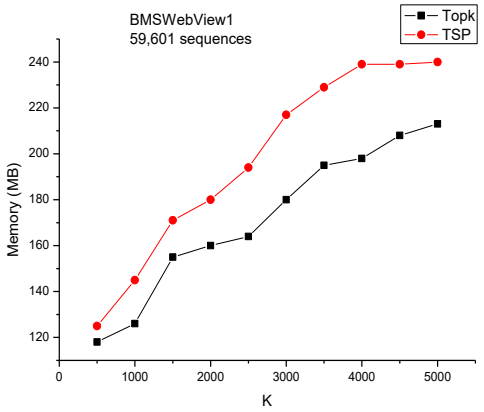

Figure 19. Memory usage (MB) of BMSWebView1 dataset- TopK vs. TSP

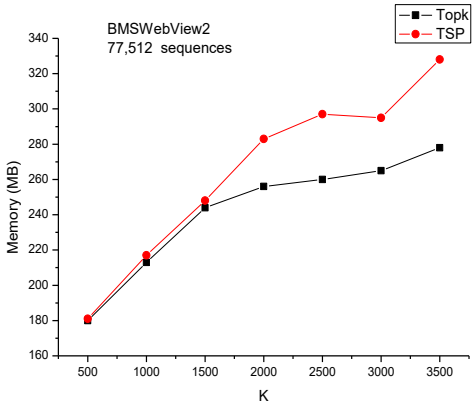

Figure 20. Memory usage (MB) of BMSWebView2 dataset- TopK vs. TSP

Our algorithm executed all datasets and was the best results than the TSP and TKS algorithms in terms of execution time and memory usage. TopK faster than TSP and TKS and use less memory.

5. Conclusions

We proposed TopK, a novel algorithm for mining frequent sequences our proposed TopK algorithm is an efficient algorithm to discover the *top-k* Click Stream Sequence Patterns as output, in term of execution time and less memory, where K is an integer number setting by the user as input. Our method employed BI-Directional to work fast and with high efficiency for discovering the *top-k* Click Stream Sequence Patterns. Our experiments show that TopK consumes order(s) of magnitude less memory and runs over an order of magnitude faster than the previously developed top-k sequential pattern mining algorithms, especially when K is high, and has better scalability with respect to k.

References

[1] Petre Tzvetkov, Xifeng Yan, Jiawei Han. *TSP: Mining Top-K Closed Sequential Patterns*. ICDM. 2013; 347-354.
[2] Han J, Kamber M. Data Mining: Concepts and Techniques. San *Francis-co: Morgan Kaufmann Publ*. 2006; 2nd ed.
[3] Agrawal R, Srikant, R. *Mining Sequential Patterns*. Proc. Int. Conf. on Data Engineering. 1995; 3-14.
[4] M. Zaki. SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning Journal*. 2001; 42: 1-2.
[5] RJ Bayardo. *Efficiently mining long patterns from databases*, In Proc. Int. Conf. Managemen of Data (SIGMOD'98). 1998; 85-93.
[6] Xiaofeng Liao, Liping Ding, Jian Gu. Extended Probabilistic Latent Semantic Analysis Model For Topics In Time-Stamped Images. *Intelligent Automation & Soft Computing*. 2013; 17(7): 997-1007.
[7] Xiaodong Zhu. On Data Mining Technology to the Quantitative Efficiency Assessment using SBM Model: An Empirical Study on Education Efficiency in Jiangxi Province. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2014; 12(2): 1933-1938.
[8] J Han, J Wang, Y Lu, P Tzvetkov. *Mining top-k frequent closed patterns without minimum support*. In *ICDM.* 2002; 211–218.

[9] Tzvetkov P, Yan X. Han J. TSP: Mining Top-k Closed Sequential Patterns. *Knowledge and Information Systems.* 2005; 7(4); 438-457.
[10] Fournier-Viger, P, Tseng, VS. *Mining Top-K Sequential Rules*. Proc. of the 7th In-tern. Conf. on Advanced Data Mining and Applications (ADMA 2011). Springer LNAI 7121. 2011; 180-194.
[11] Mao Yimin, Xue Xiaofang, Chen Jinqing. An Efficient Algorithm for Mining Top-K Closed Frequent Item sets over Data Streams over Data Streams. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2013; 11(7); 3759-3766.
[12] Kun Ta C, Huang, JL, Chen M. Mining Top-k Frequent Patterns in the Presence of the Memory Constraint. *VLDB Journal*. 2008; 17(5); 1321-1344.
[13] J. Wang, J Han. *BIDE: Efficient Mining of Frequent Closed Sequences*. ICDE .2004; 79-90.
[14] Fengqin Han, Ming Lei, Wenjuan Zhao, Jianxi Yang. New Support Vector Machine for Imbalance Data Classification. *Intelligent Automation & Soft Computing*. 2013; 18(6): 679-686.
[15] Mustafa Bin Man, Wan Aezwani Wan Abu Bakar, Zailani Abdullah, Masita@Masila Abd Jalil, Tutut Herawan. Mining Association Rules: A Case Study on Benchmark Dense Data. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2016; 3(3).
[16] R Srikant, R Agrawal. *Mining sequential patterns: Generalizations and performance improvements*. In EDBT'96. Avignon France. 1996.
[17] R Agrawal, R Srikant. *Fast algorithms for mining association rules*. In VLDB'94, Santiago, Chile.1994.
[18] Fengqing Han, Ming Lei, Wenjuan Zhao Jianxi Yang .New Support Vector Machine for Imbalance Data Classification. *Intelligent Automation & Soft Computing*. 2013; 18(6); 679-686.
[19] J Pei, J Han, B Mortazavi-Asl, J Wang, H Pinto, Q Chen, U Dayal, M Hsu. Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach. *IEEE Trans. Knowl. Data Eng*. 2004; 16(11); 1424-1440.
[20] Ho, J, Lukov, L, Chawla, S. *Sequential pattern mining with constraints on large protein databases*. In Proceedings of the 12th International Conference on Management of Data (COMAD). 2005; 89-100.
[21] J Ayres, J Gehrke, T Yiu, J Flannick. *Sequential Pattern Mining Using Bitmaps*. In Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Edmonton, Alberta, Canada, July 2002.
[22] Fournier-Viger, P, Lin, CW, Gomariz, A, Gueniche, T, Soltani, A, Deng, Z, Lam, HT. *The SPMF Open-Source Data Mining Library Version 2*. Proc. 19th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD 2016) Part III, Springer LNCS 9853. 2016; 36-40.