Serdica J. Computing **9** (2015), No. 2, 105–122

Serdica
Journal of Computing

Bulgarian Academy of Sciences
Institute of Mathematics and Informatics

AN ALGORITHM TO MINE NORMALIZED WEIGHTED SEQUENTIAL PATTERNS USING A PREFIX-PROJECTED DATABASE*

Janos Demetrovics, Vu Duc Thi, Tran Huy Duong

Abstract. Sequential pattern mining is an important subject in data mining with broad applications in many different areas. However, previous sequential mining algorithms mostly aimed to calculate the number of occurrences (the support) without regard to the degree of importance of different data items. In this paper, we propose to explore the search space of subsequences with normalized weights. We are not only interested in the number of occurrences of the sequences (supports of sequences), but also concerned about importance of sequences (weights). When generating subsequence candidates we use both the support and the weight of the candidates while maintaining the downward closure property of these patterns which allows to accelerate the process of candidate generation.

1. Introduction. Sequential Pattern Mining is an important data mining subject with broad applications in many different areas. Sequential patterns

ACM Computing Classification System (1998): H.2.8.

Key words: data mining, frequent sequential patterns, weighted, sequential patterns.

*This work is sponsored by a research grant from Vietnam National University, Hanoi (QG.15.41).

are very popular in real life data, like customer purchase patterns, disease treatment patterns, web access patterns. The main purpose of sequential pattern mining is finding all the repeated sequence patterns in a database. Many authors have proposed to mine sequential pattern algorithms. Mining approaches include AprioriAll [1], GSP [2], PrefixSpan [3], SPADE [4], SPAM [5] that solve different types of mining problems and at the same time aim at reducing the running time and computational resources. These algorithms, however, consider only the number of occurrences of subsequences but do not take into account the important levels of data items of sequences.

In this paper, we propose to mine sequential patterns in sequence databases. We are not only interested in the number of occurrences of sequences (the supports), but also concerned about the important levels of data items of sequences (weight of sequence). However, when considering the weights of data items, we cannot utilize the downward closure property of subsequences which allows to quickly check whether a subsequence is frequent or not. To deal with this problem, we propose a redundancy method of candidate generation in which the downward closure property is reserved, and checking the support and weight of the final normalised weighted patterns.

The remainder of this paper is organized as follows. Session 2 provides a study of related works. Section 3 describes the problem and proposes the mining method for normalized weighted frequent sequential patterns named WPrefixSpan. This method is based on the PrefixSpan algorithm [3] for mining frequent sequential patterns. Section 4 presents experimental results and a comparison between WPrefixSpan and PrefixSpan algorithm [3] on the BMS-WebView dataset. Conclusions and comments are presented in the last section.

2. Related work. In 1995, Agrawal and Srikant introduced the sequence pattern mining problem [1] and proposed an algorithm called AprioriAll. This algorithm is based on the Apriori algorithm for mining sequence patterns. Like Apriori, AprioriAll scans the database multiple times to generate subsequence candidates, This approach requires long running time. In 2001 J. Pei, J. Han, B.M. Asi, H. Pinto introduced the PrefixSpan [3] algorithm which is based on the pattern growth method. This method does not require multiple database scans, so it takes considerably less mining time than AprioriAll.

Other algorithms such as SPADE[4], SPAM[5] were developed to optimize the sequence pattern mining process. Additionally, the techniques based on the dynamic bit sequence to explore the closed sequence pattern are also presented in [13].

However all the sequence mining algorithms mentioned above are not interested in the important degree of each pattern (weight of pattern). Therefore, other authors have proposed solutions for the problem of mining with sequence weights, see for example [6, 7, 8, 9, 13, 14, 15]. The algorithms [11, 14] although mining the sequence pattern on a weighted database, are not interested in the binding between the weight and the support of patterns.

3. Problem statement of mining normalized weighted frequent sequential patterns using prefix-projected databases.

3.1. Preliminaries. Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of all items. Each item $i_j \in I$ is assigned with a weight $w_j, j = 1, \ldots, n$.

A sequence $\mathbf{S}_m$ is an ordered list of itemsets denoted by $\{s_1, s_2, \ldots, s_m\}$ here $s_j \subseteq I$ is an itemset which is called an element of a sequence. $S = \{s_1, s_2, \ldots, s_m\}$ and s_j denoted by $(i_1 i_2 \ldots i_k)$ with i_t is an item in I. 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. A sequence database $S = \{S_1, S_2, \ldots, S_n\}$ is a set of tuples (sid, S) where sid is an identification of a sequence and S_k is a sequence.

Definition 1 (Support of a sequence)**.** *The support of a sequence S_a in a sequence database S is the number of occurrences of records containing the sequence S_a in S.*

Definition 2 (Normalized weight of a sequence)**.** *Let $I = \{i_1, i_2, \ldots, i_n\}$ is a set of all items. Each item $i_j \in I$ is assigned a weight $w_j, j = 1, \ldots, n$.*

Then normalized weighted of a sequence $\alpha =< e_1 e_2 \cdots e_n\rangle$ is calculated with the formula:

$$NW(\alpha) = \frac{1}{k} \sum_{i_j \in \alpha} w_j.$$

Definition 3 (Normalized weighted support of a sequence)**.** *We call $NWsupport(\alpha)$ of sequence α is the normalized weighted support of sequence α*

$$NWsupport(\alpha) = NW(\alpha) * support(\alpha) = \left(\frac{1}{k} \sum_{i_j \in \alpha} w_j \right) * SC(\alpha).$$

Definition 4. (Normalized weighted frequent sequential pattern)**.** *Given a sequence database S and a support threshold $wmin\_sup$. A sequence α is called normalized weighted frequent sequential pattern if it satisfies:*

$$NWSupport(\alpha) \geq wmin\_sup.$$

Then the problem of mining normalized weighted frequent sequential patterns is stated as follows:

- Given a sequence database S, each item $i_j \subseteq I$ is assigned a weight w_j and a support threshold $wmin\_sup$. Finding all normalized weighted frequent sequential pattern in S means finding the set L:

$$L = \{S_a \subseteq S | NWsupport\,(S_a) \geq wmin\_sup\}$$

- If a normalized weighted frequent sequential pattern does not satisfy the downward closure property then some subsets of normalized weighted frequent sequential pattern are not necessarily normalized weighted frequent sequential patterns.

3.2. The proposed solution. In this subsection we propose an algorithm to mine normalized weighted frequent sequential patterns (WPrefixSpan). Definitions 5, 6, 7, 8 and Lemma 1, Lemma 2 given here are based on those presented in PrefixSpan [3] whose main approach is to push weight constraint and support threshold in mining frequent sequential pattern and ensure downward closure property.

To avoid checking every possible combination of a potential candidate sequence, we first fix the order of items within each element. Since items within an element of a sequence can be listed in any order, without loss of generality, one can assume that they are always listed alphabetically.

For example the sequence is presented as $\langle a(abc)(ac)d(cf)\rangle$ instead of $\langle a(acb)(ac)d(fc)\rangle$. By such a convention, the expression of a sequence is unique.

If we follow the order of the prefix of a sequence and project only the postfix of a sequence, we can examine in an orderly manner all the possible subsequences and their associated projected database.

Definition 5 (Prefix)**.** *Itemsets in an element of sequence are ordered alphabetically* [3]*. Let* $\alpha = \langle e_1 e_2 \cdots e_n \rangle$ *be a sequence, a sequence* $\beta = \langle e'_1 e'_2 \cdots e'_m \rangle$ *with* $(m \leq n)$ *is a Prefix of* α *if:*

- $e'_i = e_i$ *with* $(i \leq m-1)$
- $e'_m \subseteq e_m$
- *every itemset in* $(e_m - e'_m)$ *is ordered in* e'_m

For example: Given a sequence $s = \langle a(abc)(ac)d(cf) \rangle$, then prefixes of s are: $\langle a \rangle, \langle aa \rangle, \langle a(ab) \rangle, \langle a(abc) \rangle, \ldots$.

Definition 6 (Postfix)**.** *Let* $\alpha = \langle e_1 e_2 \cdots e_n \rangle$ *be a sequence and a sequence* $\beta = \langle e'_1 e'_2 \cdots e'_{m-1} e'_m \rangle$ *with* $(m \leq n)$ *be the Prefix of* α*. The sequence* $\mu = \langle e''_m e_{m+1} \cdots e_n \rangle$ *is called Postfix of* α *with regard to Prefix* β*, denote as* $\mu = \alpha/\beta$ *with* $e''_m = (e_m - e'_m)$ *or* $\alpha = \beta.\mu$*. If* β *is not a subsequence of* α *then the Postfix of* α *with regard to* β *is null.*

For example: Given sequence $s = \langle a(abc)(ac)d(cf) \rangle$, sequence $\langle (abc)(ac)d(cf) \rangle$ is a Postfix with regard to Prefix $\langle a \rangle$; $\langle (\_bc)(ac)d(cf) \rangle$ is Postfix with regard to Prefix $\langle aa \rangle$, $\langle (\_c)(ac)d(cf) \rangle$ is Postfix with regard to Prefix $\langle a(ab) \rangle$ Using the concepts of prefix and postfix, the problem of mining Normalized weighted frequent sequential patterns can be decomposed into a set of subproblems as shown in the following lemmas:

Lemma 1.

- *Let* $\{\langle x_1 \rangle, \langle x_2 \rangle, \ldots, \langle x_n \rangle\}$ *be the complete set of length-*1 *sequential patterns in a sequence database S. The complete set of sequential patterns in S can be divided into n disjoint subsets. The i-th subset* $(1 \leq i \leq n)$ *is the set of sequential patterns with prefix* $\{x_i\}$*.*
- *Let* α *be a length-l sequential pattern and* $\{\beta_1, \beta_2, \ldots, \beta_m\}$ *be the set of all length-*$(l+1)$ *sequential patterns with prefix* α*. The complete set of sequential patterns with prefix* α*, except for* α *itself, can be divided into m disjoint subsets. The j-th subset* $(1 \leq j \leq m)$ *is the set of sequential patterns prefixed with* β_j*.*

P r o o f. We show the correctness of the second half of the lemma. The first half is a special case where $\alpha = \varnothing$.

For a sequential pattern μ with prefix α, where α is of length l, the length-$(l+1)$ prefix of μ must be a sequential pattern, according to the a priori heuristic. Furthermore, the length-$(l+1)$ prefix of μ is also a prefix of α, according to the definition of prefix. Therefore, there exists some j $(1 \leq j \leq m)$ such that β_j is the length-$(l+1)$ prefix of μ. Thus, μ is in the j-th subset. On the other hand, since the length-k prefix of a sequence μ is unique, μ belongs to only one determined subset. That is, the subsets are disjoint. So, we have the lemma. □

Based on Lemma 1, the problem can be partitioned recursively. That is, each subset of sequential patterns can be further divided when necessary. This forms a divide-and-conquer framework. To mine the subsets of sequential patterns, the corresponding projected databases can be constructed.

Definition 7 (Conditions database)**.** *Suppose we have a sequence $\alpha = \langle e_1 e_2 \cdots e_n \rangle$ belonging to a sequence database S. A α-conditions database is denoted as $S|_\alpha$, that is the collection of postfixes of sequences in S with regard to Prefix α.*

Definition 8 (Support count in projected database)**.** *Let α be a sequential pattern in sequence database S, and β be a sequence with prefix α. The support count of β in α-projected database $S|_\alpha$, denoted as $support_{S_\alpha}(\beta)$, is the number of sequences μ in $S|_\alpha$ such that $\beta \sqsubseteq \alpha.\mu$.*

Lemma 2. *Let α and β be two sequential patterns in a sequence database S such that α is a prefix of β.*

- $S|_\beta = (S|_\alpha)|_\beta$
- *For any sequence μ with prefix α, $support_S(\mu) = support_{S|_\alpha}(\mu)$.*
- *The size of α-conditions database cannot exceed that of S.*

Definition 9 (Candidate sequence pattern)**.** *Given a support threshold wmin_sup, a sequence α is called candidate weighted sequence pattern if it satisfies:*

$$Support(\alpha) * MaxW \geq wmin\_sup$$

where MaxW is the maximum value of weights of items in S. Candidate sequence

patterns are built for the purpose of pruning the search space and still ensure the downward closure property in mining normalized weighted frequent sequential patterns.

Then, the problem of mining normalized weighted frequent sequential patterns using the prefix condition database is stated as:

- Given a sequence database S, each item $i_j \subseteq I$ is assigned a weight w_j, a support threshold $wmin\_sup$. We have to find all normalized weighted frequent sequence patterns in S, which means finding the set L:

$$L = \{S_a \subseteq S | NW support\,(S_a) \geq wmin\_sup$$

- The normalized weighted frequent sequence pattern doesn't satisfy the downward closure property, which mean a subset of a normalized weighted frequent sequence pattern is not necessarily a normalized weighted frequent sequence pattern.

3.3. Example of mining normalized weighted frequent sequential patterns using a prefix condition database. Given a sequence database S in Table 1, weighted values of items in Table 2, $wmin\_sup = 2$, then mining normalized weighted frequent sequential patterns in sequence database S with WPrefixSpan method is done according to the following steps:

Table 1. Sequence database S

Data sequences
$\langle a(abc)(ac)d(cf)\rangle$
$\langle (ad)c(bc)(ae)bc\rangle$
$\langle (ef)(ab)(df)cb\rangle$
$\langle eg(af)cbc\rangle$
$\langle a(ab)(cd)egh\rangle$
$\langle a(abd)bc\rangle$

Table 2. Weight of items

Item	**Weight**
a	0.9
b	0.75
c	0.8
d	0.85
e	0.75
f	0.7
g	0.85
h	0.8

Step 1: Find all normalized weighted frequent sequence pattern candidate of length-1. Scan S for the first time to find all candidates for normalized weighted frequent sequence patterns whose length is 1, then count the support of each item.

A length-1 item may not be a normalized weighted frequent sequence pattern, but it can be combined with other items with higher support or weighted values to become a normalized weighted frequent sequence pattern of greater length.

Then we have support count for each item as follows:

$\langle a\rangle : 6$, $\langle b\rangle : 6$, $\langle c\rangle : 6$, $\langle d\rangle : 5$, $\langle e\rangle : 4$, $\langle f\rangle : 3$, $\langle g\rangle : 2$, $\langle h\rangle : 1$

Weighted values: (a : 0.9; b : 0.75; c : 0.8; d : 0.85; e : 0.75; f : 0.7; g : 0.85, h : 0.8)

$MaxW = 0.9$

Following Definition 9, we eliminate g and h because $support(g)*MaxW = 2 * 0.9 = 1.8 < wmin\_sup$ and $support(h) * MaxW = 1 * 0.8 = 0.8 < wmin\_sup$.

Then we have length-1 candidates of normalized weighted frequent sequence patterns:

$$C_1 = \langle a\rangle, \langle b\rangle, \langle c\rangle, \langle d\rangle, \langle e\rangle, \langle f\rangle$$

Following Definition 4 with candidates in C_1, we obtain length-1 normalized weighted frequent sequence patterns as:

$NWsupport(a) = 6 * 0.9 = 5.4$ $\quad$ $NWsupport(b) = 6 * 0.75 = 4.5$

$NWsupport(c) = 6 * 0.8 = 4.8$ $\quad$ $NWsupport(d) = 5 * 0.85 = 4.25$

$NWsupport(e) = 4 * 0.75 = 3$ $\quad$ $NWsupport(f) = 3 * 0.7 = 2.1$

$$L = \langle a\rangle, \langle b\rangle, \langle c\rangle, \langle d\rangle, \langle e\rangle, \langle f\rangle$$

Step 2: Divide search space. All candidates and normalized weighted frequent sequential patterns are mined in 6 regions corresponding to the following 6 prefixes:

(1) Sequence pattern with prefix $\langle a\rangle$

(2) Sequence pattern with prefix $\langle b\rangle$

(3) Sequence pattern with prefix $\langle c\rangle$

(4) Sequence pattern with prefix $\langle d\rangle$

(5) Sequence pattern with prefix $\langle e\rangle$

(6) Sequence pattern with prefix $\langle f\rangle$

Step 3: Mining subset candidates and normalized weighted sequence pattern. All subset candidates and normalized weighted frequent sequence patterns are mined by building condition databases corresponding to the prefix and mining them with recursive method. The steps are as follows:

A. Finding candidates and normalized weighted frequent sequence patterns with prefix $\langle a \rangle$.

If a sequence contains $\langle a \rangle$, only the subsequence prefixed with the first occurrence of $\langle a \rangle$ should be considered. For example, in sequence $\langle a(ac)(abc)d(cf) \rangle$ only the subsequence $\langle (abc)(ac)d(cf) \rangle$ should be considered, similarly with $\langle (ad)c(bc)(ae)bc \rangle$ only the subsequence $\langle (\sim d)c(bc)(ae)bc \rangle$ should be considered. Then condition database with prefix $\langle$a$\rangle$ includes 6 sequences:

Table 3. Condition database with prefix $\langle a \rangle$

iSID	**Sequence data**
1	$\langle (abc)(ac)d(cf) \rangle$
2	$\langle (\sim d)c(bc)(ae)bc \rangle$
3	$\langle (\sim b)(df)cb \rangle$
4	$\langle (\sim f)cbc \rangle$
5	$\langle (ab)(cd)e \rangle$
6	$\langle (abd)bc \rangle$

By scanning the condition database with prefix $\langle a \rangle$, the support counts of all items are: $a : 4$, $b : 6$, $c : 6$, $d : 4$, $e : 2$, $f : 2$, $(\sim b) : 4$, $(\sim d) : 1$, $(\sim e) : 1$ and $(\sim f) : 1$.

Following Definition 9, we find all length-2 candidate patterns with prefix $\langle$a$\rangle$:

$support(a) * MaxW = 4 * 0.9 = 3.6$ $\quad$ $support(b) * MaxW = 6 * 0.9 = 5.4$
$support(c) * MaxW = 6 * 0.9 = 5.4$ $\quad$ $support(d) * MaxW = 4 * 0.9 = 3.6$
$support(e) * MaxW = 2 * 0.9 = 1.8$ $\quad$ $support(f) * MaxW = 2 * 0.9 = 1.8$
$support((\sim b)) * MaxW = 4 * 0.9 = 3.6$ $\quad$ $support((\sim d)) * MaxW = 1 * 0.9 = 0.9$
$support((\sim e)) * MaxW = 1 * 0.9 = 0.9$ $\quad$ $support((\sim f)) * MaxW = 1 * 0.9 = 0.9$

Eliminated itemsets are: $\langle e \rangle$, $\langle f \rangle$, $\langle (\sim d) \rangle$, $\langle (\sim e) \rangle$, $\langle (\sim f) \rangle$.

Length-2 candidate patterns with prefix $\langle a \rangle$ which satisfy support counts with maximum weight are:

$$C_2\langle a \rangle = \langle aa \rangle, \langle ab \rangle, \langle ac \rangle, \langle ad \rangle, \langle (ab) \rangle.$$

Following Definition 4 with candidates in $C_2\langle a \rangle$, we have length-2 normalized weighted sequential patterns with prefix $\langle a \rangle$ are:

$$NWsupport(aa) = 4 * (0.9 + 0.9)/2 = 5.4$$
$$NWsupport(ab) = 6 * (0.9 + 0.75)/2 = 4.95$$
$$NWsupport(ac) = 6 * (0.9 + 0.8)/2 = 5.1$$
$$NWsupport(ad) = 4 * (0.9 + 0.85)/2 = 3.5$$
$$NWsupport((ab)) = 4 * (0.9 + 0.9 + 0.75)/3 = 3.4$$
$$L = L \cup \{\langle aa\rangle, \langle ab\rangle, \langle ac\rangle, \langle ad\rangle, \langle (ab)\rangle\}$$

According to the recursive nature, candidates and normalized weighted frequent sequence patterns with prefix $\langle a\rangle$ will be further divided into 5 regions corresponding to 5 prefixes including:

(1) Sequence pattern with prefix $\langle aa\rangle$
(2) Sequence pattern with prefix $\langle ab\rangle$
(3) Sequence pattern with prefix $\langle ac\rangle$
(4) Sequence pattern with prefix $\langle ad\rangle$
(5) Sequence pattern with prefix $\langle (ab)\rangle$

A.1. With sequence pattern with prefix $\langle aa\rangle$, we build a condition database with prefix $\langle$aa$\rangle$ with items in candidate set in $C_2\langle a\rangle$. Then

Table 4. Condition database with prefix $\langle aa\rangle$

iSID	**Sequence data**
1	$\langle(\sim bc)(ac)dc\rangle$
2	$\langle bc\rangle$
5	$\langle(\sim b)(cd)\rangle$
6	$\langle(\sim bd)bc\rangle$

By scanning the condition database with prefix $\langle aa\rangle$, the support counts of all items are: $a:1$, $b:2$, $c:4$, $d:2$, $(\sim b):3$, $(\sim c):1$.

Following Definition 9, we find all length-3 candidate patterns with prefix $\langle aa\rangle$:

$support(a) * MaxW = 1 * 0.9 = 0.9$ $\quad$ $support(b) * MaxW = 2 * 0.9 = 1.8$
$support(c) * MaxW = 4 * 0.9 = 3.6$ $\quad$ $support(d) * MaxW = 2 * 0.9 = 1.8$
$support((\sim b)) * MaxW = 3 * 0.9 = 2.7$ $\quad$ $support((\sim c)) * MaxW = 1 * 0.9 = 0.9$

Eliminated items are $\langle a\rangle$, $\langle b\rangle$, $\langle (d)\rangle$, $\langle (\sim c)\rangle$

Length-3 candidates of normalized weighted sequence pattern with prefix $\langle aa\rangle$ are:

$$C_3\langle aa\rangle = \langle aac\rangle, \langle a(ab)\rangle.$$

Following Definition 4 with candidates in $C_3\langle aa\rangle$, we have length-3 normalized weighted frequent patterns with prefix $\langle aa\rangle$:

$NWsupport(aac) = 4 * (0.9 + 0.9 + 0.8)/3 = 3.46$
$NWsupport(a(ab)) = 3 * (0.9 + 0.9 + 0.75)/3 = 2.55$

$$L = L \cup \{\langle aac\rangle, \langle a(ab)\rangle\}.$$

According to the recursive nature, candidates and normalized weighted frequent sequence patterns with prefix $\langle aa\rangle$ will be further divided into 2 regions corresponding to 2 prefixes including:

(1) Sequence pattern with prefix $\langle aac\rangle$
(2) Sequence pattern with prefix $\langle a(ab)\rangle$

A.1.1. With sequence pattern with prefix $\langle aac\rangle$, we build the condition database with prefix $\langle aac\rangle$ with items in candidate set in $C_3\langle aa\rangle$.

The condition database with prefix $\langle aac\rangle$ has one sequence c.

A.1.2. With sequence pattern with prefix $\langle a(ab)\rangle$, we build condition database with prefix $\langle a(ab)\rangle$ with items in candidate set in $C_3\langle aa\rangle$.

Then condition database with prefix $\langle aac\rangle$ has sequences:

Table 5. Condition database with prefix $\langle a(ab)\rangle$

iSID	**Sequence data**
1	$\langle(\sim c)(ac)c\rangle$
5	$\langle c\rangle$
6	$\langle(\sim d)bc\rangle$

By scanning the condition database with prefix $\langle a(ab)\rangle$, support counts of all items are: $a : 1$, $b : 1$, $c : 3$, $(\sim d) : 1$, $(\sim c) : 1$.

Following Definition 9, we find all length-4 candidate patterns with prefix $\langle a(ab)\rangle$:

$support(a) * MaxW = 1 * 0.9 = 0.9$ $\quad support(b) * MaxW = 1 * 0.9 = 0.9$
$support(c) * MaxW = 3 * 0.9 = 2.7$ $\quad support((\sim d)) * MaxW = 1 * 0.9 = 0.9$
$support((\sim c)) * MaxW = 1 * 0.9 = 0.9$

Eliminated items are $\langle a\rangle$, $\langle b\rangle$, $\langle(\sim d)\rangle$, $\langle(\sim c)\rangle$.

The length-4 candidates of normalized weighted sequence pattern with prefix $\langle a(ab)\rangle$ are:

$$C_4\langle a(ab)\rangle = \langle a(ab)c\rangle$$

Following Definition 4 with candidates in $C_4\langle a(ab)\rangle$, we have length-4 normalized weighted frequent patterns with prefix $\langle a(ab)\rangle$:

$$NWsupport(a(ab)c) = 3 * (0.9 + 0.9 + 0.75 + 0.8)/4 = 2.51$$

$$L = L \cup \{\langle a(ab)c\rangle\}.$$

According to the recursive nature, candidates and normalized weighted frequent sequence patterns with prefix $\langle a(ab)\rangle$ will be further divided into one region corresponding to one prefix including:

(1) Sequence pattern with prefix $\langle a(ab)c\rangle$

A.1.2.1. With the sequence pattern with prefix $\langle a(ab)c\rangle$, we build the condition database with prefix $\langle a(ab)c\rangle$ with items in candidate set in $C_4\langle a(ab)\rangle$. The condition database with prefix $\langle a(ab)c\rangle$ has one sequence $\langle (ac)c\rangle$.

A.2. With the sequence pattern with prefix $\langle ab\rangle; \langle ac\rangle; \langle ad\rangle and \langle (ab)\rangle$, we build the condition database with prefix corresponding with items in candidate set in $C_2\langle a\rangle$. Normalized weighted frequent sequence pattern mining corresponding to each prefix is performed similarly to step A.1.

B. Finding candidates and normalized weighted frequent sequence patterns with prefix $\langle b\rangle$, $\langle c\rangle$, $\langle d\rangle$, $\langle e\rangle, \langle f\rangle$. With the sequence pattern with prefix $\langle b\rangle$; $\langle c\rangle$; $\langle d\rangle$; $\langle e\rangle$; $\langle f\rangle$, we build the condition database with prefix corresponding with items in candidate set in C_1. Normalized weighted frequent sequence pattern mining corresponding to each prefix is also performed similarly to step A with recursive method.

Step 4: The result is the normalized weighted frequent sequence pattern in the sequence database S. Normalized weighted frequent sequence patterns are mined in turn in a recursive process for each prefix. In this method, the number of results will be less than the number sequence patterns without data item weights.

3.4. Mining sequential pattern with normalized weighted using a prefix condition database algorithm (WPrefixSpan).

– Input:

(1) Sequence database SDB

(2) Support threshold: *wmin_sup*

(3) Weight of items: w_i

– Output: Normalized weighted frequent sequential patterns L

WPrefixSpan algorithm
Start

1) Scan database S first time, calculate support and find length-1 candidates satisfying: Itemset P is a candidate itemset if it satisfies condition 1.1, P will be put into C_k

Condition 1.1. $Support(P) * MaxW \geq wmin\_sup$

2) Check candidate itemsets in C_k, find frequent sequence patterns with normalized weights satisfying $support(P) * NW(P) \geq wmin\_sup$, put into L

3) Loop with every normalized weighted itemset P, in sequence database S

Execute recursive function $WPrefixSpan(\langle P\rangle, l, S|_P)$

End loop

End.
Function $WPrefixSpan(\alpha, l, S|_\alpha)$
Parameters:

(1) α is a candidate itemset with normalized weighted which satisfies Condition 1.1

(2) l is length of α

(3) S_α is condition database with prefix α. S is the situation when $\alpha = \oslash$

Start:

1) Scan $S|\alpha$, calculate support of each item and find candidate itemsets with normalized weighted denoted as β in sequences. A candidate β which satisfies condition 1.1 is put into L

Condition 1.1. $support(\beta) * MaxW \geq wmin\_sup$

– with each β in C_k:

(a) β can be appended to α to create a candidate for a normalized weighted frequent sequence pattern or

(b) $\langle\beta\rangle$ can be joined to α to create a candidate for a normalized weighted frequent sequence pattern

– Check candidates β in C_k to find a normalized weighted frequent sequence pattern which satisfies $support(\beta) * NW(\beta) \geq wmin\_sup$, put into L.

2) Loop with each normalized weighted frequent sequence candidate pattern β

– Join β with α to create a normalized weighted frequent sequence candidate pattern α' and the output is α'

3) Loop with each α'.

– Build a condition database with prefix α' denoted as $S|'_\alpha$

– Execute recursively function $WPrefixSpan(\alpha', l+1, S|_{\alpha'})$

End.

4. Algorithm complexity and experimental results.

4.1. Complexity of WPrefixSpan algorithm. In the general case, the complexity of the recursive algorithm is exponential. Specifically: Problem $P(n)$ has the data size n, the complexity of the problem P is called $O(n)$

– In first recursive call $O(n) = n * O(n-1)$

– In second recursive call $O(n-1) = (n-1) * O(n-2)$

– ...

Thus, the complexity in the general case is $O(n) = n * (n-1) * (n-2) * \ldots * 1 = O(n^n)$

In the specific case of the WPrefixSpan algorithm:

– First database scans to find length-1 prefix, the complexity is $O(n)$; here n is the data size (number of sequences and maximum length of the sequences)

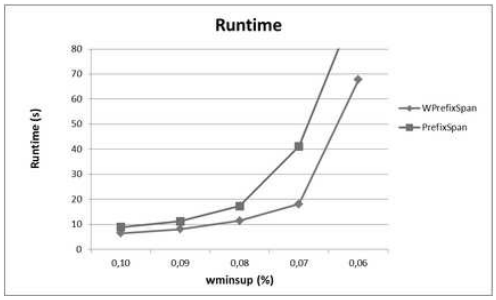
– Second scan, calling recursively with prefix length 1, the complexity is $n * O(n-1)$ (the data size is reduced by 1)

– And as above, the algorithm complexity in the general case is exponential $O(n^n)$.

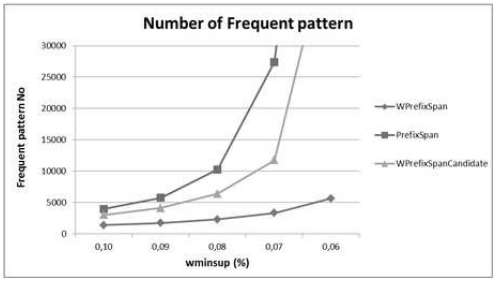
So the WPrefixSpan algorithm just tries to reduce the data space to find normalized weighted frequent patterns according to set targets, evaluating the effectiveness based on the experimental results.

4.2. Experimental results. In this section, we present the experimental results and the comparison between WPrefixSpan algorithm and PrefixSpan algorithm on a dataset from UCI Machine Learning. The dataset is BMS-WebView with 59601 sequence data, 497 data items, average length of one sequence 2.42 data items, with some long sequences (more than 318 sequences store more than 20 items) (`http://www.philippe-fournier-viger.com/spmf/datasets/BMS1_spmf`). Weighted values of items in WPrefixSpan algorithm are in the range $0,2 \leq w_j \leq 0.9$. WPrefixSpan is a mining normalized weighted frequent sequence pattern algorithm which is interested in the balance between 2 elements: weight and support of sequences. PrefixSpan is only interested in support of sequences.

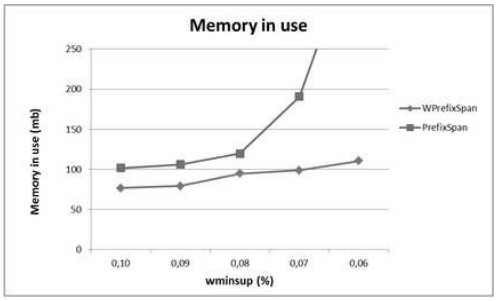
All the experiments were performed on an Intel Core2 Dual 2.53GHz PC with 3 GB main memory, running Microsoft Windows XP SP3. All the programs



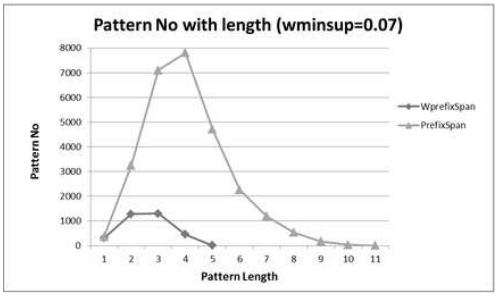
(a) Runtime



(b) Frequent Sequential pattern



(c) Memory in use



(d) Sequence pattern length

Fig. 1. Experimental result

were written in Java 1.6 using Eclipse IDE.

The experimental results show:

- Figure 1.a: we can see WPrefixSpan algorithm is more effective than PrefixSpan [3] algorithm; when wmin_sup is reduced WPrefixSpan's runtime is reduced faster than PrefixSpan's.
- Figure 1.b: the number of frequent sequential pattern found in WPrefixSpan algorithm is lower than that of PrefixSpan. WPrefixSpan adds constraints between weight and support of sequences while PrefixSpan only cares about support. The number of candidate patterns is less too according to definition 9 which prune sequences, then the search space is reduced in WPrefixSpan algorithm.
- Figure 1.c: WPrefixSpan is more effective than PrefixSpan in memory usage, because search space is reduced in WPrefixSpan.
- Figure 1.d: With the same support threshold wmin_sup =0.7, length of frequent sequence pattern in WPrefixSpan is shorter than that of PrefixSpan.

5. Conclusions. In this paper, we develop an algorithm called WPrefixSpan which detects normalized weighted frequent sequence patterns based on the candidate pattern growth model. With this approach, our algorithm doesn't need to create candidate sequence patterns like other Apriori [1] approaches.

We use a prefix condition database building method which allows to significantly reduce the search space when mining the frequent sequence patterns.

By adding weighted values of items in the sequence database, we are interested in the constraints between support count and weights; besides, in the process of building the prefix conditions database, we check the condition to prune items which do not belong to candidate patterns, which in turn reduce significantly the search space but still ensure the downward closure property of the algorithm.

With the above comments, we can conclude that WPrefixSpan is an efficient algorithm for mining normalized weighted frequent sequence patterns.

REFERENCES

[1] Agrawal R., R. Srikant. Mining sequential patterns. In: Proceedings of the Eleventh International Conference on Data Engineering (ICDE), Taipei, 6–10 March 1995, 3–14.

[2] Agrawal R., R. Srikant. Mining sequential patterns: generalizations and performance improvements. In: Proceedings of the International Conference on Extending DataBase Technology (EDBT), Lecture Notes in Computer Science, Vol. **1057** (1996), 3–17.

[3] Pei J., 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, 215–224.

[4] Zaki M. An Efficient Algorithm for Mining Frequent Sequences. *Machine Learning*, Vol. **40** (2000), 31–60.

[5] Ayres J., J. Gehrke, T. Yiu, J. Flannick. Sequential Pattern Mining using Bitmap Representation. In: Proceedings of ACM SIGKDD02, 2002, 429–435.

[6] Khan M. S., M. Muyeba, F. Coenen. Weighted Association Rule Mining from Binary and Fuzzy Data. In: Proceedings of 8th Industrial Conference, ICDM 2008, 200–212.

[7] Tao F., 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, 2003, 661–666.

[8] Yun U. An efficient mining of weighted frequent patterns with length decreasing support constraints. *Knowledge-Based Systems*, **21** (2008), No 8, 741–752.

[9] Yun U., 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, 636–640.

[10] Hirate Y., H. Yamana. Generalized Sequential Pattern Mining with Item Intervals. *JCP*, **1** (2006), No 3, 51–60.

[11] Lan G. C., T. P. Hong, H. Y. Lee. An efficient approach for finding weighted sequential patterns from sequence databases. *Applied Intelligence*, **41** (2014), No 2, 439–452.

[12] Tran M. T., B. Le, B. Vo. Combination of dynamic bit vectors and transaction information for mining frequent closed sequences efficiently. *Engineering Applications of Artificial Intelligence*, **38** (2015), 183–189.

[13] Vo B., F. Coenen, B. Le. A new method for mining Frequent Weighted Itemsets based on WIT-trees. *Expert Systems with Applications*, **40** (2013), No 4, 1256–1264.

[14] Yun U., G. Pyun, E. Yoon. Efficient Mining of Robust Closed Weighted Sequential Patterns Without Information Loss. *International Journal on Artificial Intelligence Tools*, **24** (2015), No 1, 28 pages.

[15] Yun U., K. H. Ryu. Approximate weighted frequent pattern mining with/without noisy environments. *Knowledge-Based Systems*, **24** (2011), No 1, 73–82.

Janos Demetrovics
Institute for Computer and Control (SZTAKI)
Hungarian Academy of Sciences
Budapest, Hungary e-mail: demetrovics@sztaki.mta.hu

Vu Duc Thi
Information Technology Institute
Vietnam National University (VNU)
Hanoi, Vietnam
e-mail: vdthi@vnu.edu.vn

Tran Huy Duong
Institute of Information Technology
Vietnam Academy of Science and Technology (VAST)
Hanoi, Vietnam
e-mail: huyduong@ioit.ac.vn

Received May 11, 2015
Final Accepted December 11, 2015