Weighted Contiguous Sequential Pattern Mining

Tingfu Zhou[1], Wensheng Gan[2,3], and Zhenlian Qi[4*]

[1] Jinan University of Birmingham Joint Institute, Jinan University, Guangzhou, China
[2] College of Cyber Security, Jinan University, Guangzhou 510632, China
[3] Pazhou Lab, Guangzhou 510330, China
[4] Guangdong Eco-Engineering Polytechnic, Guangzhou 510520, China
Email: tfzhou@gmail.com, wsgan001@gmail.com, qzlhit@foxmail.com

***Abstract*—In real-life, big data is contiguous, such as traffic flow and network flow, and thus some contiguous mining algorithms have been developed. It has been noticed that the significance of data (e.g., DNA sequences) is often different, and the real data may have various weights. However, the continuity of the mined data is not fully considered in the existing weighted mining algorithms. In this study, we are the first to formulate the problem of mining weighted contiguous sequential patterns and propose a new algorithm named WCSpan. Based on the usage of modified prefix pattern expansion and a tight weighted upper-bound model, we proved that WCSpan can efficiently mine the weighted contiguous sequential patterns. Experimental results showed that compared with existing similar algorithms, the proposed algorithm has advantages in execution time and memory usage. Besides, the integrity of the outcome patterns of WCSpan is preserved while data omission is avoided. In addition, the generation of patterns by the WCSpan method is faster than other methods, where the weighted upper-bound model can prune redundant candidates precisely to save memory. Both of them significantly improve the performance of WCSpan.**

***Index Terms*—data mining, contiguous pattern, weight sequential pattern, upper bound.**

I. Introduction

The purpose of data mining [1], [2] is to dig out the desired information from the original dataset. This information can be the underlying data patterns. In general, sequential pattern mining (SPM) [3], [4] and association rule mining (ARM) [5] are two of the most attractive fields in data mining. Compared with traditional data mining that mines discrete patterns, contiguous data mining can maintain the continuity of original data. This feature enables contiguous data mining to have a broad application prospect in the research of biological genetic fragments, traffic data mining, and network traffic mining. However, in practical application, the importance of data is often different. For example, in virus analysis, the genes of a virus have four different types of nitrogenous bases, they are A, T, C, and G. These four kinds of bases can form contiguous gene segments, such as Seq_1: "$...ACGGCT...$" and Seq_2: "$...ATGAGG...$". These contiguous segments can record genetic information about traits, while different traits affect the transmission of the virus to varying degrees. In this case, Seq_1 records the genetic information of trait t_1 whose degree of influence is w_1, and Seq_2 records the genetic information of trait t_2 whose degree of influence is w_2. And trait t_1 evolves from trait t_2. For the virus that has trait t_1, its transmission capacity is significantly more powerful than the virus with trait t_2. That is $w_1 \gg w_2$. Thus, these weighted values, w_1 and w_2, need to be considered when mining the potential pathogenic genes. In order to effectively capture this kind of feature, it is necessary to put forward a contiguous data mining algorithm that can also measure the weight of patterns behind massive data.

Many contiguous pattern mining algorithms have been proposed, such as the CCSpan algorithm [6] which generates candidates by splitting sequences, the CCPM algorithm [7] which mines sequential patterns based on the PrefixSpan method [8], and the Apriori-like method [9]. However, they face several challenges because too many invalid candidate sets are generated, and the candidate generation processes are quite time-consuming. In addition, the concept of weight is absent in these algorithms. Another challenge for mining sequential patterns is the weighted-based pruning strategy, which can reduce the range of potential objective patterns. The IUA algorithm [10] introduced a tighter upper-bound. It overcomes the problem of inefficiency caused by the traditional pruning strategies that take the maximal weight in the sequence set as a weighted upper-bound for each sequence. However, the IUA algorithm also faces the challenge that too many candidate sets need to be calculated, and it doesn't take contiguous factor from the sequence data into account.

Hence, to face these challenges and solve the problem of lacking an appropriate weighted contiguous pattern mining algorithm, this paper proposes a Weighted Contiguous Sequential Pattern mining algorithm (WCSpan) using prefix pattern growth and a projected database based on the PrefixSpan algorithm [8]. Moreover, inspired by the IUA algorithm [10], a weighted upper-bound model is designed to find the set of weighted contiguous sequential patterns (WCSP). The main contributions of our work are summarized as follows.

- To our best knowledge, we are the first to formulate the problem of mining weighted contiguous sequential patterns. Then we proposed the WCSpan algorithm that can discover the objective pattern correctly and efficiently. In addition, we proved that it can guarantee the integrity and closure of the patterns in sequences.

*Corresponding author.

- We modified the existing weighted pruning strategies to accommodate the weighted contiguous sequential patterns. Moreover, we improved the contiguous candidate-generated method to reduce the generation and calculation of candidates.
- We conducted several experiments on several real datasets to examine the effectiveness and efficiency of the WCSpan algorithm. They showed that WCSpan can basically maintain the high efficiency of the original algorithm based on adding the new weighted contiguous mining function. Moreover, the discovered patterns have validity.

The remainder of this article is organized as follows: Section II briefly introduces the related work. Section III introduces the related definition and the description of the addressed problem. Section IV introduces the proposed WCSpan algorithm. The experimental results and analysis are presented in detail in Section V. Finally, the summary is discussed in Section VI.

II. Related Work

A. Contiguous sequential pattern mining

Sequential pattern mining (SPM) [3], [11] has been widely studied before. Contiguous data mining is the mining of useful patterns that exceed a certain threshold value under the condition of preserving data continuity. Contiguous sequential pattern mining was first proposed by Chen and Cook [12]. Recently, some contiguous pattern mining algorithms have been designed. Yang *et al.* [6] proposed the CCSpan algorithm for mining closed contiguous sequential patterns, which generates candidate sets by splitting the original sequences. This method generates contiguous sub-sequential candidates of different lengths and then prunes invalid candidates by snippet pruning, pre-post-subsequential pruning, and support checking. Moreover, it uses super-pattern checking to guarantee closure. Based on the splitting method of CCSpan, some algorithms were proposed [13], [14]. Different from the splitting method, CSP algorithm [9] uses another contiguous sequential pattern generated method. It is modified from the GSP algorithm [15] that is based on the Apriori method [16] with gap constrains. Hence, it inherits the same weakness of the Apriori-like methods, that is, many irrelevant candidates slow down the efficiency.

In order to solve the problem of irrelevant candidates during the generation process, the CCPM algorithm [7] utilizes the PrefixSpan method to overcome this drawback. And the BP-CCSM algorithm [17] of Yang *et al.* also applies the prefix-tree structure to extract protocol specifications. By matching the prefix-tree and the two reverse order trees, it can construct a tire tree that directly reflects the desired contiguous sequential patterns (CSPs). Compared to the CCSpan algorithm, the BP-CCSM algorithm based on FP-tree reduces the generation of some redundant candidate sets. However, the generation of multiple trees and the multiple computations of each tree node is also time-consuming. Based on PrefixSpan, the C3RO algorithm [18] can discover contiguous sequential patterns with two constraints, namely robustness and extended-closure. It adopts the prefix and projected database for pattern-growth to find the CSPs in the noise database. Nevertheless, all of these algorithms treat the weight of the data in the same way. They don't consider the weight of the data that is important in real practice.

B. Weighted sequential pattern mining

In real life, the significance of different items is varied [19], [20]. Weight in pattern mining was first proposed by Ramkumar *et al.* [21]. In 2003, Tao *et al.* [22] proposed a method for mining weighted patterns by using the mean value of the items' weights (tw) in the transaction and the weighted support (ws). w_s is defined by the quotient of the sum of the transaction's weight values that contains the candidate over the total transaction weights in the whole database. Then, Yun *et al.* [23] introduced a new topic called weighted sequential pattern mining (WSPM). However, the weighted support does not fit the anti-monotone property. To solve this problem, Yun and Leggett [23] use the maximum weight of the items in the database as the weighted upper-bound value of each transaction, which also constructs a downward-closure property in WSPM. However, some transactions may not contain the maximum weighted value items. Thus, the IUA algorithm [10] uses a tightened upper-bounds model of weighted support to improve the performance of the pruning strategies, which is well performed in experiments. The weighted upper-bound model is widely used in mining weighted sequential patterns in different areas like dynamic data [24], robust pattern [25], uncertain data [26], and so on.

III. Preliminaries and Problem Formulation

To better explain weighted contiguous sequential patterns, we assume that the original sequence is given in Table I. Each sequence contains two parts: the sequence identification (SID) and the items contained in each sequence. The weight of each individual term is given as $\{A$: 0.23, B: 0.75, C: 0.12, D: 0.88, E: 0.30, F: 0.22$\}$. The relevant definitions for weighted contiguous sequential pattern mining (WCSPM) are given below.

Let $I = \{i_1, i_2, i_3, ..., i_n\}$ be a set of items. A term is the smallest unit that cannot be divided. An item is represented by an integer or a character. Let sequence S be an ordered list of items. Here, the order refers to the order in which events occur. The functionality of itemsets can here be covered by the functionality of sequences and subsequences. Therefore, the concept of itemsets is not included in the original data in this article.

*Definition 1 (**Contiguous sequential candidate**):* The contiguous sequential candidate is a fragment of a sequence in the database. It has the continuity characteristic that if an item A in the candidate set has an adjacent former item or adjacent subsequent item, then these two items are exactly the same items in the original database. And its length is greater or equal to one item.

For example, $Seq_1 = \{C\ D\ E\ F\ G\}$ and $c_1 = \{D\ E\ F\}$, thus c_1 is a contiguous sequential candidate of Seq_1. A

TABLE I: Four sequences in running example

SID	Sequence
Seq1	$A\ B\ C\ E\ F\ D$
Seq2	$B\ C\ D\ A\ B$
Seq3	$A\ B$
Seq4	$C\ E\ F\ A\ B\ C\ A$

database $\mathcal{D}$ is a set of sequence. That is $\mathcal{D}$ = $\{Seq_1, Seq_2, Seq_3, ..., Seq_n\}$ in which the Seq_i is the i-th sequence of database. The weight of item i is in the domain of $(0, 1)$. Note that if the original weight data is not in this range, the original weight data set can be normalized.

*Definition 2 (**Contain**):* Given two sequences Seq_1 = $\{m_1, m_2, m_3, ..., m_n\}$ and Seq_2 = $\{n_1, n_2, n_3, ..., n_m\}$, Seq_2 is contained in Seq_1, which is equal to Seq_2 is a subsequence of Seq_1, denote as $S_2 \sqsubseteq S_1$ (if $S_1 \neq S_2$ denote as $S_2 \sqsubset S_1$), if and only if $\forall$ a strictly increasing sequence of integers $\{K_i\}$ ($i \in N^*$ whose upper bound is n), such that $n_1 = m_{K_1}$, $n_2 = m_{K_2}$, ..., $n_n = m_{K_n}$. It is also known Seq_1 is a contiguous subset of Seq_2, and Seq_1 is a contiguous super-set of Seq_2.

For example, given S_1 = $\{C\ D\ E\ F\ G\}$ and S_2 = $\{D\ E\ F\}$, then $S_2 \sqsubset S_1$. Given a database $\mathcal{D}$ and a sequence Seq_1, the support of Seq_1 is the number of sequences in $\mathcal{D}$, i.e., $|\{\langle Seq_j\ | Seq_j \in \mathcal{D} \wedge Seq_1 \sqsubseteq Seq_j \rangle\}|$. The following are the definitions of upper bound model that refer to the IUA algorithm [10].

Definition 3: The weight value of a subsequence S_i is W_{S_i} which is the sum of all the weights of all the items in S_i over the number of items in S_i. That is $W_{S_j} = \frac{\sum_{i \in S_j}^{|S_j|} W_i}{|S_j|}$.

For example, in Table I, the weight value of the items B and C is 0.75 and 0.12, respectively. Thus, the weight value of subsequence BC is w_{BC} = (0.75 + 0.12) ÷ 2 = 0.435. The sequential maximal weight of a sequence Seq is smw_{Seq} which is the maximal weight of an item in the sequence Seq. For example, the maximal weight value of $Seq1$ of Table I is the item D which is 0.88. Thus, smw_{Seq_1} = 0.88.

*Definition 4 (**Maximal weight**):* The total sequential maximal weight of a sequential database $\mathcal{D}$ is *tsmw* which is the addition of the *smw* of all sequences in the database $\mathcal{D}$. That is *tsmw* = $\sum_{Seq_j \subseteq \mathcal{D}} smw_{Seq_j}$.

For example, the *smw* of four transactions in Table I are 0.88, 0.88, 0.75, and 0.75, respectively. Then *tsmw* = 0.88 + 0.88 + 0.75 + 0.75 = 3.26.

Definition 5: The weighted supported value of a subsequence S_i is $wsup_{S_i}$, which is W_{S_i} multiply the number of the sequences that contain S_i, then over the *tsmw* of $\mathcal{D}$. That is $wsup_{S_i} = \frac{\sum_{S_i \subseteq Seq_j \wedge Seq_j \subseteq DB} W_{S_i}}{tsmw}$

For example, the w_{BC} = (0.75 + 0.12) ÷ 2 = 0.435 in Table I and this subsequence is contained in Seq_1, Seq_2, Seq_4. Thus, by this definition, the weighted support of subsequence BC is $wsup_{BC}$ = (0.432 + 0.432 + 0.432) ÷ 3.26 = 39.75%.

Definition 6: The sequential weighted upper-bound of a subsequence S_i is $swub_{s_i}$, which is the summation of smw_{Seq} in $\mathcal{D}$ over the *tsmw* of the sequential database. That is $swub_{S_i}$ = $\frac{\sum_{S_i \subseteq Seq_i \wedge Seq_j \subseteq \mathcal{D}} smw_{Seq_j}}{tsmw}$

For example, subsequence BC is contained in Seq_1, Seq_2, Seq_4 in Table I and the sequential maximum weight of these three sequences are 0.88, 0.88, and 0.75, respectively. Then $swub_{BC}$ = (0.88 + 0.88 + 0.75) ÷ 3.26 = 76.80%.

*Definition 7 (**Closure**):* Given a weighted contiguous sequential pattern P_1 and a sequential database $\mathcal{D}$, P_1 is a closed weighted contiguous sequential pattern if and only if there does not exist a weighted contiguous sequential pattern P_i such that $P_i \sqsubset P_1$ and $wsup_{P_1}$ = $wsup_{P_i}$.

Definition 8: Set δ is the user-defined minimal weighted supported threshold. A contiguous subsequence S_i is called an objective weighted contiguous sequential pattern (*SP* for short) if $wsup_{s_i} \geq \delta$. A contiguous subsequence S_i is called an objective weighted contiguous upper-bound pattern (*UBSP* for short) if $swub_{s_i} \geq \delta$. Given an original database $\mathcal{D}$, the goal of this paper is to discover a set of contiguous sequential patterns whose total weight is no less than the threshold value according to the existing original database and the weight of a single item.

IV. The Proposed Algorithm

This section introduces the proposed weighted contiguous sequential pattern mining algorithm. It consists of three parts: (1) the upper-bound model, (2) candidate generation, and (3) the pruning strategies.

A. Upper-bound model

In the traditional upper-bound model [23], the maximal weight equals the maximum value of all the sequences in a sequential database. In the case of Table I, this value is 0.88. Then 0.88 is used as the global upper-bound value of all subsequences in each sequence of the database. For example, item (subsequence) C is contained in Seq_1, Seq_2, and Seq_4, thus its upper-bound weighted value is (0.88 + 0.88 + 0.88), which is 2.64. In the new model, the maximal weighted value in a sequence plays the role of the upper-bound model, which is the summation of the smw of the three sequences, and this upper-bound is equal to (0.88 + 0.88 + 0.75). Thus, the upper bound of the item C is 2.51 which is tighter than the traditional upper bound. This upper-bound model (*smw*) has been proved for its completeness in IUA [10], that is, no WSPs are skipped in any discrete WSPM situation. Here, we prove this upper-bound model can also keep completeness when mining contiguous weighted sequential patterns.

First, Theorem 1 proves that the upper-bound model can keep the down-closure property in the case of contiguous weighted SPM.

Theorem 1: The sequential weighted upper-bound model smw of a contiguous pattern p can guarantee the downward-closure property.

Proof: Assume $x \subseteq$ *UBSP*, y be a super-sequential pattern of x. The frequency of x shown in the database is greater or equal to the frequency of y. For a super-sequential pattern of x, y can not exist when x is absent. Then, by the definition of *swub*, $swub_x$ is the maximal upper-bound of the weighted

value of y. Thus, if $swub_x$ is smaller than the user-defined threshold δ, then both x and y are not belong to *UBSP*. ■

Next, Theorem 2 proves the contained relationship between *UBSP* and *SP*.

Theorem 2: SP is a subset of *UBSP*, i.e., $SP \subseteq UBSP$.

Proof: By the definition 3 and definition 4, W_{S_i} must be less or equal to smw_{Seq_j}. Hence, using the definition of *SP* and *UBSP*, for a weighted sequential pattern a, $SP_a \leq UBSP_a$. On the contrary, if $a \subseteq SP_a$, then $a \subseteq UBSP_a$. ■

Using the above theorems, Theorem 3 can be drawn.

Theorem 3: Taking smw in a sequence as an upper bound can still avoid information lost for weighted contiguous data.

Proof: It is directly come from Theorem 2 that every contiguous weighted sequential pattern *SP* is also belonged to contiguous weighted frequent upper-bound pattern *UBSP*. ■

B. Generate candidate

In contrast with the splitting method, we use the PrefixSpan-like method to generate candidates. PrefixSpan [27] uses a prefix pattern growth method and projected database to find the potential candidates. It has the advantage of avoiding generating a large set of invalid candidates and shrink the size of the database during pattern growth. These characteristics improve the efficiency of our algorithm. It was used in contiguous sequential pattern mining like CCPM [7] and achieved high performance. Hence, another version of our WCSpan algorithm will use a modified PrefixSpan-like method to satisfy the features of weighted contiguous sequential patterns.

The core of our PrefixSpan-like method is the usage of prefix-trees. The prefix-tree is designed for depth-first candidate mining. Assume we take Table I as an example. The root of the prefix-tree is null, and the value of the nodes that form the first level of the prefix-tree is the 1-length patterns in *UBSP*. And the child of a node, for example, pattern A, is a projected database of item A in the original database temporally. It is shown as the right part of the schematic diagram 1. When extending the child of pattern A, each adjacent item that follows item A in the projected database of item A will be formed as a new prefix pattern. For example, the new prefix candidate pattern here is *AB*. Then the *wsup* and *swub* of pattern *AB* will be calculated. If its *swub* is bigger than threshold δ, this pattern will be added as a child of pattern A, and the child of pattern *AB* is the projected database of pattern *AB* which is formed from the projected database of pattern A. In addition, if the *wsub* of *AB* is bigger than threshold δ, pattern *AB* will be added into the set *SP*. This process is shown in Fig. 1. In this depth-first method, all viable candidates will be found. Besides, due to the characteristics of the prefix-extend method, the prefix support and sub-pattern support will be naturally checked during the pattern extension process.

In order to improve the efficiency of searching which sequence contains the pattern and locating the positions where the pattern is, two HashMaps are used. The key of the first HashMap is the ID of the sequence, and the value is the sequence itself. The key of the second HashMap is the same; however, the value is the index of the first item of the pattern

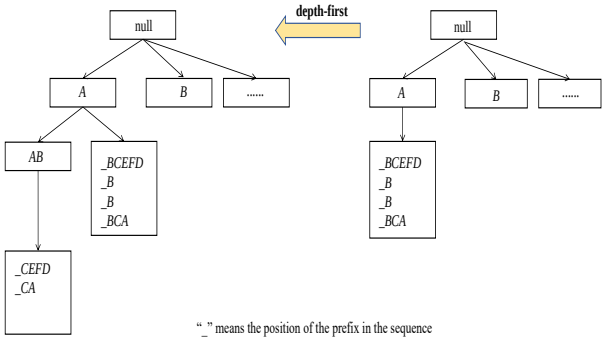

Fig. 1: Depth-first mining process of the WCSpan algorithm

that is located in the sequence. Using these two HashMaps to generate adjacent post-super candidates can be quickly done, since the *UBSP* are downward closed by Theorem 1.

C. Pruning strategies

A large number of invalid candidate sets are generated when candidate sets are generated by the adjacent candidate generation method. In this chapter, the invalid candidate sets are reduced through three kinds of checking: repeated checking, upper-bound checking, and threshold checking, to reduce the operation of the proposed WCSPan algorithm.

Repeated check. Before a candidate is put into the calculation, it will be checked to see if there is any repetition of existing patterns. This simple operation ensures that potential duplicate patterns are clipped before they are calculated.

Backward closure check. To prune the pattern that is not closed, the backward closure check is proposed to be used. Because all the candidate patterns are generated by a post-subsequence of prefix, as described in Section 4.2. By the definition of closure in Definition 8, we only need to compare the *wsup* of the new pattern and *wsup* of its prefix. If these two *wsup* are not equal and the *wsup* of the new pattern is greater than the predefined threshold, then the new pattern will be included in the SP set. This simple check can prune all the unclosed patterns without affecting the efficiency of the WCSPan algorithm.

Threshold check. Theorem 2 has proved that if a pattern does not belong to *UBSP*, then it does not belong to *SP* neither. Thus, if the *swub* of a pattern is less than the user-defined threshold δ, it must not be a weighted sequential pattern *SP*. Since *SP* is contained in *UBSP*, in order to avoid missing possible *SP*, it is necessary to recursively mine *SP* in the set of *UBSP* which leads to the proposed WCSPan algorithm.

D. Proposed algorithm

The final version of our WCSPan algorithm uses prefix-trees to discover all the contiguous sequential patterns. The details of our algorithm and the pseudocode are stated in the following.

In prefix-tree mining (cf. Algorithm 1), the original database $\mathcal{D}$, the weight value of each item, and the pre-defined threshold

Algorithm 1: Prefix-tree mining

Input: the original sequential database $\mathcal{D}$, the weight value of each item; δ, the pre-defined threshold.
Output: $\mathcal{SP}$: the set of all contiguous weighted sequential patterns.

1 **for** each Seq_y in $\mathcal{D}$ **do**
2 calculate the *smw* of each sequence in $\mathcal{D}$: $smw_y = max\{w_{y_1}, w_{y_2}, ..., w_{y_i}\}$;
3 **end**
4 calculate the *tsmw* of the $\mathcal{D}$: $tsmw = \sum_{Seq_y \subseteq \mathcal{D}} smw_y$;
5 root of *Prefix-tree* $\leftarrow$ *null*;
6 *SP* $\leftarrow$ *null*;
7 *UBSP* $\leftarrow$ *null*, and *UBSP* here is global;
8 **for** each unique length 1 candidate c_i in $\mathcal{D}$ **do**
9 calculate $swub_{c_i}$ and $wsup_{c_i}$;
10 **if** $swub_{c_i} \geq \delta$ **then**
11 put c_i into $UBSP_1$;
12 call *Prefix-extend*(root of *Prefix-tree*, c_i, $\mathcal{D}$, δ) to extend the prefix-tree;
13 **end**
14 **if** $wsup_{c_i} \geq \delta$ **then**
15 put c_i into SP_1;
16 **end**
17 **end**
18 **return** SP

Algorithm 2: Prefix-extend

Input: c, a node of the Prefix-tree; c_i, a node which will be added to the prefix-tree; $\mathcal{D}_c$ the projected database of c; δ, the pre-defined threshold.
Output: c_i: a new node of the Prefix-tree. If the new project database is empty, it returns c.

1 c.child $\leftarrow$ c_i; $\mathcal{D}_{c_i} \leftarrow$ inherit $(\mathcal{D}_c)$;
2 c_i.child $\leftarrow$ $\mathcal{D}_{c_i}$;
3 **if** $\mathcal{D}_{c_i}$ is empty **then**
4 return c;
5 **end**
6 **for** each adjacent item that following c_i in $\mathcal{D}_{c_i}$ **do**
7 new pattern $p \leftarrow$ prefix c_i + the adjacent item;
8 **if** p is unique with other children of c_i //repeat check **then**
9 calculate $swub_p$ and $wsup_p$;
10 **if** $swub_P \geq \delta$ **then**
11 put P into $UBSP_{size(p)}$;
12 call *Prefix-extend*(c_i, p, $\mathcal{D}_{c_i}$, δ);
13 **end**
14 **if** $wsup_p \geq \delta$ && $wsup_p \neq wsup_{c_i}$ **then**
15 put c_i into $SP_{size(p)}$;
16 **end**
17 **end**
18 **end**
19 **return** c_i

δ will be given. It will return all the weighted contiguous sequential patterns. Firstly, the *smw* of each sequence in the database $\mathcal{D}$ and the *tsmw* will be calculated (Lines 1 to 4). Then, based on the prefix-tree, two sets of SP and *UBSP* will be initialized (Lines 5 to 7). Next, for each unique item in $\mathcal{D}$, the *swub* and the *wsup* of it will be calculated. If its *swub* is bigger than threshold δ, it will be added into the set of length 1 *UBSP* and a depth-first prefix mining of it will be started. Besides, if its *swub* is bigger than threshold δ, it will be added into the set of length 1 *SP*.

In Prefix-extend function (cf. Algorithm 2), given the projected database of c, the threshold δ, and a node c of the prefix-tree. A pattern c_i which will be added as a child of c, and this function will return a node (c or c_i) of prefix-tree. Firstly, c_i will be added into the prefix-tree as a child of c and a new projected database $\mathcal{D}_{c_i}$ which is based on the projected database of c will be added as a child of c_i (Lines 1-2). If the $\mathcal{D}_{c_i}$ is empty, the node c will be returned. Then, for each adjacent item that follows c_i, a new pattern p will be formed by joining the adjacent item and prefix c_i (Line 6). After the repeat check (Line 8), the *swub* and *wsup* of the new pattern will be calculated (Line 10). If its *swub* is bigger than threshold δ, it will be added into the set of $UBSP_{size(p)}$ and a depth-first prefix mining will be started from the new pattern (node). In addition, if its *swub* is bigger than threshold δ, and it passes the backward closure test (Line 15), it will be added into the set of $SP_{size(p)}$.

V. Performance Evaluation

In this section, extensive experiments with two real-world datasets are used to evaluate the effectiveness and efficiency of our proposed WCSpan algorithm. We chose the state-of-the-art contiguous sequential pattern mining algorithm as a comparison. Besides, the tested state of the art of contiguous sequential patterns mining algorithm is the BP-CCSM algorithm [17] since its PrefixSpan-like method is significantly faster than the splitting method, e.g., the CCSpan algorithm [6] and its successor [14], [13]. Hence, the BP-CCSM algorithm is modified (denoted as BP-CCSM*) to be used in effectiveness analysis. In the following experiments, we first evaluated the mining outcomes of WCSpan and BP-CCSM* with equal-weighted datasets. Then, the running time and maximum memory usage of different versions of WCSpan were compared and evaluated, respectively.

A. Experimental setup and datasets

All the algorithms in the experiments are written in Java and executed on a PC with AMD Ryzen 7 4800H and 16 GB of memory, running on the 64 bit Microsoft Windows 10 OS. The test data consists of two real-world datasets of different kinds. They have diverse characteristics which can represent the main types of data in real-life situations, while the weight of each item is randomly generated, ranging from 0 to 1. The characteristics of these two datasets are described as follows.

- **Rosalind**$_{1836}$[1] is a chromosome with different fragments. This DNA dataset has 50 transactions with the 970

[1] https://github.com/jonhewz/DNASequences/blob/master/full_data_set.txt

TABLE II: Tested datasets

SID	Sequence
$Frag_{56}$	$ATTAGACCTG$
$Frag_{57}$	$CCTGCCGGAA$
$Frag_{58}$	$AGACCTGCCG$
$Frag_{59}$	$GCCGGAATAC$

average lengths of items. The items are four types of nitrogen bases. The size of this dataset is 52 KB.

- **uniflna**$_{test}$[2] is the genetic fragment of a coronavirus. It has 50 transactions. The average length of transactions is 1616 items. The items are 11 types of items. The size of this dataset is 80 KB.

To make a valid comparison, the support counting standard of the BP-CCSM* algorithm was modified to be the same as WCSpan, which is *wsup*. And the weighted value of the original data was set to an equal number. Moreover, the proposed WCSpan algorithm with different strategies is denoted as WCSpan$_{s,UB}$ (using the splitting method to generate candidates instead of using the prefix-extend method and using the upper-bound model), WCSpan$_{p,nUB}$ (using the prefix-extend method but does not use the upper-bound model), WCSpan$_{p,UB}$ (using the prefix-extend method and upper-bound model). The efficiency of these three versions of the WCSpan algorithm, together with modified BP-CCSM*, was evaluated in aspects of running time and maximum memory usage.

B. Effectiveness analysis

A case study with a short virus DNA dataset was performed this analysis. The test data and experimental results are shown in Table II, Table III, and Table IV. Table III showed the top-10 derived patterns from the running example with the threshold set as zero, and Table IV showed the top-10 derived results from the same example with a threshold of 0.5. It can be seen that the derived top-10 patterns and their corresponding values are different. The reason is that the occurrences of BP-CCSM* are the occurrences of a pattern in each dataset. Compared to WCSpan, this counting method will cause redundancy. However, due to the process of building the reversed tree, this problem would hardly be solved.

Despite the above effects, the effectiveness of the proposed WCSpan algorithm can still be checked. The top pattern *CC* of *WCSpan* in Table III are the most frequent sub-patterns in the top-10 patterns of BP-CCSM* while pattern AC is considered as a not closed pattern in WCSpan. And other top-10 patterns of WCSpan can also be found on the patterns of BP-CCSM* whose *wsup* value is 2. In addition, this result will not alter with the change in the value of the threshold shown in Table IV. Although the order of patterns with the same *swup* value will change, it will not have any effect. This kind of overlap (contains) can show the effectiveness of the proposed WCSpan.

Hence, from the above analysis of the results of found patterns, it can be concluded that although the derived patterns

[2]https://github.com/lukas-gust/data_mining_influenza/blob/master/uniflna_test1.csv

TABLE III: Derived patterns with threshold zero

WCSPan	$wsup_{WCSpan}$	BP-CCSM*	$wsup_{BP\text{-}CCSM*}$
CC	1.67	$AGACC$	2.00
CCT	1.00	$AGACCTG$	2.00
GA	1.00	AT	2.00
T	1.00	$ACCT$	2.00
$CCTG$	1.00	$AGAC$	2.00
G	1.00	ACC	2.00
C	1.00	AC	2.00
A	1.00	$ACCTG$	2.00
$GACCTG$	0.67	AGA	2.00
GAC	0.67	$AGACCT$	2.00

TABLE IV: Derived patterns with threshold 0.5f

WCSPan	$wsup_{WCSpan}$	BP-CCSM*	$wsup_{BP\text{-}CCSM*}$
CC	1.67	AC	2.00
GA	1.00	AGA	2.00
CCT	1.00	$ACCTG$	2.00
T	1.00	AG	2.00
$CCTG$	1.00	$AGACC$	2.00
G	1.00	$AGACCTG$	2.00
C	1.00	AT	2.00
A	1.00	$ACCT$	2.00
$GACCT$	0.67	$AGAC$	2.00
$GACC$	0.67	$AGACCT$	2.00

of WCSpan and BP-CCSM* are different due to the support counting, the derived patterns of WCSpan are more accurate and less redundant than those of BP-CCSM*. WCSpan can mine weighted contiguous patterns, and the contiguous patterns are just a particular case.

C. Efficiency analysis

To evaluate the efficiency of the proposed WCSpan algorithm, the performances of different versions of WCSpan (WCSpan$_{s,UB}$, WCSpan$_{p,nUB}$, WCSpan$_{p,UB}$) and BP-CCSM* (in equal weight case) are presented in terms of execution time and maximum memory usage. The following experiments of Fig. 2 and Fig. 3 are the experimental results using Rosalind$_{1836}$ to test execution time and maximum memory usage in the case of equal weight, respectively. Note that for all the figures, the x label is the value of the pre-defined threshold δ. Fig. 4 is the experiments in case of weighted item using the two datasets to test the execution time of WCSpan$_{s,UB}$, WCSpan$_{p,nUB}$ and WCSpan$_{p,UB}$. In Fig. 2, the upper subfigure shows the whole results of all compared algorithms, and the lower subfigure shows the clear partial results, that are the results of WCSpan$_{p,UB}$ and WCSpan$_{p,nUB}$. For each database in Fig. 4, the left subfigure shows the whole results of all compared algorithms, and the right subfigure shows the clear partial results, that are the results of WCSpan$_{p,UB}$ and WCSpan$_{p,nUB}$. Fig. 5 is the experiment using the two weighted datasets to test the maximum memory usage of WCSpan$_{s,UB}$, WCSpan$_{p,nUB}$ and WCSpan$_{p,UB}$, respectively. Note that due to the overflow of Java heap space, the dataset does not reach the level of MB, but it does not affect the proportion of different gaps in execution time and memory usage between these algorithms when running a large dataset.

Based on Fig. 2 to Fig. 5, we have the following observations. (1) In the experiment with an equal weight situation

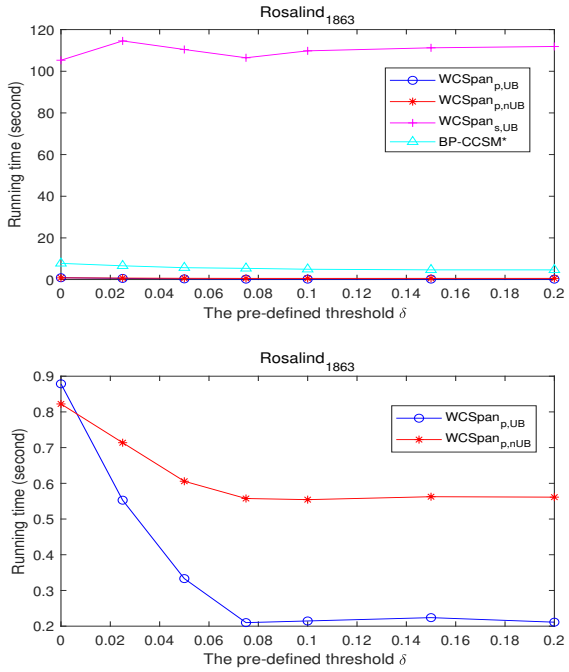

Fig. 2: Execution time in equal weighted case

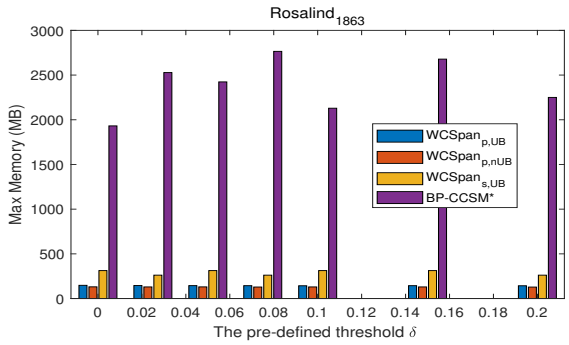

Fig. 3: Memory usage in equal weighted case

shown in Fig. 2, the execution speed of WCSpan$_{p,nUB}$ and WCSpan$_{p,UB}$ are nearly three orders of magnitude faster than WCSpan$_{s,UB}$ and one order of magnitude faster than BP-CCSM*. This means that the prefix-extend patterns mining structure is far more efficient than the splitting candidate generation method and the FP-tree-like structure in the case of (equal-weighted) contiguous sequential patterns mining. (2) In the experiments with random weight situation shown in Fig. 4, the execution speed of WCSpan$_{p,UB}$ is obviously faster than that of WCSpan$_{p,nUB}$, and the efficiency gap between them will become wider with the increase of threshold value. The reason is that the upper-bound mode can prune the invalid patterns that do not belong to the contiguous weighted upper-bound pattern *UBSP*. As the threshold goes up, the model becomes more powerful at pruning invalid patterns. Eventually, the running time of both WCSpan$_{p,UB}$ and WCSpan$_{p,nUB}$ will stabilize due to the stabilization of derived patterns. (3) In the random weighted case, algorithms in Rosalind$_{1836}$ and uniflna$_{test}$ almost have the same performances. Compared WCSpan$_{p,UB}$ to WCSpan$_{s,UB}$ in experiments with random weight (Fig. 4), WCSpan$_{p,UB}$ is far more efficiency than WCSpan$_{s,UB}$. It can be explained by the inefficiency of the modified splitting method in candidate generation. It can also explain why the running time does not have a big fluctuation with the growth of the threshold. The reason is that the time used in the calculation is far less than that of splitting. (4) In the maximum memory experiments with equal weight (Fig. 3), the memory used for BP-CCSM* is obviously larger than that of WCSpan$_{p,UB}$, WCSpan$_{p,nUB}$ and WCSpan$_{s,nUB}$. This is due to the usage of depth-first order in WCSpan. It can discover the objective patterns, while BP-CCSM* will store all the information into a tree and keep constructing such a tree to discover all the objective patterns at a time. In the max memory experiments with random weigh (Fig. 5), the maximum memory of WCSpan$_{s,UB}$ is double or triple than that of WCSpan$_{p,UB}$ and WCSpan$_{p,UB}$. This result shows that the projected dataset and prefix-extend patterns mining take advantage of memory usage compared to the splitting candidate generation method. Besides, the maximum memory of WCSpan$_{p,UB}$ is slightly more than that of WCSpan$_{p,nUB}$. It is because the calculation of *swub* takes some memory during the mining process.

VI. Conclusion

In this paper, we propose the WCSpan algorithm that can discover weighted contiguous sequential patterns. WCSpan uses a PrefixSpan-like method to generate potential candidates. In addition, with the use of a modified upper-bound model that utilizes the maximal weighted item of each transaction, WCSpan can prune the invalid candidate efficiently. Several experiments show that in the equal-weighted case, compared with the existing state-of-the-art contiguous pattern mining algorithm, the result of WCSpan is more accurate and less tedious. The WCSpan algorithm also has advantages in execution speed and memory usage. Finally, the upper-bound model and prefix-extend structure of WCSPan have greatly improved the efficiency of execution speed and memory usage in the general case.

VII. Acknowledgment

This research was supported in part by the National Natural Science Foundation of China (Grant No. 62002136), Natural Science Foundation of Guangdong Province of China (Grant No. 2022A1515011861), Guangzhou Basic and Applied Basic Research Foundation (Grant No. 202102020277).

References

[1] W. Gan, J. C. W. Lin, H. C. Chao, and J. Zhan, "Data mining in distributed environment: a survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 7, no. 6, p. e1216, 2017.

[2] W. Gan, J. C.-W. Lin, P. Fournier-Viger, H. C. Chao, V. S. Tseng, and P. S. Yu, "A survey of utility-oriented pattern mining," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 4, pp. 1306–1327, 2021.

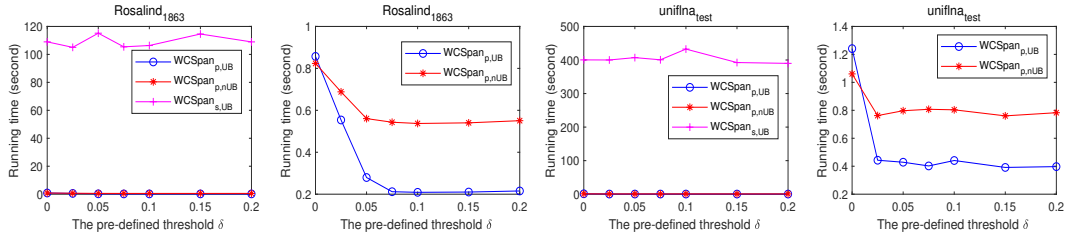

Fig. 4: Execution time in random weighted case

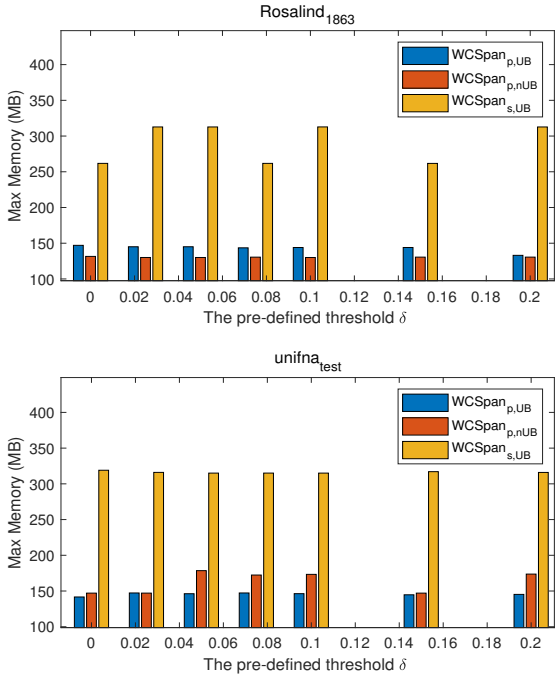

Fig. 5: Memory usage in random weighted case

[3] W. Gan, J. C. W. Lin, P. Fournier-Viger, H. C. Chao, and P. S. Yu, "A survey of parallel sequential pattern mining," *ACM Transactions on Knowledge Discovery from Data*, vol. 13, no. 3, pp. 1–34, 2019.

[4] P. Fournier-Viger, J. C. W. Lin, R. U. Kiran, Y. S. Koh, and R. Thomas, "A survey of sequential pattern mining," *Data Science and Pattern Recognition*, vol. 1, no. 1, pp. 54–77, 2017.

[5] R. Agrawal, R. Srikant *et al.*, "Fast algorithms for mining association rules," in *Proceedings of the 20th International Conference on Very Large Data Bases*, vol. 1215, 1994, pp. 487–499.

[6] J. Zhang, Y. Wang, and D. Yang, "CCSpan: Mining closed contiguous sequential patterns," *Knowledge-Based Systems*, vol. 89, pp. 1–13, 2015.

[7] Y. Abboud, A. Boyer, and A. Brun, "CCPM: a scalable and noise-resistant closed contiguous sequential patterns mining algorithm," in *International Conference on Machine Learning and Data Mining in Pattern Recognition*. Springer, 2017, pp. 147–162.

[8] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining frequent patterns without candidate generation: A frequent-pattern tree approach," *Data Mining and Knowledge Discovery*, vol. 8, no. 1, pp. 53–87, 2004.

[9] Y. H. Goo, K. S. Shim, M. S. Lee, and M. S. Kim, "Protocol specification extraction based on contiguous sequential pattern algorithm," *IEEE Access*, vol. 7, pp. 36 057–36 074, 2019.

[10] G. C. Lan, T. P. Hong, and H. Y. Lee, "An efficient approach for finding weighted sequential patterns from sequence databases," *Applied Intelligence*, vol. 41, no. 2, pp. 439–452, 2014.

[11] C. Zhang, Z. Du, W. Gan, and P. S. Yu, "TKUS: Mining top-k high utility sequential patterns," *Information Sciences*, vol. 570, pp. 342–359, 2021.

[12] J. Chen and T. Cook, "Mining contiguous sequential patterns from web logs," pp. 1177–1178, 2007.

[13] L. Bermingham and I. Lee, "Mining distinct and contiguous sequential patterns from large vehicle trajectories," *Knowledge-Based Systems*, vol. 189, p. 105076, 2020.

[14] V. E. Adeyemo, A. Palczewska, and B. Jones, "LCCspm: l-length closed contiguous sequential patterns mining algorithm to find frequent athlete movement patterns from gps," pp. 455–460, 2021.

[15] R. Srikant and R. Agrawal, "Mining quantitative association rules in large relational tables," in *ACM SIGMOD Record*, vol. 25, no. 2. ACM, 1996, pp. 1–12.

[16] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," in *ACM SIGMOD Record*, vol. 22, no. 2. ACM, 1993, pp. 207–216.

[17] C. Yang and G. Gidófalvi, "Mining and visual exploration of closed contiguous sequential patterns in trajectories," *International Journal of Geographical Information Science*, vol. 32, no. 7, pp. 1282–1304, 2018.

[18] Y. Abboud, A. Brun, and A. Boyer, "C3Ro: an efficient mining algorithm of extended-closed contiguous robust sequential patterns in noisy data," *Expert Systems with Applications*, vol. 131, pp. 172–189, 2019.

[19] W. Gan, J. C. W. Lin, P. Fournier-Viger, H. C. Chao, J. Zhan, and J. Zhang, "Exploiting highly qualified pattern with frequency and weight occupancy," *Knowledge and Information Systems*, vol. 56, no. 1, pp. 165–196, 2018.

[20] W. Gan, J. C. W. Lin, J. Zhang, P. Fournier-Viger, H. C. Chao, and P. S. Yu, "Fast utility mining on sequence data," *IEEE Transactions on Cybernetics*, vol. 51, no. 2, pp. 487–500, 2021.

[21] G. Ramkumar, S. Ranka, and S. Tsur, "Weighted association rules: Model and algorithm," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Citeseer, 1998, pp. 661–666.

[22] F. Tao, F. Murtagh, and M. Farid, "Weighted association rule mining using weighted support and significance framework," in *The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003, pp. 661–666.

[23] U. Yun and J. J. Leggett, "WFIM: weighted frequent itemset mining with a weight range and a minimum weight," in *The SIAM International Conference on Data Mining*. SIAM, 2005, pp. 636–640.

[24] S. Z. Ishita, F. Noor, and C. F. Ahmed, "An efficient approach for mining weighted sequential patterns in dynamic databases," in *Industrial Conference on Data Mining*. Springer, 2018, pp. 215–229.

[25] U. Yun, G. Pyun, and E. Yoon, "Efficient mining of robust closed weighted sequential patterns without information loss," *International Journal on Artificial Intelligence Tools*, vol. 24, no. 01, p. 1550007, 2015.

[26] K. K. Roy, M. H. H. Moon, M. M. Rahman, C. F. Ahmed, and C. K.-S. Leung, "Mining weighted sequential patterns in incremental uncertain databases," *Information Sciences*, vol. 582, pp. 865–896, 2022.

[27] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M. C. Hsu, "Mining sequential patterns by pattern-growth: The PrefixSpan approach," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 11, pp. 1424–1440, 2004.