Journal of Computer Science and Cybernetics, V.34, N.3 (2018), 249–263
DOI 10.15625/1813-9663/34/3/13053

MINING TOP-K FREQUENT SEQUENTIAL PATTERN IN ITEM INTERVAL EXTENDED SEQUENCE DATABASE

TRAN HUY DUONG[1,a], NGUYEN TRUONG THANG[1], VU DUC THI[2], TRAN THE ANH[1]

[1]*Institute of Information Technology, Vietnam Academy of Science and Technology*
[2]*Information Technology Institute, Vietnam National University (VNU)*
[a]*HuyDuong@ioit.ac.vn*

Abstract. Frequent sequential pattern mining in item interval extended sequence database *(iSDB)* has been one of the interesting tasks in recent years. Unlike classic frequent sequential pattern mining, the pattern mining in *iSDB* also considers the item interval between successive items; thus, it may extract more meaningful sequential patterns in real life. Most previous frequent sequential pattern mining in *iSDB* algorithms needs a minimum support threshold (*minsup*) to perform the mining. However, it's not easy for users to provide an appropriate threshold in practice. The too high *minsup* value will lead to missing valuable patterns, while the too low *minsup* value may generate too many useless patterns. To address this problem, we propose an algorithm: TopKWFP - top-K weighted frequent sequential pattern mining in item interval extended sequence database. Our algorithm doesn't need to provide a fixed *minsup* value, this *minsup* value will dynamically raise during the mining process.

Keywords. Sequential pattern; Item interval; Top-K.

1. INTRODUCTION

Sequential pattern mining is an important task in data mining field with wide applications. In real life, sequential pattern data are very popular, like customer purchase sequential patterns, medical treatment sequential patterns, weblogs sequential patterns,... The main purpose of sequential pattern mining is finding all subsequences that frequently occur in a sequence database.

Some well-known sequential pattern mining algorithms are AprioriAll [1], GSP [2], PrefixSpan [3], SPADE [4], SPAM [5]. These algorithms only consider the occurrence frequency (support), Hirate and Yamana [6] proposed an algorithm which considers the item interval between items. At these frequencies-based algorithms, the downward closure property (or Apriori [1] property) plays a fundamental role in identifying frequent sequence patterns. However, these algorithms only consider the occurrence frequency of sequential patterns, regardless of their significance. To indicate the significance of data items, each item can be assigned a weighted value. Some algorithms with weighted items are MINWAL [7], WAR [8], WARM [9], FWARM [10], WFIM [11], WPrefixSpan [12].

In [13], a WIPrefixSpan algorithm is built for mining sequential pattern in ISDB. This algorithm not only considers item interval, occurrence frequency but also the significance (weighted value) of each item. Although WIPrefixSpan can extract weighted sequential patterns with item interval due to

a minimum threshold *wminsup* and four constraints *C1, C2, C3, C4*; it's really difficult to specify an appropriate minimum threshold and to directly extract the most valuable patterns. Because there are multiple factors which affect the result: the distribution of items and weights, density of database, the lengths of the sequences,... Hence, with the same threshold, some datasets may produce millions of patterns while others may produce nothing.

The traditional sequential pattern framework faces the same challenge. Therefore, some top-K pattern mining algorithms were proposed in [14, 15, 16, 17], (itemset mining) and [18, 19, 20, 21, 22] (sequential pattern mining) to find the highest frequency patterns. In the top-K frequent pattern mining, instead of letting a user specify a threshold, the top-K pattern selection algorithms allow a user to set the number of top-K high frequency patterns to be discovered. Those top-K frequent pattern mining algorithms only interest in occurrence frequency, but not item interval and weights of items. In fact, top-K sequential pattern mining with item interval and weight has many differences with a classic top-K sequential pattern mining, thus brings more challenges. In order to address those challenges, we propose a TopKWFP algorithm.

The remainder of the paper is organized as follows. Section 2 defines the problem of mining top-K weighted sequential pattern mining with item interval. Section 3 details the TopKWFP algorithm. Section 4 shows experimental results and evaluation. The conclusion is presented in Section 5.

2. PROBLEM STATEMENT

Let $I = \{i_1, i_2, ..., i_n\}$ be a set of distinct items. Each item $i_j \in I$ is assigned a weight w_j where $j = 1, ..., n$. A sequence is an ordered list of itemsets denoted by $S = \langle (t_{1,1}, s_1), (t_{1,2}, s_2), ..., (t_{1,m}, s_m) \rangle$ with $s_j \subseteq I$ where $1 \leq j \leq m$ is an itemset which is called an element of sequence, $t_{\alpha\beta}$ is item interval between s_α and s_β. A sequence S is eliminated if it has only one item. An item can occur at most once in an element of a sequence s_j, but can occur multiple times in different elements of a sequence S.

The size $|S|$ of a sequence is the number of elements in the sequence S. The length $l(S)$ of the sequence S is the number of instances of items in S. An item interval sequence database $(iSDB) = \{S_1, S_2, ..., S_m\}$ is a set of tuples $(iSID, S)$ where $iSID$ is an identification of a sequence and S_k is a sequence.

For example, Table 1 is an $iSDB$ with 3 sequences, first sequence with $iSID = 10$ shows that item a occurs first in the sequence, then item a, b, c occurs at the same time with item interval 1, then item a, c occurs at the same time with item interval 3. Table 2 is weights of items.

Definition 1. Support, Normalized weight and Normalized weighted support of a sequence:

- The (absolute) support of a sequence α in a sequence database SDB is defined as the number of sequences that contain α, and is denoted by $support(\alpha)$. In other words,

$$support(\alpha) = |\{s | \alpha \subseteq s \wedge s \in SDB\}|.$$

- Given a sequence $\alpha = \langle (t_{1,1}, s_1), (t_{1,2}, s_2), ..., (t_{1,m}, s_m) \rangle$ where s_i is $(x_{i1}x_{i2}...x_{i|s_i|})$, $|s_i|$ denotes the length of element s_i. The Normalized weight of the sequence α , denoted $NW(\alpha)$,

Table 1. An $iSDB$

$iSID$	Sequence
10	$< (0, a), (1, abc), (3, ac) >$
20	$< (0, ad), (3, c) >$
30	$< (0, aef), (2, ab) >$

Table 2. Weights of items

Items	Weight
a	0,9
b	0,75
c	0,8
d	0.85
e	0.75
f	0.7

is defined as follows

$$NW(\alpha) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{|s_i|} weight(x_{ij})}{\sum_{i=1}^{m} |s_i|}.$$

- We call the quantity

$$NWsupport(\alpha) = NW(\alpha) * support(\alpha)$$

the Normalized weighted support of sequence α.

For example, for $\alpha = \langle (0, a), (2, a) \rangle$, we have

$$NWsupport(\langle (0, a), (2, a) \rangle) = \frac{0,9 + 0,9}{2} * 2 = 1,8.$$

Definition 2. Subsequence of another sequence.

A sequence $\alpha = \langle (t_{1,1}, a_1), (t_{1,2}, a_2), ..., (t_{1,n}, a_n) \rangle$ is called a subsequence of another sequence $\beta = \langle (t_{1,1}, b_1), (t_{1,2}, b_2), ..., (t_{1,m}, b_m) \rangle$, and β is a supersequence of α, denoted as $\alpha \subseteq \beta$, if there exist integers $1 < j_1 < j_2 < ... < j_n \leq m$ such that $a_1 \subseteq b_{j_1}, a_2 \subseteq b_{j_2}, ..., a_n \subseteq b_{j_n}$. For example, if $\alpha = \langle (ab), d \rangle$, and $\beta = \langle (abc), (de) \rangle$, where a, b, c, d, and e are items, then α is a subsequence of β and β is a supersequence of α.

Definition 3. Prefix and subfix of a sequence.

Suppose that all the items within an event are listed alphabetically. For example, instead of listing the items in an event as, say, (bac), we list them as (abc) without loss of generality. Given a sequence $\alpha = \langle e_1, e_2, ..., e_n \rangle$, a sequence $\beta = \langle e'_1, e'_2, ..., e'_m \rangle (m \leq n)$ is called a prefix of α if and only if:

- $e'_i = e_i$ for $(i \leq m - 1)$,
- $e'_m \subseteq e_m$,
- all the frequent items in $(e_m - e'_m)$ are alphabetically after those in e'_m.

Sequence $\gamma = \langle e''_m, e_{m+1}, ..., e_n \rangle$ is called the postfix of α with respect to prefix β. We also denote $\alpha = \beta.\gamma$. Note if β is not a subsequence of α, the postfix of α with respect to β is empty.

Definition 4. Item interval constraints.

Let $\langle (t_{1,1}, s_1), (t_{1,2}, s_2), (t_{1,3}, s_3), ..., (t_{1,m}, s_m) \rangle$ be an extracted interval extended sequence. The four item interval constraints are defined as follows:

- $C1$: Let $min\_interval$ be a minimum item interval between any two adjacent items, $C1$ is defined as $t_{i,i+1} \geq min\_interval$ for all $\{i|1 \leq i \leq m-1\}$.
- $C2$: Let $max\_interval$ be a maximal item interval between any two adjacent items, $C2$ is defined as $t_{i,i+1} \leq max\_interval$ for all $\{i|1 \leq i \leq m-1\}$.
- $C3$: Let $min\_whole\_interval$ be a minimum item interval between the head and tail of the sequence, $C3$ is defined as $t_{1,m} \geq min\_whole\_interval$.
- $C4$: Let $max\_whole\_interval$ be the maximal item interval between the head and tail of the sequence, $C4$ is defined as $t_{1,m} \leq max\_whole\_interval$.

Definition 5. Candidate sequence pattern.

Given a support threshold wminsup. An α sequence is called candidate weighted sequence pattern if it satisfies

$$Support(\alpha) * MaxW \geq wminsup \text{ and } \alpha \text{ satisfies } C1, C2, C3, C4,$$

where $MaxW$ is the maximum value of weights of the items in $iSDB$. Candidate sequence patterns are built for the purpose of pruning the search space and still ensure downward closure property in the mining item interval normalized weighted frequent sequential patterns.

Definition 6. Top-K item-interval weighted frequent sequential patterns.

A sequence t is called a top-K item-interval weighted frequent sequential patterns if there are less than k sequences having normalized weighted support higher than $NWSupport(t)$ and t satisfies item interval constraints $C1, C2, C3, C4$. The optimum $wminsup$ is denoted and defined as ε = min$\{NWSupport(t)|t \in T\}$ where T means the set of top-K item-interval weighted frequent sequential patterns.

Given an item interval extended sequence database $iSDB$ and an integer k, the problem of finding the set of top-K item-interval weighted frequent sequential patterns is to discover all the sequential patterns t which have $NWSupport(t) \geq \varepsilon$ and t satisfies item interval constraints $C1, C2, C3, C4$.

3. TopKWFP ALGORITHM

We introduced the problem of finding the set of top-K item-interval weighted frequent sequential patterns in the previous section. In this section, we specify and present an efficient algorithm, TopKWFP, for mining top-K item-interval weighted frequent sequential patterns. TopKWFP is based on WIPrefixSpan [12] which uses a prefix sequence database and growth patterns approach. Firstly, we present a basic TopKWFP algorithm with raising the weighted support threshold *(wminsup)* strategy. Then, we add an efficient strategy to create the most promising patterns.

A. Raising minimum weighted threshold *wminsup*:

TopKWFP algorithm finds top-K item-interval weighted frequent sequential patterns which use Prefixspan's pattern-growth method. Firstly, *wminsup* is set to zero, then sequential patterns are found by applying pattern-growth method. Whenever a pattern is found, it will be inserted into an ordered-by-weighted-support list L. This list is used to maintain the top-K pattern on-the-fly.

Once there are k patterns in the list L, the internal $wminsup$ variable is raised to the weighted support of the pattern with the lowest weighted support in L. With this raising minimum weighted threshold $wminsup$ strategy, the TopKWFP algorithm's search space is reduced. After k patterns are found in list L and wminsup value is raised, the newly found pattern will be inserted to L if it has weighted support value higher than $wminsup$ and the patterns with weighted support lower than new $wminsup$ will be eliminated from L. The internal $wminsup$ value is thereafter raised to the weighted support of the new pattern with the lowest weighted support in L,... The TopKWFP algorithm continues until there is no pattern found, then the algorithm is finished and output the set of top-K item-interval weighted frequent sequential patterns. However, an algorithm simply incorporating raising minimum weighted threshold strategy does not have good performance.

B. Generating the most promising candidates:

To improve the performance of TopKWFP, we have added a second strategy: Generating the most promising candidates. It is to try to generate the most promising candidate sequential patterns first. The rationale of this strategy is that if patterns with high support are found earlier, it allows TopKWFP to raise its internal $wminsup$ variable faster, and thus to prune a larger part of the search space. To implement this strategy, TopKWFP uses an internal variable R to maintain at any time the set of patterns that can be extended to generate candidates. TopKWFP then always extends the pattern having the highest support first. It is noticed that all pattern in the R was ordered by support instead of $NWSupport$, because R contains only candidate patterns but not frequent sequence patterns.

The pseudo code of the TopKWFP algorithm is shown below:

Algorithm TopKWFP

Input : – Item interval extended sequence database $iSDB$
– Weight value of each item i *W(i)*
– Item interval constraint $C1, C2, C3, C4$
– a number k

Output : The set of top-K item interval weighted frequent sequential patterns.

1: **Start**
2: $R = \varnothing; L = \varnothing; wminsup := 0;$
3: Scan $iSDB$ first time, count the support of each item i in $iSDB$, denoted as $support(i)$, and count the $MaxW$=Max$\{W(i)\}$;
4: **for** each item i in $iSDB$ **do**
5: $\alpha = \langle(0, i)\rangle;$
6: **if** $support(\alpha) * MaxW \geq wminsup$ **then**
7: $R = R \cup \alpha;$
8: **end if**
9: **if** $support(\alpha) * NW(\alpha) \geq wminsup$ **then**
10: SAVE$(\alpha, L, k, wminsup)$;

end if
end for
if $k <$ number of all item i in $iSDB$ **then**
Scan $iSDB$ second time, eliminate all items i in $iSDB$ don't satisfy condition $support(i) * MaxW \geq wminsup$;
end if
while $\exists r \in R$ and $support(r) * MaxW \geq wminsup$ **do**
$r =$ the highest Support value sequence in R;
Build r-projected database $iSDB|r$;
PROJECTION$(iSDB|r, W(i), C1, C2, C3, C4, wminsup, k)$;
Remove r from R;
Remove from R all item s which $support(s) * MaxW \leq wminsup$;
end while
Return L;
End

The PROJECTION procedure

procedure PROJECTION$(iSDB|r, W(i), C1, C2, C3, C4, wminsup, k)$
Scan $iSDB|r$ to find all pairs of item $(\triangle t; i)$ that satisfy $support(i) * MaxW \geq wminsup$, $C1$ and $C2$, with i is an item data and $\triangle t$ is item interval between r and i;
for each $(\triangle t; i)$ **do**
$r = \langle r, (\triangle t; i)\rangle$;
if r satisfies $C4$ **then**
$R = R \cup r$;
if r satisfies $C3$ and $support(r) * NW(r) \geq wminsup$ **then** SAVE$(r, L, k, wminsup)$;
end if
end if
end for
end procedure

The SAVE procedure

procedure SAVE $(r, L, k, wminsup)$
$L = L \cup \{r\}$;
if $|L| > k$ **then**
if $NWSupport(r) > wminsup$ **then**
while $|L| > k$ and $\exists s \in L \mid NWSupport(s) = wminsup$ **do**
REMOVE s from L;
end while
end if
Set $wminsup$ to the lowest weighted support of patterns in L;
end if
end procedure

The TopKWFP algorithm first initializes the variables R and L as the empty set, and $wminsup$ to 0 (line 2). Then, $iSDB$ is scanned first time to find all item i in $iSDB$ and the $MaxW$ value. With each item i, create initial interval extended sequences $\alpha = \langle(0, i)\rangle$ (line 5), then check condition $support(\alpha) * MaxW \geq wminsup$ and put the sequences satisfying that condition into R (line 6 to 8). We continue with checking condition $support(\alpha) * NW(\alpha) \geq wminsup$, with each sequence α satisfies the condition, call the SAVE procedure (line 9 to 11).

If there are more items in $iSDB$ than k value, the $wminsup$ will rise above zero, so we will scan $iSDB$ second time to eliminate all items which is not a candidate (line 13-15). After that, a while loop is performed. It recursively gets the highest support sequential pattern (line 16-17), then generates patterns by building a project database and call the PROJECTION procedure in (line 18-19). After that, pattern r is removed from R as well as all other patterns which have $support(s) * MaxW \leq wminsup$ (line 20-21). The ideal of the while loop has been to always extend the pattern having the highest support first because it is more likely to generate patterns having a high weighted support and thus to allow to raise $wminsup$ more quickly for pruning the search space. The loop terminates when there is no more candidate in R with $support(r) * MaxW \geq wminsup$. At this moment, the set L contains the top-K item interval weighted sequential patterns (line 23).

The PROJECTION procedure scans projected database $iSDB|r$ to generate candidates and add to the R. Firstly, it scans project database $iSDB|r$ to find all itemized interval pairs $(\triangle t;i)$ that satisfy $support(i) * MaxW \geq wminsup$ and constraints $C1, C2$ (line 2). Then, with each pattern found, the procedure appends $(\triangle t;i)$ to r to become a new pattern $r = \langle r, (\triangle t;i)\rangle$ (line 4). Next, the procedure checks whether the new pattern satisfies constraint $C4$ or not (line 5). If it satisfies $C4$, we consider it a candidate and add to set R (line 6). After that, the new pattern is checked with constraint $C3$, if it satisfies $C3$ then the SAVE procedure is called to add it into L (line 7-9). PROJECTION procedure checks whether the extracted frequent interval extended sequences satisfy $C3$ or not, after they have been extracted with satisfying minimum support constraint, $C1, C2$, and $C4$. This is because that we are not able to judge the satisfaction of constraint $C3$ before other constraints. Although an interval extended sequence δ does not satisfy the constraint $C3$, some supersets ε, which include δ as a subset, may satisfy the constraint $C3$. On the other hand, when a candidate extracted sequence does not satisfy $C3$, it is not extracted as a result sequence.

The SAVE procedure raises $wminsup$ and update the list L when a new weighted frequent pattern r is found. The first step of SAVE is to add the pattern r to L (line 2). Then, if L contains more than k patterns and the weighted support is higher than $wminsup$, patterns from L that have exactly the weighted support equal to $wminsup$ can be removed until only k patterns are kept (line 4 to 7). Finally, $wminsup$ is raised to the weighted support of the pattern in L having the lowest weighted support (line 8). By this simple scheme, the top-K pattens found are maintained in L.

4. EXPERIMENTAL RESULTS AND EVALUATION

In this session, we evaluate the performance of TopKWFP on a variety of datasets. According to our study, there is no algorithm can solve the top-K item interval weighted frequent sequential pattern problem, so we compare TopKWFP in 2 situations: use only raising minimum weighted thres-

hold *wminsup* strategy (TopKWFP1) and use both strategies raising minimum weighted threshold *wminsup* and generating the most promising candidates (TopKWFP2).

In the general case, the complexity of the algorithm TopKWFP is exponential $O(n^L)$, where n is the number of items in the dataset and L is the maximum length of the sequence in the whole database.

Experiments were performed on a computer with a 7^{th} generation Core i7 processor running Windows 10 and 8 GB RAM. The TopKWFP algorithm was implemented in Java. All memory measurements were done using the Java API. Experiments were carried on five real-life datasets having varied characteristics and representing four different types of data (web click stream, text from books and sign language utterances). These datasets are Bible, BMS-WebView1, FIFA, Leviathan, Sign. Table 3 summarizes their characteristics. All datasets were downloaded from SPMF datamining framework `http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php`

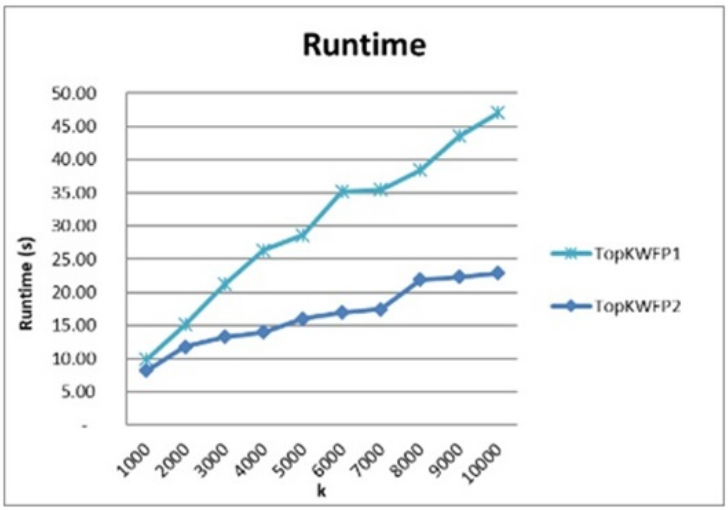
Table 3. Datasets'characteristics

Dataset	**Sequence count**	**Distinct item count**	**Avg. seq. length (items)**	**Type of data**
Bible	36369	13905	21.64	book
BMS-WebView1	59601	497	2.42	web click stream
FIFA	20450	2990	34.74	web click stream
Leviathan	5834	9025	33.81	book

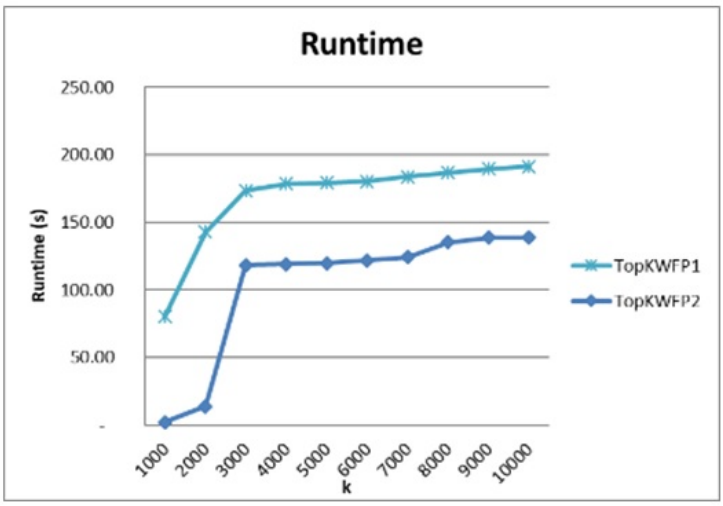
All above datasets have no item interval and weight data, so we must generate item interval and weight for each. Item interval is incrementally generated, two adjacent items have one item interval distant. Weighted values are randomly generated in range [0.2;0.8].

In the first test, we ran the algorithm on each dataset with k varied from 1000 to 10000 to evaluate the influence of k on the runtime and the memory usage. The four constraints were set as $C1$=0; $C2$= 5; $C3$= 0; $C4$= 15. The results are shown in Figure 1 and Figure 2. It can be seen that the TopKWFP2 is more efficient than TopKWFP1 in both runtime and memory usage aspect. The algorithm also has good scalability in both cases, while increasing k value. By applying 2 strategies, the performance of the algorithm has increased.

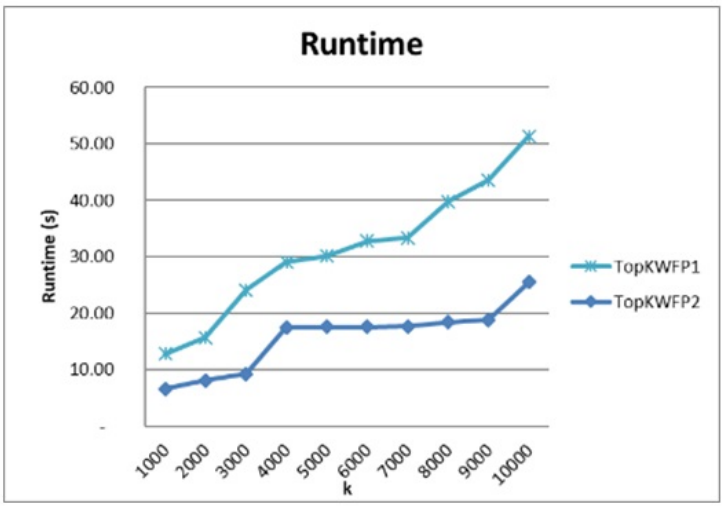
In the second test, we compare the TopKWFP algorithm which uses both strategies with the WIPrefixSpan with optimum support (which is hard for the user to choose). We do that by first running the TopKWFP algorithm to find the optimum support and then use this support as a parameter for the WIPrefixSpan algorithm. The results are shown in Figure 3. We can see that TopKWFP mines these datasets very efficiently and in most cases runs several times faster than WIPrefixSpan. The reason of the better performance of TopKWFP is that TopKWFP uses generating the most promising candidates. This strategy only chooses the most promising patterns (the highest support patterns) to extend while WIPrefixSpan must extend all patterns in the search space.



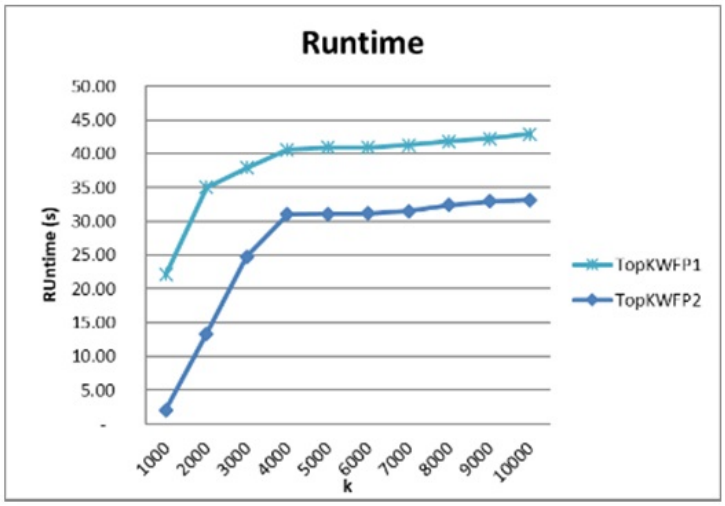
a) Bible



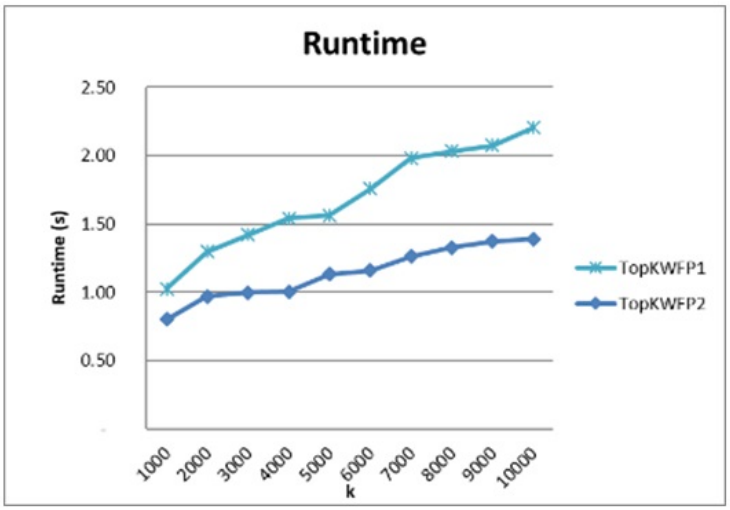
b) BMS-WebView1



c) Fifa

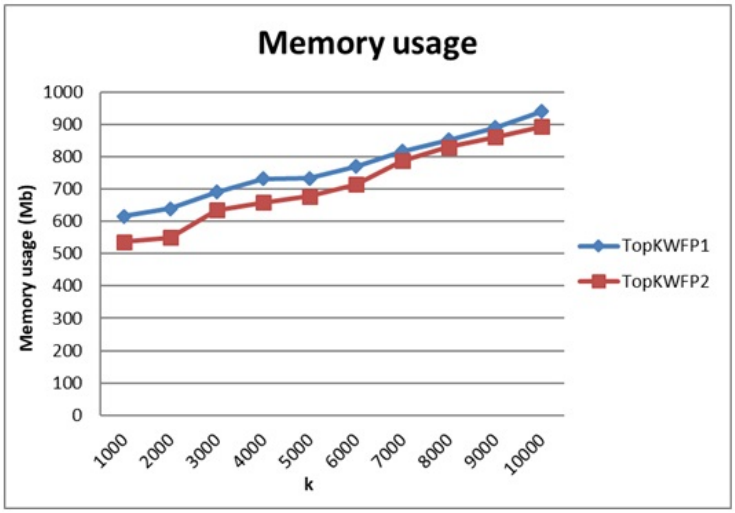


d) Levithan

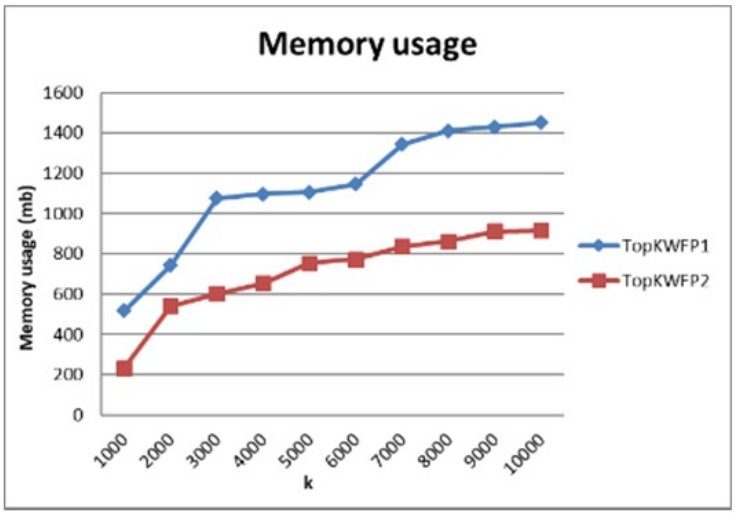


e) Sign

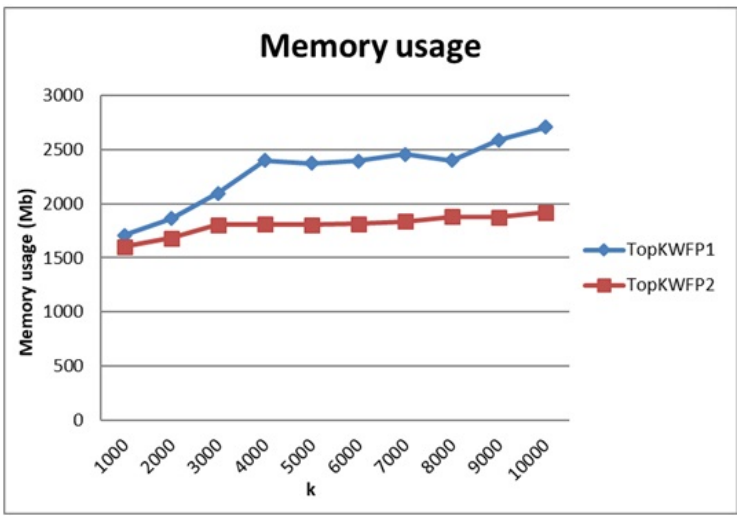
Figure 1. Runtime on Bible, BMS-WebView1, Fifa, Levithan and Sign dataset



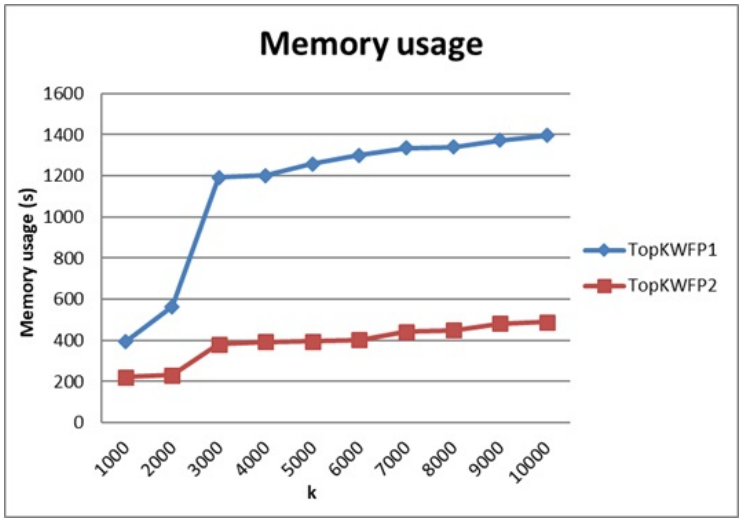
a) *Bible*



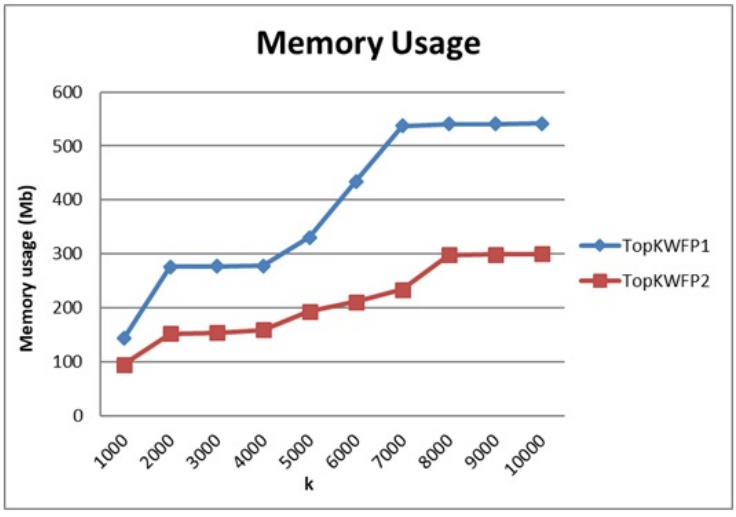
b) *BMS-WebView1*



c) *Fifa*

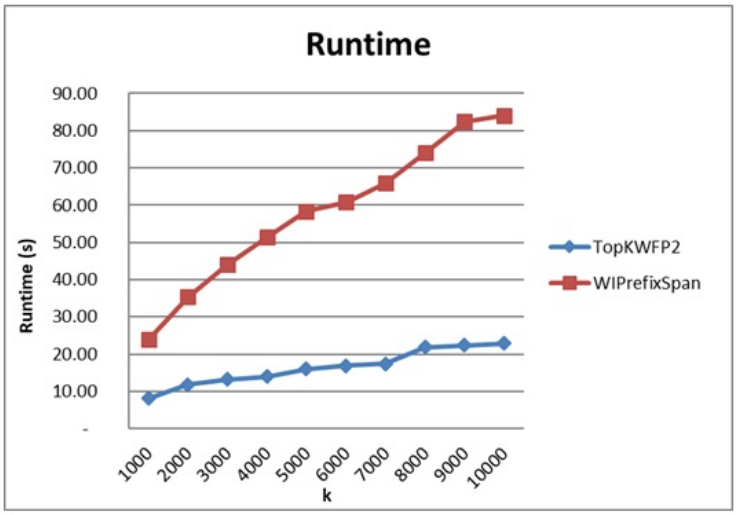


d) *Levithan*

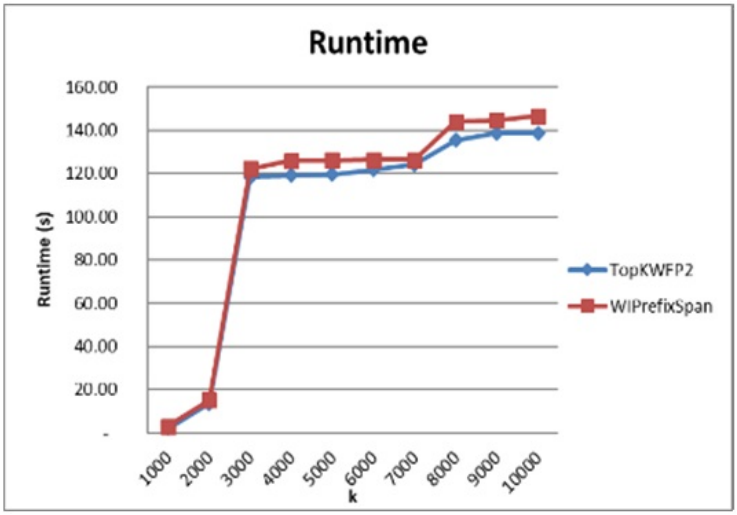


e) *Sign*

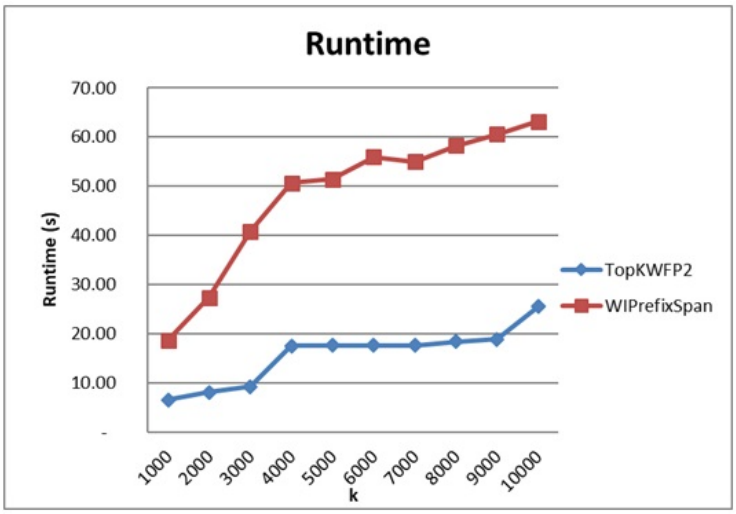
Figure 2. Memory usage on Bible, BMS-WebView1, Fifa, Levithan and Sign dataset



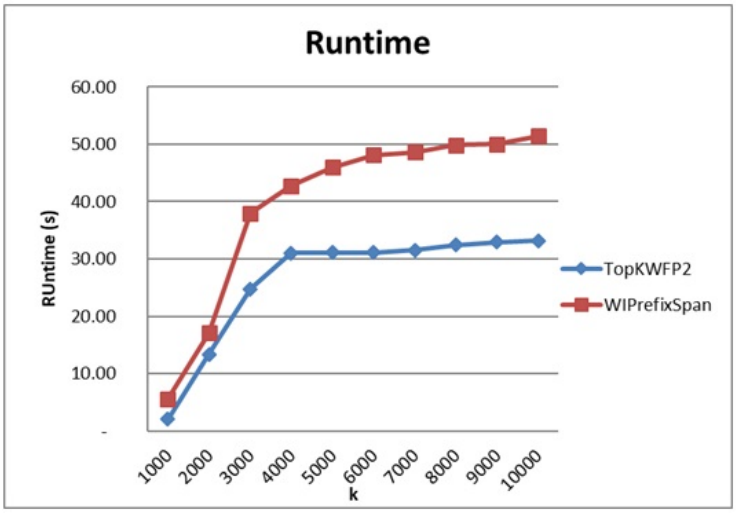
a) Bible



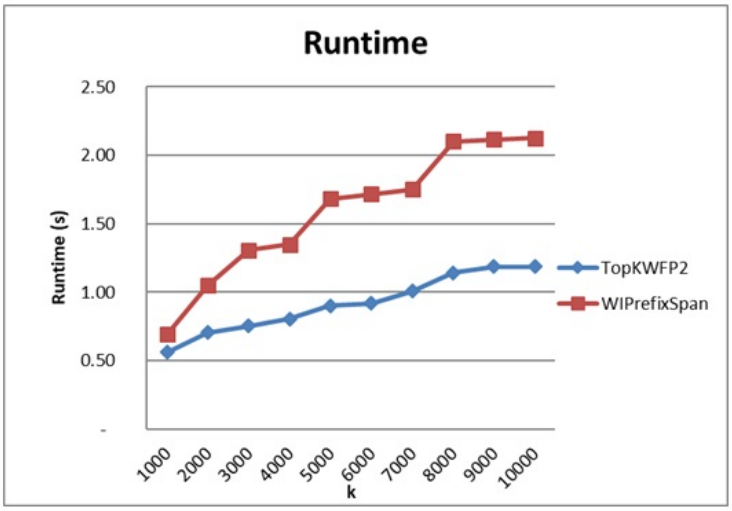
b) BMS-WebView1



c) Fifa



d) Levithan



e) Sign

Figure 3. Comparison of WIPrefixSpan and TopKWFP runtime for Bible, BMS-WebView1, Fifa, Levithan and Sign dataset

5. CONCLUSIONS

We proposed TopKWFP, an algorithm to discover the top-K item-interval weighted frequent sequential patterns having the highest weighted support, where k is set by the user. The algorithm can solve 3 problems of real life world: first, it used the weight values assigned to each item to indicate their significance; second, it extended the sequence with the item interval between items and last it can discover the top-K sequential patterns without a minimum threshold.

The TopKWFP algorithm uses 2 strategies that reduced the search space and hence increase the algorithm's performance. Our experimental study shows that the proposed algorithm delivers competitive performance and in many cases outperforms WIPrefixSpan, even when it is running with the best tuned *wminsup*.

With the above comment, we can conclude that mining top-K item-interval weighted frequent sequential patterns is practical and in many cases more preferable than the traditional minimum support threshold based sequential pattern mining.

ACKNOWLEDGMENT

This work is sponsored by a research grant from IOIT (CS.18.05 and No.12/FIRST/2a/IoIT).

REFERENCES

[1] R. Agrawal, R. Srikant, "Mining sequential patterns," in *Proceedings of the International Conference on Data Engineering (ICDE)*, 1995.

[2] R. Agrawal, R. Srikant, "Mining sequential patterns: Generalizations and performance improvements," in *Proceedings of the 5th International Conference on Extending Database Technology: Advances in Database Technology*, pp.1–17, 1996.

[3] J. Pei, J. Han, B.M. Asi, H. Pino,, "PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth," in *Proceedings of the Seventeenth International Conference on Data Engineering*, 2001.

[4] M. Zaki, "SPADE: An efficient algorithm for mining frequent sequences," *Machine Learning*, vol. 40, pp.31–60, 2000.

[5] Ayres, J., Gehrke, J., Yiu, T. and Flannick, J, "Sequential pattern mining using bitmap representation," in *Proc. of ACM SIGKDD'02*, 2002.

[6] Yu Hirate, Hayato Yamana, "Generalized sequential pattern mining with item," *Journal of Computers*, vol. 1, no. 3, pp.51–60, 2006.

[7] C.H.Cai, A.W.Chee Fu, C.H.Cheng, and W.W.Kwong, "Mining association rules with weighted items," in *Proceedings of the 1998 International Symposium on Database Engineering & Applications*, Cardiff, Wales, 1998.

[8] W.Wang, J.Yang, and P.S.Yu, "Efficient mining of weighted association rules (WAR)," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000.

[9] F. Tao, F. Murtagh, M. Farid, "Weighted association rule mining using weighted support and significance framework," in *Proceedings of 9th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, vol. 12, no. 1, pp.234–778, 2002.

[10] M. S. Khan, M. Muyeba, F. Coenen, "Weighted association rule mining from binary and fuzzy data," in *Proceedings of 8^{th} Industrial Conference, ICDM 2008*, 2008.

[11] U. Yun, J.J. Leggett, "WFIM: weighted frequent itemset mining with a weight range and a minimum weight," in *5th SIAM Int. Conf. on Data Mining*, 2005.

[12] Janos Demetrovics, Vu Duc Thi, Tran Huy Duong, "An algorithm to mine normalized weighted sequential patterns using prefix-projected database," *Serdica Journal of Computing, Sofia, Bulgarian Academy of Sciences*, vol. 2, pp.105–122, 2015.

[13] Tran Huy Duong, Vu Duc Thi, "Algorithm mining normalized weighted frequent sequential patterns with Time intervals," *Research, Development and Application on Information & Communication Technology*, vol. 2, pp.72–81, 2015.

[14] J. Wang and J. Han, TFP, "An efficient algorithm for mining top-K frequent closed itemsets," *TKDE*, vol. 17, pp.652–664, 2005.

[15] K. Chuang, J. Huang and M. Chen, "Mining top-K frequent patterns in the presence of the memory constraint," *VLDB Journal*, vol. 17, pp.1321–1344, 2008.

[16] Y. L. Cheung and A. W. Fu, "Mining frequent itemsets without support threshold: with and without item constraints," *TKDE*, vol. 16, pp.1052–1069, 2004.

[17] Sharda Khode, Sudhir Mohod, "Mining high utility itemsets using TKO and TKU to find top-K high utility web access patterns," in *2017 International Conference of Electronics, Communication and Aerospace Technology (ICECA)*, Coimbatore, 2017.

[18] P. Tzvetkov, X. Yan and J. Han, "TSP: Mining top-K closed sequential patterns," *ICDM*, pp.347–354, 2003.

[19] Z. Zheng, L. Cao, Y. Song and W. Wei, "Efficiently mining top-K high utility sequential patterns," *2013 IEEE 13^{th} International Conference on Data Mining*, pp.1259–1264, 2013.

[20] Asima Jamil, Abdus Salam and Farhat Amin, "Performance evaluation of top-K sequential mining methods on synthetic and real datasets," *International Journal of Advanced Computer Research*, vol. 7, no. 32, pp.176–184, 2017.

[21] Fournier-Viger P., Gomariz A., Gueniche T., Mwamikazi E., Thomas R, "TKS: Efficient mining of top-K sequential patterns," *Springer Advanced Data Mining and Application*, vol. 8346, pp.109–120, 2013.

[22] Karishma B Hathi , Jatin R Ambasana, "Top K sequential pattern mining algorithm," *International Conference on Information Engineering, Management and Security*, pp.115–120, 2015.

Received on August 29, 2018
Revised on October 22, 2018