Received December 5, 2019, accepted February 22, 2020, date of publication February 27, 2020, date of current version March 10, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2976662

Efficient Chain Structure for High-Utility Sequential Pattern Mining

JERRY CHUN-WEI LIN[1], (Senior Member, IEEE), YUANFA LI[2], PHILIPPE FOURNIER-VIGER[3], YOUCEF DJENOURI[4], AND JI ZHANG[5], (Senior Member, IEEE)

[1]Department of Computer Science, Electrical Engineering and Mathematical Sciences, Western Norway University of Applied Sciences, Bergen 5063, Norway
[2]School of Computer Science and Technology, Harbin Institute of Technology (Shenzhen), Shenzhen 518055, China
[3]School of Humanities and Social Sciences, Harbin Institute of Technology (Shenzhen), Shenzhen 518055, China
[4]SINTEF Digital, 0373 Oslo, Norway
[5]Faculty of Health, Engineering and Sciences, University of Southern Queensland, Toowoomba, QLD 4350, Australia

Corresponding author: Jerry Chun-Wei Lin (jerrylin@ieee.org)

This work was supported in part by the Western Norway University of Applied Sciences.

ABSTRACT High-utility sequential pattern mining (HUSPM) is an emerging topic in data mining, which considers both utility and sequence factors to derive the set of high-utility sequential patterns (HUSPs) from the quantitative databases. Several works have been presented to reduce the computational cost by variants of pruning strategies. In this paper, we present an efficient sequence-utility (SU)-chain structure, which can be used to store more relevant information to improve mining performance. Based on the SU-Chain structure, the existing pruning strategies can also be utilized here to early prune the unpromising candidates and obtain the satisfied HUSPs. Experiments are then compared with the state-of-the-art HUSPM algorithms and the results showed that the SU-Chain-based model can efficiently improve the efficiency performance than the existing HUSPM algorithms in terms of runtime and number of the determined candidates.

INDEX TERMS High utility sequential pattern mining, sequence, SU-Chain structure, data mining.

I. INTRODUCTION

Pattern mining is considered to find the valuable relationships between items/objects in the databases, and many variants of knowledge were then investigated in different applications and domains, such as association-rule mining (ARM) [1], [10], sequential-pattern mining (SPM) [2], [9], [22], [23], and high-utility-itemset mining (HUIM) [6], [12], [13], [18], among others. SPM which discovers high frequent sequence from sequence database, is one of the important research areas in data mining and knowledge discovery since it shows the correlations of the ordered events, which can be applied in many real-life applications and situations. For example, the sequence data can be extracted from the Weblog, DNA sequence, or trajectory datasets. Several algorithms [9], [22], [23] have been proposed to improve the mining efficiency regarding SPM but most of them do not, however, consider the other factors or attributes in the databases (i.e., importance, weight or interestingness). To solve this limitation and provide more useful and meaingful information, the high-utility sequential pattern mining (HUSPM) [17], [29], [32], [33] was presented to consider both utility and sequence factors to reveal the set of high-utility sequential patterns (HUSPs) from the databases. It takes the quantities and unit profits of the items into account to mine the set of HUSPs as the required knowledge for decision making. In the HUSPM, a sequence is considered as a HUSP if its sequence utility is no less than the pre-defined minimum utility value. However, the task of HUSPM is more complex than that of traditional SPM since the sequence utility does not hold the downward closure property, thus the search space to discover the required HUSPs has become huge. Many algorithms were presented to design the pruning strategies and new upper-bound values to reduce the search space such as USpan [32], HUS-Span [29], ProUM [7] and HUSP-ULL [8]. USpan was introduced to utilize the lexicographic quantitative sequence tree for mining the HUSPs, but the upper-bound value is over-estimated, thus the search space to find the required information is too huge. HUS-Span consists of two tighter upper bounds, respectively named as prefix extension utility (PEU) and reduced sequence utility (RSU) to establish the upper-bound values of the promising candidates. With the designed pruning strategies and the new upper bounds, the HUS-Span can greatly reduce the size of the unpromising candidates to mine the HUSPs. However, the HUS-Span still has to generate many unpromising candidates to completely mine

The associate editor coordinating the review of this manuscript and approving it for publication was Changsheng Li.

the HUSPs, and the database is required to be projected at each time by the level-wise approach. Moreover, the used PEU in the HUS-Span is not updated at each iteration, thus the upper bound is still overestimated; more unpromising candidates are still required to be determined. ProUM [7] and HUSP-ULL [8] are the state-of-the-art approaches by introducing the projection mechanism and efficient pruning strategies to mine the HUSPs.

The above algorithms still suffer the limitation of memory usage (i.e., the state-of-the-art HUSP-ULL), we thus design an efficient sequence-utility (SU)-Chain structure to keep more information for the later mining progress. The projection approach is also utilized in the SU-Chain-based algorithm to speed up the generation progress of the promising candidates. Moreover, several pruning strategies are utilized in the SU-Chain structure to identify the irrelevant information for the early pruning progress, thus those items can be removed from the projected database, and the search space can be also reduced. From the experiments, we can also observe that the developed SU-Chain structure can produce better performance compared to the previous works regarding the runtime and number of examined candidates.

The organization of this paper is stated below. Literature review is discussed in Section 2. Preliminaries and problem statement of the designed model are studied in Section 3. The designed sequence-utility-chain-based model is developed in Section 4. Experiments are conducted and discussed in Section 5. Finally, the conclusion and future works are mentioned in Section 6.

II. LITERATURE REVIEW

High utility itemset mining (HUIM) [6], [12], [13], [18], [28], [31] is to consider the utility factor of the itemsets to reveal the high profitable itemsets from the databases. Compared with association rule mining (ARM) [1], [10], conventional HUIM does not hold the downward closure property thus the search space of the promising candidates is huge. To efficiently mine the high-utility itemsets (HUIs) and reduce the size of the search space, Liu *et al.* [13] proposed the transaction-weighted utility (TWU) concept, which is used to estimate the upper bound of the itemset utility value. Tseng et al. extended the FP-tree and proposed the UP-growth [25] and UP-growth+ [26] algorithms to exploit the nature of the tree for compressing the search space. Lin *et al.* [14] also presented the HUP-tree, which is based on the TWU concept and FP-tree structure [10]. The HUP-tree uses the tree structure to keep the necessary information of the promising candidates, and the HUP-growth mining algorithm was presented to discover the required HUIs based on the HUP-tree structure. Liu and Qu [15] proposed the HUI-Miner algorithm, which converts the original database into an utility-list structure and mines the HUIs efficiently from the utility-list to avoid the generation progress of the huge candidates. Zida *et al.* [35] designed a novel algorithm called EFIM, which consists of two upper bounds on utility to reduce the size of the search space. Several algorithms [11], [20], [30] using the evolutionary computations have been discussed to find the HUIs in a limit time. Research directions include the improvement of high-utility itemset mining [19], high-utility itemset mining for IoT uncertain data [18], and mining top-*k* high-utility itemsets [27] are also the interesting issues and been developed in progress.

High-utility sequential-pattern mining (HUSPM) is an emerging field in recent decades since it considers both utility and sequence factors to discover the utility utility sequential patterns (HUSPs) from the sequence dataset. HUSPM can also be considered for sequence mining of Website logs [34]. Shie *et al.* [24] proposed the UMSP algorithm and the UM-span algorithm for mining high-utility mobile sequences based on the mobile-business applications. To exploit the usefulness of web page access sequence data, Ahmed *et al.* [3] proposed two tree structures, respectively called UWAS-tree and IUWAS-tree, to process the static and dynamic databases. Subsequently, Ahmed *et al.* [4] proposed a high-utility sequential pattern mining algorithm for processing general sequences, namely, the layer-by-layer search UL algorithm and the pattern-extended US algorithm. However, there is no formal definition of high-utility sequential pattern mining. Yin *et al.* [32] officially defined high-utility sequential pattern mining and proposed an efficient algorithm, USpan, for mining general sequence patterns with utility values. To simplify the parameter setting, Yin *et al.* [33] then proposed the TUS algorithm for discovering the top-*k* high-utility sequential patterns. Lan *et al.* [16], [17] first introduced the concept of fuzziness into sequence mining and then proposed a high-utility sequential pattern mining algorithm to simplify the mining results and reduce the search space. Alkan and Karagoz [5] proposed a high-utility sequential pattern extraction (HuspExt) algorithm, which is used to calculate the Cumulated Rest of Match (CRoM) to obtain a smaller upper bound. The complexity of the search space can thus be reduced. Wang *et al.* [29] subsequently proposed the HUS-Span algorithm to reduce the unpromising candidates by introducing two utility upper bounds called PEUs and RSUs. However, it is still challenging to find the HUSPs from a very big dataset. In addition, Gan et al. then presented a projection method called ProUM [7] and the HUSP-ULL [8] to efficiently mine the HUSPs, which are the state-of-the-art approaches based on the utility-list structure.

III. PRELIMINARIES AND PROBLEM STATEMENT

Let $I = \{i_1, i_2, \ldots, i_m\}$ be the finite set of m distinct items. A quantitative item, abbreviated as q-item, is denoted as (i_k, q_k), which is used to represent the item with its purchase quantity. An itemset is denoted as $w = [i_1, i_2, \ldots, i_h]$ is a subset of I where $h \leq m$. A quantitative itemset, abbreviated as q-itemset, is denoted as $v = [(i_1, q_1), (i_2, q_2), \ldots, (i_n, q_h))]$ is the set of several q-items. Without loss of generality, here we assume that the items (q-items) are sorted as *alphabetic* order in itemset (q-itemset) through the items in itemset. A sequence is denoted as $t = < w_1, w_2, \ldots, w_d >$ is the

TABLE 1. A quantitative sequence database.

sid	sequence
s_1	<[(*c*:6)(*a*:3)],[(*e*:2)(*c*:2)],[(*e*:6)(*b*:3)],[(*c*:2)]>
s_2	<[(*e*:4)(*b*:2)],[(*e*:5)(*a*:2)(*d*:3)],[(*e*:1)(*c*:6)(*b*:4)]>
s_3	<[(*a*:1)],[(*e*:3)(*b*:7)],[(*c*:5)(*d*:4)],[(*c*:1)(*a*:5)]>
s_4	<[(*e*:2)(*b*:8)],[(*c*:8)(*a*:6)],[(*c*:8)(*d*:5)(*f*:6)]>

TABLE 2. A profit table.

item	*a*	*b*	*c*	*d*	*e*	*f*
profit	3	4	2	5	1	4

sorted list of one or more itemset. A quantitative sequence, abbreviated as *q*-sequence is denoted as $s = < v_1, v_2, \ldots, v_d >$, which is the sorted list of one or more *q*-itemset. A quantitative sequence database, abbreviated as *q*-sequence database $S = \{s_1, s_2, \ldots, s_n\}$ is the set of *q*-sequence where each *q*-sequence is associated with a unique identifier called *sid*. Table 1 shows a quantitative sequential database. It has four *q*-sequences and six items, denoted from *a* to *f*. Table 2 shows the unit profits of the items that appear in Table 1.

Here, several definitions regarding the HUSPM are given below.

Definition 1: Let the utility of an item i_r in a *q*-itemset *v* denote as $u(i_r, v)$, and is defined as:

$$u(i_r, v) = q(i_r, v) \times pr(i_r), \quad (1)$$

where $q(i_r, v)$ is the quantity in a *q*-itemset *v* and $pr(i_r)$ is the profit of an item i_r.

Example 1: Take an example as follows. The utility of an item *a* in the first *q*-itemset of s_1 in Table 1 is calculated as: $u(a,[(c{:}6)(a{:}3)]) = q(a,[(c{:}6)(a{:}3)]) \times pr(a) = 3 \times 3 = 9$

Definition 2: Let the utility of a *q*-itemset in a *q*-sequence *s* denote as $u(X, s)$, and is defined as:

$$u(X, s) = \sum_{X \in s \wedge i_r \in X} u(i_r, X) \quad (2)$$

Example 2: Take an example as follows. The utility of a *q*-itemset [(*c*:6)(*a*:3)] in *q*-sequence s_1 is calculated as: $u([(c{:}6)(a{:}3)], s_1) = u(c,[(c{:}6)(a{:}3)]) + u(a,[(c{:}6)(a{:}3)]) = 6 \times 2 + 3 \times 3 = 21$.

Definition 3: Let the utility of a *q*-sequence in a quantitative sequential database *D* denote as $u(s)$, and is defined as:

$$u(s) = \sum_{s \in D \wedge X \in s} u(X, s) \quad (3)$$

Example 3: Take an example as follows. The utility of the *q*-sequence s_1 in Table 1 is calculated as: $u(s_1) = u([(c : 6)(a : 3)], s_1) + u([(e : 2)(c : 2)], s_1) + u([(e : 6)(b : 3)], s_1) + u([(c : 2)], s_1) = 21 + 6 + 18 + 4 (= 49)$.

Definition 4: Let the utility of a quantitative sequential database *D* denote as $u(D)$, which is the sum of the utility of each its *q*-sequence and defined as:

$$u(D) = \sum_{s \in D} u(s) \quad (4)$$

Example 4: Take an example as follows. The utility of the quantitative sequential database *D* in Table 1 is calculated as: $u(D) = u(s_1) + u(s_2) + u(s_3) + u(4) (= 49 + 67 + 81 + 133) (= 330)$.

Definition 5: Given two itemsets, $w_a = [i_{a_1}, i_{a_2}, \ldots, i_{a_m}]$ and $w_b = [i_{b_1}, i_{b_2}, \ldots, i_{b_n}]$, where $i_{a_k} \in I (1 \leq k \leq m)$ and $i_{b_{k'}} \in I (1 \leq k' \leq n)$, if there exists positive integers $1 \leq j_1 \leq j_2 \leq \cdots \leq j_m \leq n$ such that $i_{a_1} = i_{b_{j_1}}, i_{a_2} = i_{b_{j_2}}, \ldots, i_{a_m} = i_{b_{j_m}}$, then w_b contains w_a, which is denoted as $w_a \subseteq w_b$.

Example 5: The itemset [*a*, *c*] contains the itemsets [*a*], [*c*] and [*a*, *c*].

Definition 6: Given a *q*-itemset $v_a = [(i_{a_1}, q_{a_1})(i_{a_2}, q_{a_2})\ldots(i_{a_m}, q_{a_m})]$ and a *q*-itemset $v_b = [(i_{b_1}, q_{b_1})(i_{b_2}, q_{b_2})\ldots(i_{b_n}, q_{b_n})]$, where $i_{a_k} \in I (1 \leq k \leq m)$ and $i_{b_{k'}} \in I (1 \leq k' \leq n)$, if there exists positive integers $1 \leq j_1 \leq j_2 \leq \cdots \leq j_m \leq n$ such that $i_{a_1} = i_{b_{j_1}} \wedge q_{a_1} = q_{b_{j_1}}, i_{a_2} = i_{b_{j_2}} \wedge q_{a_2} = q_{b_{j_2}}, \ldots, i_{a_m} = i_{b_{j_m}} \wedge q_{a_m} = q_{b_{j_m}}$, then v_b contains v_a, which is denoted as $v_a \subseteq v_b$.

Example 6: Take an example as follows. The *q*-itemset [(*c*:6)(*a*:3)] in *q*-sequence s_1 in Table 1 contains the *q*-itemset [(*c*:6)], *q*-itemset [(*a*:3)] and *q*-itemset [(*c*:6)(*a*:3)].

Definition 7: Given two sequences $t_a = < w_{a_1}, w_{a_2}, \ldots, w_{a_m} >$ and $t_b = < w_{b_1}, w_{b_2}, \ldots, w_{b_n} >$, where $w_{a_k} \subseteq I$ $(1 \leq k \leq m)$ and $w_{b_{k'}} \subseteq I$ $(1 \leq k' \leq m)$ are both itemsets, if there exists positive integers $1 \leq j_1 \leq j_2 \leq \cdots \leq j_m \leq n$ such that $w_{a_1} \subseteq w_{b_{j_1}}, w_{a_2} \subseteq w_{b_{j_2}}, \ldots, w_{a_m} \subseteq w_{b_{j_m}}$, then t_a is the subsequence of t_b, which is denoted as $t_a \subseteq t_b$.

Example 7: Take an example as follows. A sequence <[*a*,*b*],[*a*,*c*],[*b*,*c*]> is the subsequence of the sequence <[*a*,*b*],[*a*,*b*,*c*], [*a*,*b*],[*b*,*c*]>.

Definition 8: Given two *q*-sequences $s_a = < v_{a_1}, v_{a_2}, \ldots, v_{a_m} >$ and $s_b = < v_{b_1}, v_{b_2}, \ldots, v_{b_n} >$, where v_{a_k} $(1 \leq k \leq m)$ and $v_{b_{k'}}$ $(1 \leq k' \leq n)$ are both *q*-itemsets, if there exists positive integers $1 \leq j_1 \leq j_2 \leq \cdots \leq j_m \leq n$ such that $v_{a_1} \subseteq v_{b_{j_1}}, v_{a_2} \subseteq v_{b_{j_2}}, \ldots, v_{a_m} \subseteq v_{b_{j_m}}$, then s_a is the *q*-subsequence of s_b, which is denoted as $s_a \subseteq s_b$.

Example 8: Take an example as follows. The *q*-sequences <[(*c*:6)],[(*e*:2)(*c*:2)]> and <[(*b*:2)],[(*e*:6)(*b*:3)]> are two *q*-subsequences of the *q*-sequence s_1 in Table 1.

Definition 9: Given a *q*-sequence $s = < v_1, v_2, \ldots, v_n >$ and a sequence $t = < w_1, w_2, \ldots, w_m >$, if $n = m$ and the items in v_i are same as the items in w_i, where $1 \leq i \leq n$, then *s* is said to match *t*, which can be denoted as $t \sim s$.

Example 9: Take an example as follows. A sequence <[*c*][*e*,*b*]> matches the s_1 in Table 1. Notice that the two *q*-itemsets may be considered as different although they contain the same itemset because of the quantities and the position of a *q*-sequence. Therefore, it is possible that more than one *q*-subsequence of a *q*-sequence matches the given sequence. The sequence <[*c*]> has three matches in s_1:<[(*c*:6)]>, <[(*c*:2)]>, and <[(*c*:2)]>.

Definition 10: A *q*-itemset containing *k* items is called *k*-*q*-itemset. A *q*-sequence containing *k* items is called *k*-*q*-sequence.

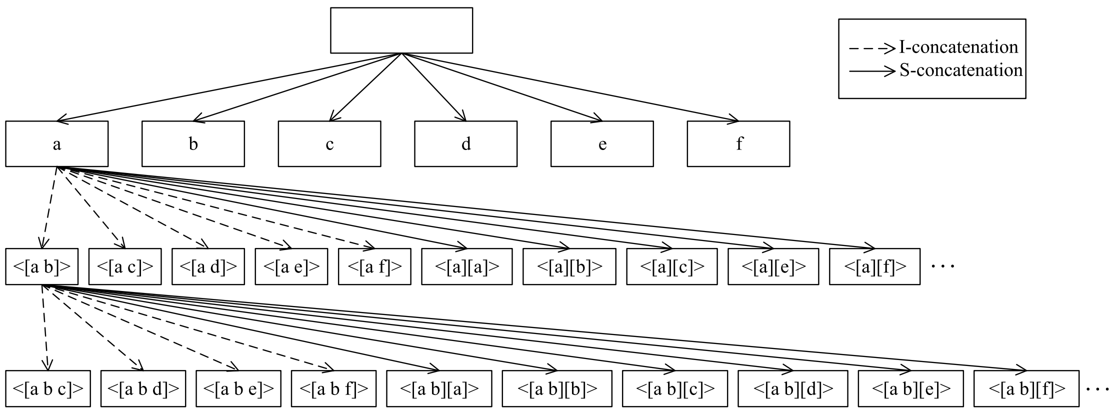

FIGURE 1. **The lexicographic enumeration (LE)-tree of the search space.**

Example 10: Take an example as follows. The q-sequence s_1 is a 7-q-sequence. The first q-itemset of q-sequence is a 2-q-itemset.

Definition 11: Let the utility of a sequence t in a q-sequence s denote as $u(t, s)$, and is defined as:

$$u(t, s) = max\{u(s_k)|t \sim s_k \wedge s_k \subseteq s\}, \qquad (5)$$

where $\sim$ denotes the matched relationship and $t \sim s_k$ represents that s_k is the match of t.

Example 11: Take an example as follows. The utility of a sequence $<[a], [b]>$ in the q-sequence s_1 of Table 1 is calculated as: $u(< [a], [c] >, s_1) = max\{u([a : 3], [c : 2], s_1), u([a : 3], [c : 2], s_1)\} = max\{13, 13\} = 24$. This example shows that a target sequence in high-utility sequential pattern mining may have multiple utility values in a transaction, which is quite different from traditional high-utility itemset mining and frequent itemset mining. Different evaluation criteria choose different utility values, and here the maximum value is used as the utility value of the target sequence.

Definition 12: Let the utility of a sequence t in a quantitative sequence database D denote as $u(t)$, and is defined as:

$$u(t) = max\{u(t, s)|t \sim s_k \wedge s_k \subseteq s\} \qquad (6)$$

Example 12: Take an example as follows. The utility of a sequence $<[a],[b]>$ in Table 1 is calculated as: $u(< [a], [c] >) = u(< [a], [c] >, s_1) + u(< [a], [c] >, s_2) + u(< [a], [c] >, s_3) + u(< [a], [c] >, s_4) = 13 + 18 + 13 + 34$ (=78).

Definition 13: A sequence t in a quantitative sequential database D is a high utility sequential pattern (HUSP) if it satisfies the condition as:

$$HUSP \leftarrow \{t|u(t) \geq \delta \times u(D)\}, \qquad (7)$$

where δ is minimum utility threshold and $u(D)$ is the total utility of the q-sequence D.

Example 13: Take an example as follows. The utility of the sequence $<[a],[b]>$ in the q-sequence database D is $u(< [a], [c] >) = 78$, and the utility of the q-sequence database D is $u(D) = 330$. If the minimum utility threshold is set to 0.2, then the sequence $<[a],[b]>$ is considered as a high-utility sequential pattern in the sequence database D since $178 \geq 0.2 \times 330$ (= 66).

Problem Statement: Given a quantitative sequence database and a user-defined minimum utility threshold, the task of high utility sequential pattern mining (HUSPM) is to find the complete set of high utility sequential patterns (HUSPs) in which the utility value of each sequence is no less than $\delta \times u(D)$ from the quantitative database.

IV. DEVELOPED SEQUENCE-UTILITY (SU)-CHAIN-BASED MODEL

In this paper, we present a novel sequence-utility (SU)-Chain structure to keep more information for further mining process. A lexicographic enumeration (LE)-tree is used here to represent the search space of the promising candidates, which can be shown as Figure 1.

In Figure 1, the *I-Concatenation* and *S-Concatenation* are used in the pattern-growth mechanism [29], [32] to generate the possible and promising HUSPs. Based on the *I-Concatenation* and *S-Concatenation* for the enumeration tree, all the possible and promising candidates can be produced and explored. In order to ensure the integrity of the mining results, we should concatenate items in a certain order [7], [8]. It is noted that the definition of sequence order is also suitable for q-sequence. According to the definition of sequence order, we could produce all candidate sequences completely without loss of integrity.

For the HUS-Span [29] and ProUM [7], it needs to generate the projection database of a sequence t using the original database. A designed sequence-utility (SU)-Chain here can be considered to produce the projection database for the sequence. While exploring the child nodes in the LE-tree, this projection database could be passed to the child nodes after updating. This progress can be used to reduce time consumption. Table 3 shows the SU-Chain of a sequence $<a>$ from Table 1. The SU-Chain is a set of projection sequences and utility-lists. The element of the utility-lists contains four fields as: **concatenation position** p_i;

TABLE 3. The built SU-Chain of the running example of the sequence <*a*>.

Projection sequence	Utility list
<[(*c*:6)],[(*c*:2)(*e*:2)],[(*b*:3)(*e*:6)],[(*c*:2)]>	[1,9,40]
<[(*d*:3)(*e*:5)],[(*b*:4)(*c*:6)(*e*:1)]>	[1,6,49]
<[],[(*b*:7)(*e*:3)],[(*c*:5)(*d*:4)],[(*a*:5)(*c*:1)]>	[1,3,78]->[4,15,2]
<[(*c*:8)],[(*c*:8)(*d*:5)(*f*:6)]>	[1,18,81]

Algorithm 1 SU-Chain Construction

```
Input: suc, SU-Chain of t; i, Concatenation Item; type,
       I-Concatenation or S-Concatenation
Output: suc', the SU-Chain of new sequence.
1 $suc' = \emptyset$ ;
2 for each proseq, ul ∈ suc do
3     find the concatenation candidate items C ;
4     if $i \in C$ then
5         build new utility list ul' ;
6         project proseq to the new projection sequence
          proseq' $suc' \leftarrow suc' \cup (ul', proseq')$
7 return suc'
```

Algorithm 2 Mining (*t*, *suc*)

```
1  if $PEU(t) \geq \delta \times u(D)$ then
2      scan projection $D_{project}$ of t to remove the irrelevant
       items;
3      Find the I-Concatenations and S-Concatenations;
4      remove the unpromising items from
       I-Concatenations and S-Concatenations;
5      for each item $i_j$ in I-Concatenation do
6          $t' \leftarrow I - Concatenation(t, i_j)$;
7          $suc' \leftarrow C$;
8          if $u(t') \geq \delta \times u(D)$ then
9              $HUSPs \leftarrow HUSPs \cup t'$;
10         Mining($t'$, $suc'$) ;
11     for each item $i_j$ in S-Concatenation do
12         $t' \leftarrow S - Concatenation(t, i_j)$;
13         build $suc'$ of $t'$;
14         if $u(t') \geq \delta \times u(D)$ then
15             $HUSPs \leftarrow HUSPs \cup t'$
16         Mining($t'$, $suc'$) ;
17     return HUSPs
```

the **maximum utility at concatenation position** p_i; the **utility of remaining sequence** $s/_{t,p_i}$; **a pointer** pointing to either the $(i+1)$-th concatenation position or *null*.

Based on the SU-Chain, the projection sequence can thus be maintained for later generation of the promising candidates for examination. Also, it is easy to find the *I-Concatenation* and *S-Concatenation* of the sequences. Thus, the computational cost can be greatly reduced to mine the required HUSPs. The designed SU-Chain structure is then presented in Algorithm 1. The main construction process is divided into three parts as: (1) find the candidate concatenation items of *I-Concatenation* or *S-Concatenation*; (2) build the new utility-list; and (3) project the required sequences.

In order to efficiently reduce the size of the search space for mining HUSPs, several pruning strategies [21], [32] can thus incorporated with the designed SU-Chain structure to improve mining performance. Several definitions, theorems and pruning strategies are then given below.

Definition 14: $SWU(t)$ is used to denote the sequence weighted utilization of *t* in the *q*-sequence database *SUD*, and defined as:

$$SWU(t) = \sum_{s \in SUD} \{u(s) | t \subseteq s\} \tag{8}$$

Theorem 1 Given a sequence *t*, for each sequence t' that could be generated by *t* using concatenation operations, we then can obtain that: $u(t') \leq SWU(t)$

Proof: As the above definition, it is obvious that $u(t') \leq SWU(t')$ holds. Since $t \subseteq t'$, $SWU(t') = \sum_{s \in D}\{u(s)|t' \subseteq s\}$ $\leq \sum_{s \in D}\{u(s)|t \subseteq s\}$.

Pruning strategy 1: According to **Theorem 1**, For a given sequence *t*, if $SWU(t)$ is less than the minimum utility value, the utility of any sequences which could be generated by *t* will be less than the minimum utility value. And these sequences could be safely pruned from the *LE-tree* without affecting the complete mining results.

Definition 15: $PEU(t, s)$ is used to denote the prefix extension utility of *t* in *q*-sequence *s*, and defined as:

$$PEU(t, s) = \begin{cases} max\{u(t, p, s) + ru(s/_{t,p})\}, \\ 0. \end{cases} \tag{9}$$

where $max\{u(t, p, s) + ru(s/_{t,p})\}$ holds if $ru(s/_{t,p}) > 0$, otherwise, the $PEU(t, s)$ is set as 0.

Definition 16: $PEU(t)$ is used to denote the prefix extension utility of *t* in *q*-sequence, and defined as:

$$PEU(t) = \sum_{s \in D \wedge t \subseteq s} PEU(t, s) \tag{10}$$

Theorem 2 Given a sequence *t*, for each sequence t' that could be generated by *t* using concatenation operation, $u(t') \leq PEU(t)$

Proof: From the above definition, $u(t') \leq PEU(t')$ holds. $PEU(t', s) = max\{u(t', p', s) + ru(s/_{t',p'})\} = max\{u(t, p, s) + u(i_j) + ru(s/_{t',p'})\}$. i_j is the concatenation item at the concatenation position p'. since $p' \geq p$, $u(i_j) + ru(s/_{t',p'}) \leq ru(s/_{t,p})$, Therefore, $PEU(t', s) \leq max\{u(t, p, s) + ru(s/_{t,p})\} = PEU(t, s)$. Therefore, $PEU(t') = \sum_{s \in D \wedge t' \subseteq s} PEU(t', s) \leq \sum_{s \in D \wedge t' \subseteq s} PEU(t', s) \leq \sum_{s \in D \wedge t \subseteq s} PEU(t, s)$. Then $u(t') \leq PEU(t)$.

Pruning strategy 2: According to **Theorem 2**, For a given sequence *t*, if $PEU(t)$ is less than the minimum utility value, the utility of any sequences which could be generated by *t* will be less than the minimum utility value. And these sequences could be safely pruned from the *LE-tree* without affecting the complete mining results.

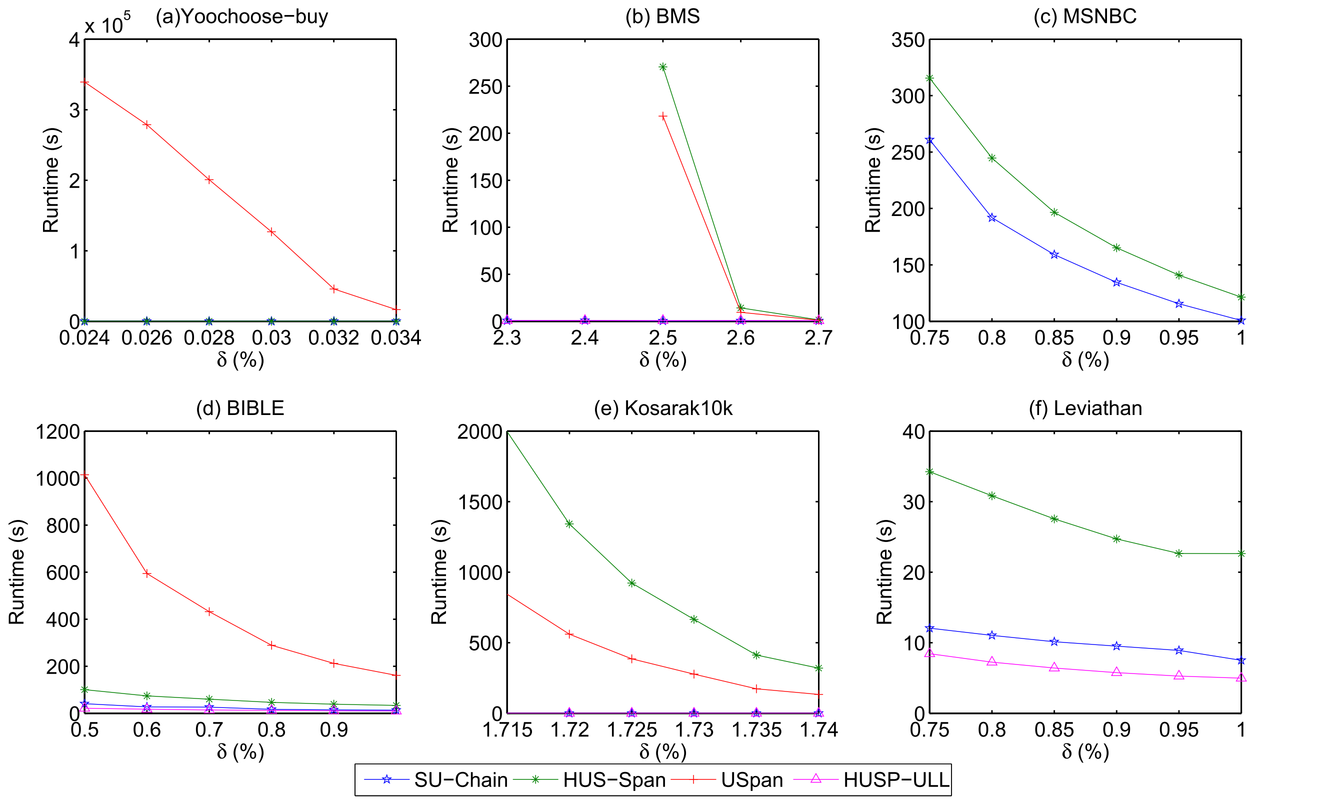

FIGURE 2. Runtime under various δ thresholds.

Furthermore, the pruning strategies used in the HUSP-ULL [8] can also be incorporated with the designed SU-Chain structure as follows.

Pruning strategy 3: Given a sequence t' and t, t' is generated by t and i_j using concatenation operation. Then i_j is the concatenation candidate item of sequence t. Thus, if $\sum_{s \in D \wedge t' \subseteq s} PEU(t, s)$ is less than the minimum utility value, then i_j called *unpromising item* is removed from the set of concatenation candidate items not to generate the sequence t'. Therefore, if i_j is the I-concatenation candidate item, then we can remove i_j from the set of I-concatenation items; if i_j is the S-concatenation candidate item, then we can remove i_j from the set of S-concatenation items.

Pruning strategy 4: Given an item i_j and a sequence t. t'_1 is generated by t and item i_j using I-concatenation; and t'_2 is generated by t and item i_j using S-concatenation. if $\sum_{s \in D \wedge t'_1 \subseteq s} PEU(t, s)$ is less than the minimum utility value and $\sum_{s \in D \wedge t'_2 \subseteq s} PEU(t, s)$ is less than the minimum utility value. Then we can remove the item i_j called *irrelevant item* from the projection database of the sequence t since this sequence is an super sequence generated by t and item i_j could be high utility sequential pattern.

Using the **Pruning strategy 4** to remove irrelevant items from projection database of a sequence t could reduce the size of the projection database of the sequence t and its super-sequence because the projection databases of these sequences do not need to contain the irrelevant items. As the same time, Removing the irrelevant items could lower the upper bound value of $PEU(t)$.

TABLE 4. Characteristics of the datasets.

Dataset	$\#\|D\|$	$\#\|I\|$	C	MaxLen
Yoochoose-buy	1,150,753	54,287	40	66
BMS	59,601	497	2.5	267
MSNBC	31,790	17	13.3	100
BIBLE	36,369	13,905	21.6	100
Kosarak10k	10,000	10,094	8.1	608
Leviathan	5,834	9,025	33.8	100

V. EXPERIMENTAL EVALUATION

In this section, several experiments were conducted to evaluate the proposed SU-Chain compared to the state-of-the-art USpan [32], HUS-Span [29] and HUSP-ULL [8] approaches. Six real-life datasets were used in the experiments to evaluate the performance in terms of runtime and number of generated candidates. The characteristics of six datasets are shown in Table 4. The parameters of the used datasets indicate: $\#|D|$ states the total number of sequences; $\#|I|$ is the number of distinct items; **C** is the average number of itemsets per sequence; and **MaxLen** states the maximum number of items per sequence.

A. RUNTIME

Experiments were conducted under the various minimum utility threshold δ and the results are then shown in Figure 2.

TABLE 5. Number of determined candidates and final HUSPs under various δ thresholds.

		# of Pattern under various δ values					
		δ_1	δ_2	δ_3	δ_4	δ_5	δ_6
(a) Yoochoose-buy	USpan	325,473	304,534	278,137	250,109	201,440	158,219
	HUS-Span	486,600	455,296	416,905	372,473	299,739	234,566
	HUSP-ULL	-	-	-	-	-	-
	SU-Chain	324,677	303,779	277,880	249,250	200,659	157,398
	#HUSPs	317,682	296,890	273,926	238,273	191,203	146,777
(b) BMS	USpan	-	-	9,824,707	279,130	10,258	1,231
	HUS-Span	-	-	10,236,255	300,523	11,008	1,487
	HUSP-ULL	376	214	203	186	166	155
	SU-Chain	413	228	221	215	206	196
	#HUSPs	2	2	0	0	0	0
(c) MSNBC	USpan	-	-	-	-	-	-
	HUS-Span	946,988	686,318	507,292	392,296	309,330	247,873
	HUSP-ULL	-	-	-	-	-	-
	SU-Chain	414,852	296,440	225,085	174,751	137,635	109,971
	#HUSPs	38,422	30,313	24,417	20,058	16,800	14,275
(c) BIBLE	USpan	92,565	59,044	40,768	29,183	21,227	16,488
	HUS-Span	100,867	64,332	44,377	31,800	23,134	17,970
	HUSP-ULL	15,855	10,377	7,352	5,476	4,187	3,277
	SU-Chain	15,932	10,463	7,457	5,544	4,255	3,365
	#HUSPs	2760	1714	1124	764	553	411
(c) Kosarak10k	USpan	37,459,553	24,629,747	16,781,104	12,150,844	7,461,373	5,548,726
	HUS-Span	43,831,361	28,854,740	19,677,668	14,256,083	8,762,924	6,517,837
	HUSP-ULL	672	659	647	622	603	575
	SU-Chain	1,363	1,351	1,340	1,325	1,315	1,304
	#HUSPs	22	22	22	21	21	21
(c) Leviathan	USpan	-	-	-	-	-	-
	HUS-Span	164,450	139,524	118,858	102,509	89,497	78,536
	HUSP-ULL	29,704	25,332	21,844	18,920	16,506	14,523
	SU-Chain	29,824	25,441	21,946	19,012	16,618	14,628
	#HUSPs	4,294	3,528	2,916	2,453	2,117	1,802

From the results, it can be observed that the designed SU-Chain-based algorithm outperforms the state-of-the-art USpan and HUS-Span algorithms in terms of runtime performance. The state-of-the-art HUSP-ULL algorithm has slightly better performance than that of the SU-Chain-based algorithm, for example in Figure 2(d), when the threshold is set as 1%, the SU-Chain-based model requires 13.1 seconds and the state-of-the-art HUSP-ULL needs 9.6 seconds. When the threshold is set as 1.715%, the SU-Chain-based model needs 0.79 seconds and the HUSP-ULL requires 0.2 seconds. When the threshold is set as 1%, the SU-Chain-based model requires 7.5 seconds while the HUSP-ULL needs 4.9 seconds. However, for the databases shown in Figures 2(a), 2(b), and 2(c), the designed SU-Chain-based model needs less runtime than that of the HUSP-ULL algorithm, especially the HUSP-ULL has the memory leakage problem in Figures 2(a) and 2(c). Generally, the designed SU-Chain-based algorithm can obtain better performance compared to the most HUSPM algorithms, especially it has better capacity to keep more information for efficiency improvement.

B. NUMBER OF GENERATED CANDIDATES

In order to evaluate the effectiveness of the compared algorithms, the number of generated candidates and the number of discovered HUSPs under different minimum utility thresholds are then conducted and shown in Table 5. The *#HUSPs* represents the number of HUSPs and "-" denotes that the runtime of the performed algorithm exceeds 10,000 seconds or it cannot be performed in a limited main memory.

From the given results, it can be observed that the designed SU-Chain-based algorithm generates less candidates than the previous USpan and HUS-Span algorithms. As the less candidates are required to be explored, less runtime is needed. When the minimum utility threshold increases, the number of the determined candidates decreases, and vice versa. This is reasonable since less patterns are then generated based on the higher minimum utility threshold. We also can observe that the USpan and HUS-Span cannot generate the results in the Yoochoose-buy dataset, and the HUSP-ULL cannot obtain the results both in Yoochoose-buy and MSNBC datasets. Although the HUSP-ULL has a very slight better performance than the SU-Chain-based algorithm (almost 1-2 seconds different), the number of generated candidates are nearly similar. However, the designed SU-Chain-based algorithm can obtain good performance for handling the Yoochoose-buy and MSNCBC datasets than that of the state-of-the-art HUSP-ULL approach.

VI. CONCLUSION AND FUTURE WORK

In this paper, we present a Sequence-Utility (SU)-chain structure to keep the projection database and its utility-list structure. Based on the designed SU-Chain-based model and the utilized pruning strategies, the SU-Chain-based algorithm successfully obtains good results than the other compared algorithms, especially the designed SU-Chain-based algorithm can reduce the leakage problem of the memory compared to the state-of-the-art HUSP-ULL approach. In the future, we will then address the dynamic situation to efficiently update the discovered HUSPs for transaction insertion based on the Hadoop or Spark platform. How to efficiently design a better structure used in the MapReduce framework is also an interesting topic for the further study.

REFERENCES

[1] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proc. Int. Conf. Very Large Data Bases*, 1994, pp. 487–499.

[2] R. Agrawal and R. Srikant, "Mining sequential patterns," in *Proc. Int. Conf. Data Eng.*, 1995, pp. 3–14.

[3] C. F. Ahmed, S. K. Tanbeer, and B.-S. Jeong, "Mining high utility Web access sequences in dynamic Web log data," in *Proc. 11th ACIS Int. Conf. Softw. Eng., Artif. Intell., Netw. Parallel/Distrib. Comput.*, Jun. 2010, pp. 76–81.

[4] C. F. Ahmed, "A novel approach for mining high-utility sequential patterns in sequence databases," *Electron. Telecommun. Res. Inst. J.*, vol. 32, no. 5, pp. 676–686, Oct. 2010.

[5] O. K. Alkan and P. Karagoz, "CRoM and HuspExt: Improving efficiency of high utility sequential pattern extraction," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 10, pp. 2645–2657, Oct. 2015.

[6] R. Chan, Q. Yang, and Y. D. Shen, "Mining high utility itemsets," in *Proc. IEEE Int. Conf. Data Mining*, Nov. 2003, pp. 19–26.

[7] W. Gan, J. C. W. Lin, J. Zhang, H. C. Chao, H. Fujita, and P. S. Yu, "ProUM: Projection-based utility mining on sequence data," *Inf. Sci.*, vol. 513, pp. 222–240, Mar. 2020.

[8] W. Gan, J. Chun-Wei Lin, J. Zhang, P. Fournier-Viger, H.-C. Chao, and P. S. Yu, "Fast utility mining on complex sequences," 2019, *arXiv:1904.12248*. [Online]. Available: http://arxiv.org/abs/1904.12248

[9] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.-C. Hsu, "FreeSpan: Frequent pattern-projected sequential pattern mining," in *Proc. 6th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2000, pp. 355–359.

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

[11] S. Kannimuthu and K. Premalatha, "Discovery of high utility itemsets using genetic algorithm with ranked mutation," *Appl. Artif. Intell.*, vol. 28, no. 4, pp. 337–359, Apr. 2014.

[12] M. Liu and J. Qu, "Mining high utility itemsets without candidate generation," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2012, pp. 55–64.

[13] Y. Liu, W. Liao, and A. N. Choudhary, "A two-phase algorithm for fast discovery of high utility itemsets," in *Proc. Pacific-Asia Conf. Adv. Knowl. Discovery Data Mining*, 2005, pp. 689–695.

[14] C.-W. Lin, T.-P. Hong, and W.-H. Lu, "An effective tree structure for mining high utility itemsets," *Expert Syst. Appl.*, vol. 38, no. 6, pp. 7419–7424, Jun. 2011.

[15] M. Liu and J. Qu, "Mining high utility itemsets without candidate generation," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2012, pp. 55–64.

[16] G.-C. Lan, T.-P. Hong, H.-C. Huang, and S.-T. Pan, "Mining high fuzzy utility sequential patterns," in *Proc. Int. Conf. Fuzzy Theory Its Appl. (iFUZZY)*, Dec. 2013, pp. 420–424.

[17] G.-C. Lan, T.-P. Hong, V. S. Tseng, and S.-L. Wang, "Applying the maximum utility measure in high utility sequential pattern mining," *Expert Syst. Appl.*, vol. 41, no. 11, pp. 5071–5081, Sep. 2014.

[18] J. C.-W. Lin, W. Gan, P. Fournier-Viger, T.-P. Hong, and V. S. Tseng, "Efficient algorithms for mining high-utility itemsets in uncertain databases," *Knowl.-Based Syst.*, vol. 96, pp. 171–187, Mar. 2016.

[19] J. Liu, K. Wang, and B. C. M. Fung, "Mining high utility patterns in one phase without generating candidates," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 5, pp. 1245–1257, Dec. 2016.

[20] J. C.-W. Lin, L. Yang, P. Fournier-Viger, T.-P. Hong, and M. Voznak, "A binary PSO approach to mine high-utility itemsets," *Soft Comput.*, vol. 21, no. 17, pp. 5103–5121, Mar. 2016.

[21] J. C. W. Lin, J. Zhang, and P. Fournier-Viger, "High-utility sequential pattern mining with multiple minimum utility thresholds," in *Proc. Int. Joint Conf. (APWeb-WAIM)*, 2017, pp. 215–229.

[22] P. Sharma and G. Balakrishna, "PrefixSpan: Mining sequential patterns by prefix-projected pattern," *Int. J. Comput. Sci. Eng. Surv.*, vol. 2, no. 4, pp. 111–122, Nov. 2011.

[23] R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements," in *Proc. Int. Conf. Extending Database Technol.*, 1996, pp. 3–17.

[24] B. E. Shie, J. F. Hsiao, V. S. Tseng, and P. S. Yu, "Mining high utility mobile sequential patterns in mobile commerce environments," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, 2011, pp. 224–238.

[25] V. S. Tseng, C. Wu, B. Shie, and P. S. Yu, "Up-growth: An efficient algorithm for high utility itemset mining," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 253–262.

[26] V. S. Tseng, B.-E. Shie, C.-W. Wu, and P. S. Yu, "Efficient algorithms for mining high utility itemsets from transactional databases," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 8, pp. 1772–1786, Aug. 2013.

[27] V. S. Tseng, C. W. Wu, P. Fournier-Viger, and S. Y. Philip, "Efficient algorithms for mining top-*K* high utility itemsets," *IEEE Trans. Knowl. Data Eng.*, vol. 8, no. 1, pp. 54–67, Jan. 2016.

[28] B. Vo, H. Nguyen, T. B. Ho, and B. Le, "A novel approach to extract high utility itemsets from distributed databases," *Comput. Informat.*, vol. 31, no. 6, pp. 1597–1615, 2013.

[29] J.-Z. Wang, J.-L. Huang, and Y.-C. Chen, "On efficiently mining high utility sequential patterns," *Knowl. Inf. Syst.*, vol. 49, no. 2, pp. 597–627, Jan. 2016.

[30] J. M.-T. Wu, J. Zhan, and J. C.-W. Lin, "An ACO-based approach to mine high-utility itemsets," *Knowl.-Based Syst.*, vol. 116, pp. 102–113, Jan. 2017.

[31] H. Yao, H. J. Hamilton, and C. J. Butz, "A foundational approach to mining itemset utilities from databases," in *Proc. SIAM Int. Conf. Data Mining*, Dec. 2013, pp. 482–486.

[32] J. Yin, Z. Zheng, and L. Cao, "USpan: An efficient algorithm for mining high utility sequential patterns," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2012, pp. 660–668.

[33] J. Yin, Z. Zheng, L. Cao, Y. Song, and W. Wei, "Efficiently mining top-*K* high utility sequential patterns," in *Proc. IEEE 13th Int. Conf. Data Mining*, Dec. 2013, pp. 1259—1264.

[34] L. Zhou, Y. Liu, J. Wang, and Y. Shi, "Utility-based Web path traversal pattern mining," in *Proc. 7th IEEE Int. Conf. Data Mining Workshops (ICDMW)*, Oct. 2007, pp. 373–380.

[35] S. Zida, pp. Fournier–Viger, J. C. W. Lin, C. W. Wu, and V. S. Tseng, "EFIM: A fast and memory efficient algorithm for high-utility itemset mining," *Knowl. Inf. Syst.*, vol. 51, no. 2, pp. 595–625, 2017.

JERRY CHUN-WEI LIN (Senior Member, IEEE) received the Ph.D. degree from the Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan. He is currently a Full Professor with the Department of Computer Science, Electrical Engineering and Mathematical Sciences, Western Norway University of Applied Sciences, Bergen, Norway. He is also the Project Co-Leader of well-known SPMF: An Open-Source Data Mining Library, which is a toolkit offering multiple types of data mining algorithms. He has published more than 300 research articles in refereed journals, such as the IEEE TKDE, IEEE TCYB, ACM TKDD, and ACM TDS, and international conferences, such as the IEEE ICDE, IEEE ICDM, PKDD, and PAKDD. His research interests include data mining, soft computing, artificial intelligence and machine learning, and privacy-preserving and security technologies. He is a Fellow of the IET and ACM. He serves as the Editor-in-Chief for the *International Journal of Data Science* and *Pattern Recognition*.

YUANFA LI is currently pursuing the master's degree with the School of Computer Science and Technology, Harbin Institute of Technology (Shenzhen), China. His research interests include big data analytics and cloud computing.

PHILIPPE FOURNIER-VIGER received the Ph.D. degree in computer science from the University of Quebec, Montreal, in 2010. He is currently a Full Professor and a Youth 1000 Scholar with the Harbin Institute of Technology (Shenzhen), Shenzhen, China. He is the Founder of the popular SPMF open-source data mining library, which has been cited in more than 800 research articles. His research interests include pattern mining, sequence analysis and prediction, and social network mining. He is also the Editor-in-Chief (EiC) of the *Data Mining and Pattern Recognition* (DSPR) journal.

YOUCEF DJENOURI received the Ph.D. degree in computer engineering from the University of Science and Technology Houari Boumediene (USTHB), Algiers, Algeria, in 2014. From 2014 to 2015, he was a Permanent Teacher–Researcher with the University of Blida, Algeria, where he is currently a member of LRDSI Lab. He was granted a Postdoctoral Fellowship from Unist University, South Korea. He worked on the BPM Project supported by Unist University, in 2016. In 2017, he was a Postdoctoral Research with Southern Denmark University, where he has working on urban traffic data analysis. He was granted a Postdoctoral Fellowship from the European Research Consortium on Informatics and Mathematics (ERCIM). He worked with the Norwegian University of Science and Technology (NTNU), Trondheim, Norway. He is currently a Researcher Scientist with SINTEF Digital, Oslo, Norway. He is working on topics related to artificial intelligence and data mining, with a focus on association rules mining, frequent itemsets mining, parallel computing, swarm and evolutionary algorithms, and pruning association rules. He has published over 24 refereed conference papers, 20 international journal articles, two book chapters, and one tutorial article in the areas of data mining, parallel computing, and artificial intelligence.

JI ZHANG (Senior Member, IEEE) received the B.E. degree from the Department of Information Management and Information Systems, Southeast University, China, in 2000, the M.Sc. degree from the Department of Computer Science, National University of Singapore, in 2002, and the Ph.D. degree from the Faculty of Computer Science, Dalhousie University, Canada, in 2008. From 2008 to 2009, he was a Postdoctoral Research Fellow with the CSIRO ICT Center, Hobart, Australia. He is currently an Associate Professor in computing with the University of Southern Queensland (USQ), Australia. He has published over 140 articles in major peer-reviewed international journals and conferences. His research interests are in big data analytics, knowledge discovery and data mining (KDD), and information privacy and security. He is a member of ACM, a Fellow of Australian Endeavour, a Fellow of Queensland, Australia, and a Scholar of Izaak Walton Killam, Canada.