Journal of Computer Science and Cybernetics, V.35, N.3 (2019), 217–232
DOI 10.15625/1813-9663/35/3/13277

FUZZY COMMON SEQUENTIAL RULES MINING IN QUANTITATIVE SEQUENCE DATABASES

DO VAN THANH[1,*], TRUONG DUC PHUONG[2]

Department of Information Technology, Nguyen Tat Thanh University, Vietnam
Department of Information Technology, Hanoi Metropolitan University, Vietnam
[*] *dvthanh@ntt.edu.vn*

Abstract. Common sequential rules present a relationship between unordered itemsets in which the items in antecedents have to appear before ones in consequents. The algorithms proposed to find such rules so far are only applied for transactional sequence databases, not applied for quantitative sequence databases. The goal of this paper is to propose a new algorithm for finding the fuzzy common sequential (FCS for short) rules in quantitative sequence databases. The proposed algorithm is based on the ERMiner algorithm. It is considered to be the most effective today compared to other algorithms for finding common sequential rules in transactional sequence databases. FCS rules are more general than classical fuzzy sequential rules and are useful in marketing, market analysis, medical diagnosis and treatment.

Keywords. Quantitative Sequence Database; Fuzzy Sequence Database; Fuzzy Common Sequential Rule; Equivalence Class; Left Merger; Right Merger.

1. INTRODUCTION

Mining sequential rules is one of the most important domains in data mining. There are two kinds of sequential rules [7, 9]. The first kind of sequential rules expresses a relationship between two series of events happening one after another. In these rules, both of the antecedent and consequent parts belong in a same sequential pattern and to discover such rules, one often focuses on the mining of sequential patterns [2, 3, 4, 6, 11, 12, 13, 14, 15, 16, 18]. In the second kind of sequential rules, items in the antecedent or consequent parts do not always appear in the same order in sequences as long as items in the antecedent part need to appear before items in the consequent part [8, 9, 10].

Like the process of mining association rules [1], in general the process of mining of sequential rules of the first kind consists of two phases, in which the first phase is to discover frequent sequences(sequential patterns), and the second phase is to generate sequential rules from the discovered sequential patterns. The first phase is the most complex, most time and cost consuming. At present, there are many works for finding out sequential patterns of the first kind in transactional sequence databases as well as quantitative sequence databases [2, 3, 4, 6, 11, 12, 13, 14, 15, 16, 17, 18, 19].

Sequential rule of the second kind has just been mentioned in recent years [7, 8, 9, 10]. It is more general than the sequential rule of the first kind. It is actually useful and has been applied in practice [5]. Sequential rule of the second kind is called common sequential rule [7, 8, 9, 10]. Unlike the approach to mine sequential rules of the first kind, the proposed algorithms to mine sequential rules of the second kind so far do not discover sequential patterns, and since then generate valid sequential rules, they find rules that are common to several sequences [7, 8, 9, 10].

Details of the algorithms mining the common sequential rules in transactional sequence databases are presented in the papers [8, 9, 10]. In [9], the authors introduced an algorithm for mining common sequential rules in transactional sequence databases. It is built based on equivalence classes and is called ERMiner. This algorithm has overcome some of the disadvantages of previous algorithms [8, 10]. The experiment of the algorithms mining the common sequential rules in some transactional sequence databases having different properties showed that the ERMiner algorithm ran the fastest but in general it uses more memory than other algorithms [9].

It can be seen that the common sequential rules found so far have only been discovered in transactional sequence databases, there has been no studies on the mining of these rules in quantitative sequence databases, where attributes receive numeric and/or categorical values.

The purpose of this paper is to address the aforementioned shortcoming. Specifically, this paper proposes an algorithm to mine common sequential rules in quantitative sequence databases. In such context, the found rules are called fuzzy common sequential (FCS for short) rules and the proposed algorithm is named FERMiner. The FERMiner algorithm differs from the ERMiner algorithm mainly in that it has to convert quantitative sequence databases into fuzzy sequence databases and proposes formulas to calculate the support and the confidence of FCS rules.

The rest of the paper is organized as follows: Section 2 defines the problem of mining FCS rules. Section 3 presents the FERMiner algorithm to find out FCS rules. Section 4 experiments the proposed algorithm. Conclusions and orientations for further research are presented in Section 5.

2. PROBLEM DEFINITION

Definition 2.1. Let $E = \{e_1, e_2,..., e_u\}$ be a set of attributes, $<_{lex}$ be a total order relation of attributes in E and $e_1 <_{lex} e_2 <_{lex},\dots, <_{lex} e_u$, $s = \langle(e_{t1}, q_1), e_{t2}, q_2), (e_{t3}, q_3),\dots, (e_{tn}, q_n)\rangle$ is a quantitative sequence, where $e_{tk} \in E$ $(1\leq k \leq n)$, q_k is value of e_{tk} (q_k is numeric or categorical).

A quantitative sequence database denoted by QSD is the set of all quantitative sequences. So, $QSD = \{s_1, s_2, ..., s_v\}$ where s_i $(1\leq i \leq v)$ is a quantitative sequence, v is the total number of quantitative sequences. In a sequence, elements occurring in a same time are sorted by the $<_{lex}$ relation of the attributes. A quantitative sequence s can be presented in an alternate form s=$\langle E_1, E_2, ...,E_k\rangle$, where $\cup_{h=1}^{k} E_h = \{(e_{t1}, q_1), (e_{t2}, q_2), (e_{t3}, q_3),\dots, (e_{tn}, q_n)\}$ and all attributes e_{ti} in E_h occur in a same time. E_h is called a transaction, the sequence s=$\langle E_1, E_{2,..,}E_k\rangle$ is called a transaction sequence.

Example 2.2. Table 1 presents a quantitative sequence database. In this case, set of attributes is $E = \{a, b, c, d, e, f, g, h, i\}$. The second quantitative sequence is $\langle(d, 2),(a, 5),(d, 4)\rangle$

in which the attribute d receives a value of 2 and does not occur at the same time with the attribute a having a value of 5. Furthermore, the quantitative sequence $\langle (d,2), (a,5), (d,4)\rangle$ also can be presented in form $\langle \{(d,2)\}, \{(a,5), (d,4)\}\rangle$ or $\langle \{(d,2)\}, \{(a,5)\}, \{(d,4)\rangle$ depending on the time points of occurrence of the attributes a and d to be the same or not. The sequence $\langle \{(d,2)\}, \{(a,5), (d,4)\}\rangle$ consists of 2 transactions such as $\{(d,2)\}$ and $\{(a,5), (d,4)\}$ whereas the sequence $\langle \{(d,2)\}, \{(a,5)\}, \{(d,4)\}\rangle$ consists of 3 transactions such as $\{(d,2)\}, \{(a,5)\}$ and $\{(d,4)\}$.

Table 1. Quantitative sequence database, D

Cid	**Sequences**
1	$\langle (a,2), (b,2), (e,5)\rangle$
2	$\langle (d,2), (a,5), (d,4)\rangle$
3	$\langle (b,1), (d,2), (e,5)\rangle$
4	$\langle (f,6), (b,6), (c,1), (c,2)\rangle$
5	$\langle (a,1), (b,1), (d,2), (e,5)\rangle$
6	$\langle (a,2), (b,1), (e,1)\rangle$
7	$\langle (i,5), (a,3), (h,2)\rangle$
8	$\langle (c,6), (i,5), (f,3)\rangle$
9	$\langle (h,3), (a,1), (b,6)\rangle$
10	$\langle (a,2), (g,5), (b,2), (e,1)\rangle$

Definition 2.3. Let $FE = \{F^{e_1}, F^{e_2}, ..., F^{e_u}\}$ be a set of fuzzy sets of attributes in E, $F^{e_k} = \left\{f^{e_k}_{h_k,1},\ f^{e_k}_{h_k,2},\ ...,f^{e_k}_{h_k,h_k}\right\}$ be a set of fuzzy sets of the e_k attribute ($k = 1, 2..., u$), where $f^{e_k}_{h_k,j}$ is the j^{th} fuzzy set ($1\leq\ j\ \leq\ h_k$), h_k is the number of fuzzy sets of e_k. Each fuzzy set has its membership function μ: $X \to [0,1]$. Sequence $fs = \langle (fe_1,\ fq_1),\ (fe_2,\ fq_2),\ (fe_3,\ fq_3), \ldots, (fe_n,\ fq_n)\rangle$ is called a fuzzy sequence, where $fe_i \in FE$ $(1 \leq i \leq n)$ is a fuzzy set and is also called a fuzzy attribute, fq_i is the value of the membership function μ_{fe_i} of fe_i at q_i $(fq_i = \mu_{fe_i}(q_i))$. A fuzzy sequence database (FSD for short) is a set of all fuzzy sequences.

Similar to quantitative sequences, fuzzy sequences can also be presented like $fs = \langle E_1,\ E_2, ..., E_k\rangle$, where $\cup_{h=1}^{k} E_h = \langle (fe_1,\ fq_1),\ (fe_2,\ fq_2),\ (fe_3,\ fq_3), \ldots,\ (fe_n,\ fq_n)\rangle$ and all attributes in E_h occur at a same time. E_h is called a fuzzy transaction, $fs = \langle E_1,\ E_2, ...,E_k\rangle$ is called a fuzzy transaction sequence. Denote $I_h = \{fe_i/(fe_i,\ fq_i) \in E_h\}$, it is called a short form of E_h.

Example 2.4. Assume each attribute x_m in a quantitative sequence database is associated with K fuzzy sets defined as follows: Let $f^{x_m}_{K,\ i_m}$ be i^{th}_m fuzzy set ($1\leq\ \ i_m\ \leq\ K$) of the attribute x_m ($x_m \in E$), mi and ma are respectively the minimum and maximum values of the attribute x_m, K is the number of partitions of x_m; $\mu^{x_m}_{K,\ i_m}$ is the membership function of $f^{x_m}_{K,\ i_m}$ and is determined as in [16], i.e.

$$\mu^{x_m}_{K,\ i_m}(v) = \max\{1 - |v - a^{K}_{i_m}|/b^{K}, 0\} \tag{2.1}$$

where $a_{i_m}^K = mi + (ma - mi)(i_m - 1)/(K-1);\ b^K = (ma - mi)/(K-1)$.

Suppose $K = 3$ for all attributes in the quantitative sequence database D in Example 2.2. The quantitative attribute a will be converted to 3 fuzzy attributes $f_{3,1}^a$ (i_m=1), $f_{3,2}^a$ (i_m=2), $f_{3,3}^a$ (i_m=3) with their corresponding membership functions to be $\mu_{3,1}^a$, $\mu_{3,2}^a$, $\mu_{3,3}^a$ as follows.

Due to the minimum and maximum values of the attribute a in D are $mi = 1,\ ma = 5$, according to the formula (2.1)

$$a_1^3 = 1 + (5-1)(1-1)/(3-1) = 1;$$

$$a_2^3 = 1 + (5-1)(2-1)/(3-1) = 3;$$

$$a_3^3 = 1 + (5-1)(3-1)/(3-1) = 5$$

and $b^3 = (5-1)/(3-1) = 2$, so

$$\mu_{3,1}^a(v) = \max\left\{1 - \left|v - a_1^3\right|/b^3,\ \ 0\right\} = \max\left\{1 - |v-1|/2,\ \ 0\right\},$$

$$\mu_{3,2}^a(v) = \max\left\{1 - \left|v - a_2^3\right|/b^3, 0\right\} = \max\left\{1 - |v-3|/2, 0\right\},$$

and

$$\mu_{3,3}^a(v) = \max\left\{1 - \left|v - a_3^3\right|/b^3, 0\right\} = \max\left\{1 - |v-5|/2,\ \ 0\right\}.$$

The graph of membership functions of the 3 fuzzy attributes $f_{3,1}^a$, $f_{3,2}^a$, $f_{3,3}^a$ is illustrated in Figure 1 below

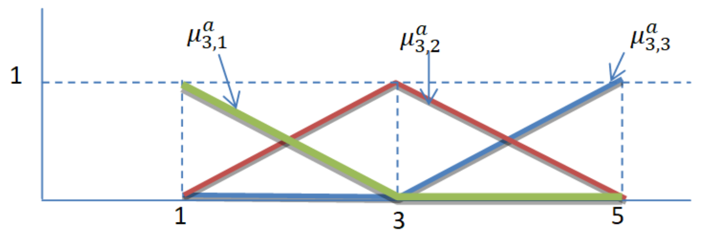

Figure 1. The graph of membership functions $\mu_{3,1}^a, \mu_{3,2}^a, \mu_{3,3}^a$

Calculating the value of membership functions:

In Cid = 1 in the quantitative sequence database D, the attribute a =2, so according to the formula (2.1), $\mu_{3,1}^a$ (2)=max{1-|2-1|/2, 0}= 0.5; $\mu_{3,2}^a$ (2)=max{1-|2-3|/2, 0}= 0.5 and $\mu_{3,3}^a$ (2)=max{1-|2-5|/2, 0}= max{1-1.5, 0}= 0.

In $Cid = 2$, the attribute a=5, by calculating under the similar way as mentioned above, we have $\mu_{3,1}^a(5) = \max\{1 - |5-1|/2, 0\} = 0$; $\mu_{3,2}^a(5) = \max\{1 - |5-3|/2,\ 0\} = 0$ and $\mu_{3,3}^a(5) = \max\{1 - |5-5|/2, 0\} = \max\{1, 0\} = 1$. The value of membership functions of the 3 fuzzy attributes associated with the quantitative attribute a in another $Cids$ in D is also calculated under the such a similar way.

Determining fuzzy attributes associated with each quantitative attribute in D as well as calculating the value of their membership functions are implemented similarly as for the attribute a, and finally we get the fuzzy sequence database D^* from the quantitative sequence database D as described in Table 2.

Table 2. Fuzzy sequence database, D^*

Cid	Fuzzy Sequence
1	$\langle (f^a_{3,1}, 0.5), (f^a_{3,2}, 0.5), (f^b_{3,1}, 0.6), (f^b_{3,2}, 0.4), (f^e_{3,3}, 1) \rangle$
2	$\langle (f^d_{3,1}, 1), (f^a_{3,3}, 1), (f^d_{3,1}, 1) \rangle$
3	$\langle (f^b_{3,1}, 1), (f^d_{3,1}, 1), (f^e_{3,3}, 1) \rangle$
4	$\langle (f^f_{3,3}, 1), (f^b_{3,3}, 1), (f^c_{3,1}, 1), (f^c_{3,1}, 0.6), (f^c_{3,2}, 0.4) \rangle$
5	$\langle (f^a_{3,1}, 1), (f^b_{3,1}, 1), (f^d_{3,1}, 1), (f^e_{3,3}, 1) \rangle$
6	$\langle (f^a_{3,1}, 0.5), (f^a_{3,2}, 0.5), (f^b_{3,1}, 1), (f^e_{3,1}, 1) \rangle$
7	$\langle (f^i_{3,1}, 1), (f^a_{3,2}, 1), (f^h_{3,1}, 1) \rangle$
8	$\langle (f^c_{3,3}, 1), (f^i_{3,1}, 1), (f^f_{3,1}, 1) \rangle$
9	$\langle (f^h_{3,3}, 1), (f^a_{3,1}, 1), (f^b_{3,3}, 1) \rangle$
10	$\langle (f^a_{3,1}, 0.5), (f^a_{3,2}, 0.5), (f^g_{3,1}, 1), (f^b_{3,1}, 0.6), (f^b_{3,2}, 0.4), (f^e_{3,1}, 1) \rangle$

In this table a tuple $(f^{x_m}_{K,i_m}, fq)$ means that $f^{x_m}_{K,i_m}$ is a fuzzy set of the attribute x_m, $f_q \in [0, 1]$ is the value of the membership function $\mu^{x_m}_{K,i_m}$ at the value q. For instance, in case of $(f^a_{3,2}, 0.5)$, $f^a_{3,2}$ is the second fuzzy set of three fuzzy sets for the attribute a, 0.5 is value of the membership function $\mu^a_{3,2}$ of the fuzzy set $f^a_{3,2}$ at $q = 2$.

Definitions 2.5, 2.6, 2.8 below are developed from the related definitions in the works [8, 9, 10].

Definition 2.5. A FCS rule $X \Rightarrow Y$ expresses a relationship between two sets of fuzzy attributes X and Y so that $X \cap Y = \varnothing; X \neq \varnothing$ and $Y \neq \varnothing$, and Y appears after X in a fuzzy transaction sequence fs.

Definition 2.6. Let $fs = \langle E_1, E_2, ...,, E_k \rangle$ be a fuzzy transaction sequence; $I_1, I_2, ...,, I_k$ be the short forms of $E_1, E_2, ...,, E_k$, respectively; a fuzzy attribute set X appears (or is contained) in fs if there exists an interger m, $m \leq k$ so that $X \subseteq \cup^m_{h=1} I_h$; a FCS rule $r = X \Rightarrow Y$ appears (or is contained) in fs if there exists an integer $n < k$ so that $X \subseteq \cup^n_{h=1} I_h$ and $Y \subseteq \cup^k_{h=n+1} I_h$. A fuzzy sequence is said to contain a FCS rule r if its corresponding fuzzy transaction sequence contains r.

A fuzzy sequence $\alpha = \langle (fa_1, faq_1), (fa_2, faq_2), (fa_3, faq_3), \ldots, (fa_n, faq_n) \rangle$ is a subsequence of a fuzzy sequence $\beta = \langle (fb_1, fbq_1), (fb_2, fbq_2), (fb_3, fbq_3), \ldots, (fb_r, fbq_r) \rangle$ if there are integers $1 \leq w_1 < w_2 < \ldots < w_n \leq r$ so that $fa_i = fb_{wi}$ and $faq_i = fbq_{wi}$ with $\forall i | 1 \leq i \leq n$.

Example 2.7. A fuzzy attribute set $\{f^a_{3,2}, f^b_{3,1}\}$ is contained in the fuzzy sequence

$$\langle \{(f^a_{3,1}, 0.5), (f^a_{3,2}, 0.5)\}, \{(f^b_{3,1}, 0.6), (f^b_{3,2}, 0.4)\}, \{(f^e_{3,3}, 1)\} \rangle.$$

A FCS rule $\{f^a_{3,1}, f^g_{3,1}\} \Rightarrow \{f^b_{3,2}, f^e_{3,1}\}$ is contained in the fuzzy sequence

$$\langle \{(f^a_{3,1}, 0.5), (f^a_{3,2}, 0.5)\}, \{(f^g_{3,1}, 1)\}, \{(f^b_{3,1}, 0.6), (f^b_{3,2}, 0.4)\}, \{(f^e_{3,1}, 1)\} \rangle.$$

There does not exist the FCS rule $\{f^a_{3,1},\ f^b_{3,1}\} \Rightarrow\ \{f^g_{3,1}\}$ because $\{f^g_{3,1}\}$ is not contained in the short form of a fuzzy transaction occured after the fuzzy transaction with its short form containing the fuzzy attribute $f^b_{3,1}$.

Definition 2.8. A FCS rule $r = X \Rightarrow Y$ has the size of $k * m$ if $|X|\ = k$ and $|Y|\ = m$. A FCS rule with the size of $f * g$ is greater than an other FCS rule with the size of $h * i$ if either $f > h \wedge g \geq i$ or $f \geq h \wedge g > i$.

Definition 2.9. Let $fs = \langle (fe_1, fq_1), (fe_2,\ fq_2),\ (fe_3,\ fq_3), \ldots, (fe_n,\ fq_n)\rangle$ be a fuzzy sequence, fuzzy attribute set X be contained in fs the support for X of fs is computed by

$$\gamma\,(fs) = \prod_{i=1}^{n} f_{qi}. \tag{2.2}$$

The support of the fuzzy attribute set X in the fuzzy sequence database FSD is computed as follows

$$supp\,(X) = \frac{1}{|FSD|}\ \sum \gamma(fs),\ \ fs \in FSD \text{ and }\ \ X \subseteq fs. \tag{2.3}$$

Let $r = X \Rightarrow Y$ be a FCS rule, the support of the rule r in the fuzzy sequence database FSD is defined by

$$supp(r) = supp(X \cup Y). \tag{2.4}$$

In other words, supp(r) is the percentage of the total of the support for the set $X \cup Y$ of fuzzy sequences in which the fuzzy attributes in X must appear before the attributes in Y, divided by the total number of fuzzy sequences in FSD.

The confidence of the FCS rule $r = X \Rightarrow Y$ is defined by

$$conf\,(r) = \frac{supp(r)}{supp(X)} \tag{2.5}$$

Remark 1. If all attributes in a quantitative sequence database have the value of 0 or 1 then the support of a rule r computed according to Definition 2.9 is equal to the support of this rule computed as in the case of transactional sequence databases [9]. Based on the formulas (2.2) and (2.3), it can be deduced that the support of fuzzy attribute sets has the Apriori property [1].

Definition 2.10. Let *minSup*, *minConf* $\in [0, 1]$ be two user-defined thresholds. FCS rule r is frequent if $supp(r) \geq$ *minSup*. The rule r is *confident* if $conf(r) \geq$ *minConf*. FCS rule r is called a valid if it is frequent and confident rule.

3. THE FERMiner ALGORITHM

Given a quantitative sequence database QSD, two user-defined thresholds *minSup* and *minConf*, a set of fuzzy sets FE of the quantitative attributes in QSD, each fuzzy set $f \in FE$ has its membership function.

The arised problem: find out all valid FCS rules in QSD.

The algorithm for finding all valid FCS rules is developed based on the ERMiner algorithm using equivalence classes. The following concepts are developed from the related concepts in [9].

Definition 3.1. (A left/right fuzzy equivalence class and left/right mergers) Given a fuzzy sequence database, let $\mathfrak{R}$ be a set of all frequent FCS rules, $\mathcal{E}$ be a set of all fuzzy attributes of E. A left fuzzy equivalence class $L\mathcal{E}_{W,i}$ is the set of frequent FCS rules $L\mathcal{E}_{W,i} = \{W \Rightarrow Y|W,\ Y \subseteq\ \mathcal{E} \wedge |Y| = i,\ i$ is an integer$\}$. Similarly, a right fuzzy equivalence class $R\mathcal{E}_{W,i}$ is the set of frequent FCS rules $R\mathcal{E}_{W,i} = \{X \Rightarrow W|X,\ W \subseteq\ \mathcal{E} \wedge |X| = i\}$.

Assume two FCS rules $r_1,\ r_2 \in\ L\mathcal{E}_{W,i},\ r_1 = W \Rightarrow X,\ r_2 = W \Rightarrow Y$ and $|X \cap Y| = |X - 1| = i - 1$, i.e. X and Y are identical except for a single fuzzy attribute; a left merger of $r_1,\ r_2$ is the process of merging r_1 and r_2 to obtain $r = W \Rightarrow X \cup Y$. Similary, assume two FCS rules $r_1,\ r_2 \in\ R\mathcal{E}_{W,i},\ r_1 = X \Rightarrow W,\ r_2 = Y \Rightarrow W$ and $|X \cap Y| =\ |X - 1| = i - 1$, a right merger of $r_1,\ r_2$ is the process of merging r_1 and r_2 to obtain $r = X \cup Y \Rightarrow W$.

Property 1. Let $W \Rightarrow Y$ be a frequent FCS rule, if $X \subseteq Y$ then $W \Rightarrow X$ is also frequent FCS rule. Similarly, let $Y \Rightarrow W$ be a frequent FCS rule, if $X \subseteq Y$ then $X \Rightarrow W$ is also frequent FCS rule.

Property 2. All frequent FCS rules $r = W \Rightarrow Y,\ |Y| = i + 1$ are results of a left merger of two FCS rules $r_1,\ r_2$ belonging to the left fuzzy equivalence class $L\mathcal{E}_{W,i}$. Similarly, all frequent FCS rules $r = Y \Rightarrow W,\ |Y| = i + 1$ are results of a right merger of two FCS rules $r_1,\ r_2$ belonging to the right fuzzy equivalence class $R\mathcal{E}_{W,i}$.

In essence, the proof of the Properties 1, 2 is based on the Apriori property of candidate itemsets and the way of generating a candidate k-itemsets from two frequent $(k-1)$-itemsets in the Apriori algorithm [1]. The proof of these properties is similar to the proof of related properties in [9] and is simple, so it is omitted here. From the two properties, we have a following remark.

Remark 2:

- FCS rule r generated by merging two FCS rules $r_1,\ r_2$ always has the support less than or equal to the support of these two FCS rules.

- If the support of a FCS rule r less than *minSup* then this rule is not merged with any FCS rule to generate a new frequent FCS rule.

This remark is used to prune the search space of frequent FCS rules. On the other hand, due to a FCS rule may be created under different combination ways of left and right mergers, so it can generate redundant FCS rules. In order to overcome this drawback, this paper uses the solution proposed in [9]. Namely, this solution only allows to perform a right merger after a left merger and does not allow to perform a left merger after a right merger.

Similarly to the paper [9], because there is not a pruning for the confidence, so in order to find valid FCS rules, the computation of the confidence of FCS rules done in the space of frequent FCS rules is a way to reduce the search space of valid FCS rules.

The FERMiner algorithm for finding valid FCS rules in quantitative sequence databases is developed based on the ERMiner algorithm [9] and is as follows: In this algorithm, the *fleftSore* variable stores all left fuzzy equivalence classes and the *frules variable* stores all valid FCS rules. The **fleftSearch** procedure performs merging of all left fuzzy equivalence classes.

Algorithm 1 (The FERMiner algorithm)

```
Input: Let $QSD$ be a quantitative sequence database; $minSup$, $minConf$ be user's thresholds; $FE$ be fuzzy
    sets of the quantitative attributes in $QSD$;
Output: Valid FCS rules;
 1: fleftStore ← ∅;
 2: frules ← ∅;
 3: Scanning $QSD$ once to create fuzzy sequence database $FSD$.
 4: Calculating $EQ$ that is the equivalence class with size of 1*1;
 5: Foreach left equivalence class $H \in EQ$ do
 6:     fleftSearch($H$, frules)
 7: end
 8: Foreach right equivalence class $J \in EQ$ do
 9:     frightSearch($J$, frules, fleftStore)
10: end
11: Foreach left equivalence class $K \in$ fleftStore do
12:     fleftSearch($K$, rules)
13: end
14: Return frules
```

The **frightSearch** procedure performs merging of all right fuzzy equivalence classes, and this procedure also allows a left merger to be made after a right merger, so it can generate new left fuzzy equivalence classes. Because of this, after finishing the **frightSearch** procedure, the **fleftSearch** procedure must be performed again to find out all valid FCS rules.

The FERMiner algorithm is similar to the ERMiner algorithm [9]. The main differences between these two algorithms are that after scaning the quantitative sequence database QSD, the FERMiner algorithm will transform this database into a fuzzy sequence database FSD and the support of FCS rules in the FERMiner algorithm is computed according to the formulas (2.3), (2.4), (2.5) above. The computation is implemented in the **fleftSearch** and **frightSearch** procedures.

Algorithm 2 (**fleftSearch** procedure)

```
Input: $LE$ is the left fuzzy equivalence class; frules: set of valid FCS rules found up to this time;
 1: fleftStore ← ∅;
 2: foreach $r \in LE$ do
 3:     $LE' \leftarrow \varnothing$;
 4:     foreach rule $s \in LE$ so that $r \neq s$ & pair $(r, s)$ has not been processed do
 5:         If (MergingCondition) = true then
 6:             $t \leftarrow RightMerge(r, s)$;
 7:             ComputeSupport($t, r.s$);
 8:             If $Supp(t) \geq$ minSup then
 9:                 $LE' \leftarrow LE' \cup \{t\}$;
10:                 ComputeConfidence $(t, r, s)$;
11:                 If Conf(t) $\geq$ minConf then
12:                     frules ← frules $\cup \{t\}$;
13:                 end
14:             end
15:         end
16:     end do
17:     fleftSearch($LE'$, frules);
18: end do
```

Here *MergingCondition* = true means $|X \cap Y| = |X - 1|$, where X and Y are consequent parts of the rules r and s, respectively.

Unlike the ERMiner algorithm, the **fleftSearch** procedure in the FERMiner algorithm does not use the sparse matrix structure to prune the search space when checking the left/right merging conditions of the two rules in a same equivalence class [8, 9]. The main reason is that the computation of the support of a fuzzy attribute is rather complex due to it must be implemented according to the formulas (2.3) and (2.4). Except this, the **fleftSearch** procedure in the FERMiner algorithm is quite similar to the **leftSearch** procedure in the ERMiner algorithm [9]. The same remark is also true for the **frightSearch** procedure in the FERMiner algorithm, so this procedure is ignored and is not introduced in the paper.

4. EXPERIMENT

The algorithm is executed in the Java programming language and run on Chip Intel Core i5 2.5 GHz, RAM 4 GB, Windows 7 OS.

4.1. Data sets

The Online Retail and the QtyT40I10D100K are very large datasets [20]. The Online Retail is the retail dataset of 37 countries with 541.909 instances and 3.684 items (Stockcode) while the QtyT40I10D100K is the dataset of 100 customers with 3.960.456 instances in which the *Time* attribute receives 99.999 values and the *Trans* attribute receives 942 values (items). Two datasets for experiment include the Online Retail_France dataset extracted from the Online Retail [20] from December 1, 2010 to December 9, 2011 with the value of the *Country* attribute to be 'France' and the QtyT40I10D100K_10K dataset extracted from the QtyT40I10D100K [20] with the first 10.000 transactions (value of the *Time* attribute from 1 to 10.000) and with the first 100 items (value of the *Trans* attribute from 1 to 100).

The Online Retail_France includes the information as follows:

- Customer ID – the identify of the customer;
- Invoice Date – the date of the invoice;
- Stock Code – the code of the Stock;
- Quantity – the quantity of the bought StockCode.

The QtyT40I10D100K_10K has the information as follows:

- CustomerID – the identify of the customer;
- Time – the time of the transaction;
- Trans – the value of an item;
- Quantity – the quantity of the item.

The characteristics of the two experimental data sets are described in Table 3.

Table 3. Experimental datasets

Datasets	**Number of attributes (I)**	**Number of transactions (D)**	**Number of sequences (S)**	**Average of the length of transactions (T)**	**Average of the length of sequences**
Online Retail_France	1523	365	87	21.38	95.88
QtyT40I10D100K_10K	100	10000	100	4.26	420

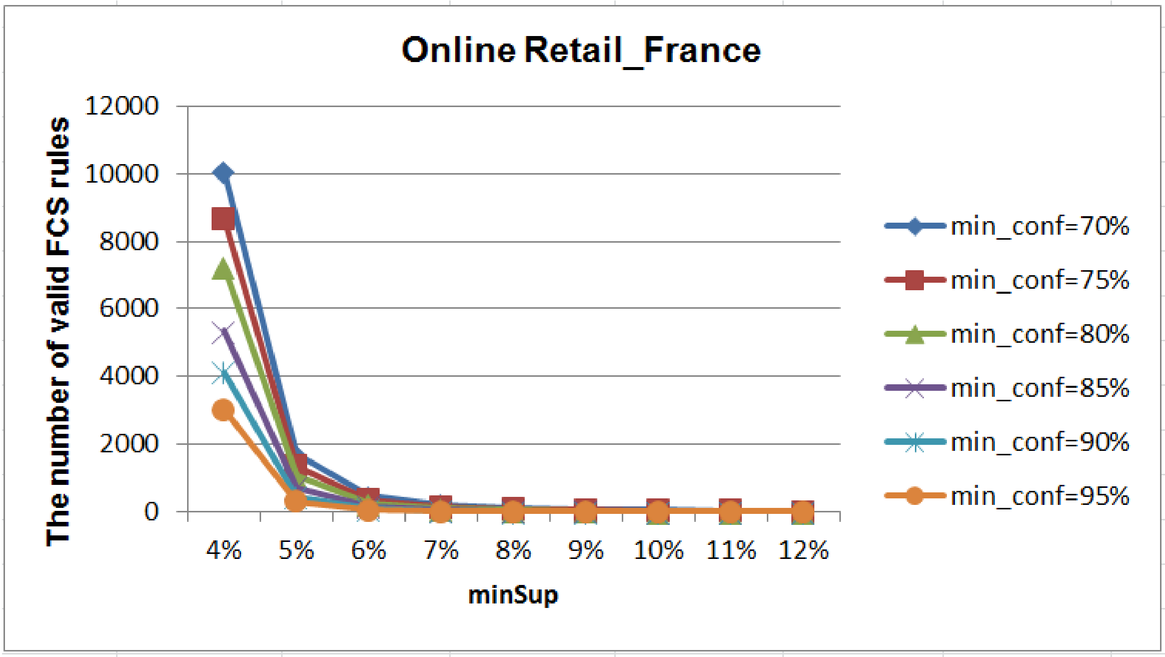

Figure 2. For the sequence database Online Retail_France

To match the input data of the algorithm, the Online Retail_France dataset is converted to the corresponding quantitative sequence database as follows:

- CusId: the ID of the customer
- *Time*: an integer represents the number of days of the Invoice Date differed from the first date of the data (01/02/2010). If it is the first day, Invoice Date will be the value 1;
- *StockCode*: the code of Stock
- *Quantity*: the number of StockCode in a transaction.

Each item in the datasets is partitioned into fuzzy sets and their membership functions defined by the formula (2.1) in Example 2.4 with $K = 3$ and $A_{3,1}^{x_m} = x_{m\_Small}$, $A_{3,2}^{x_m} = x_{m\_Average}$, $A_{3,3}^{x_m} = x_{m\_Large}$. The *ma*, *mi* values correspond to the largest, smallest values purchased of the *stockcode* x_m. The *Quantity* attribute is used to calculate the fuzzy values of the purchased *stockcodes*.

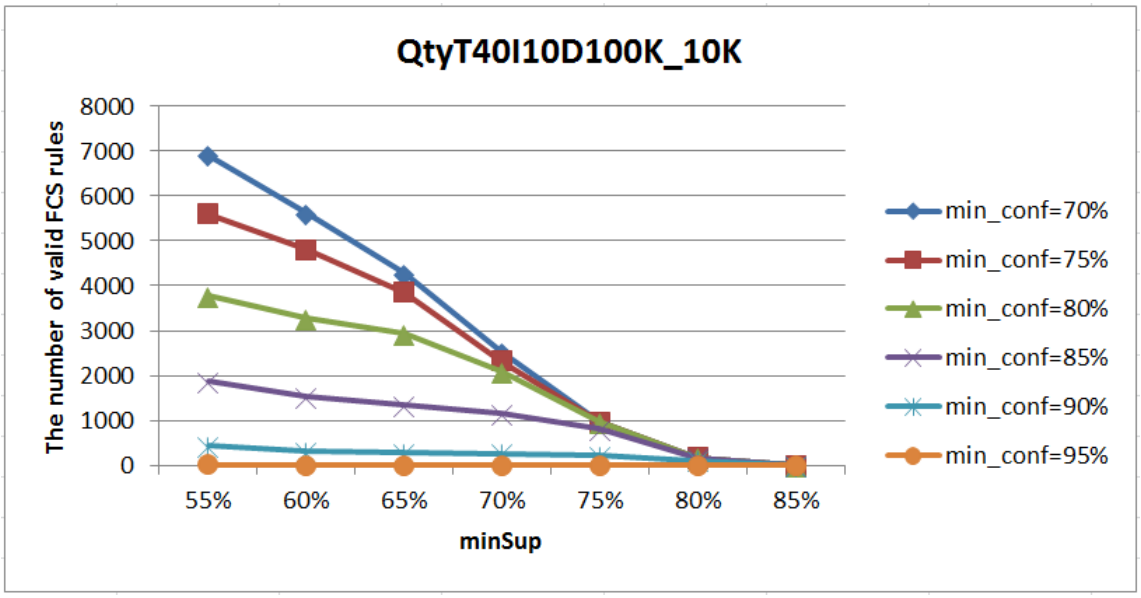

Figure 3. For the sequence database QtyT40I10D100K_10K

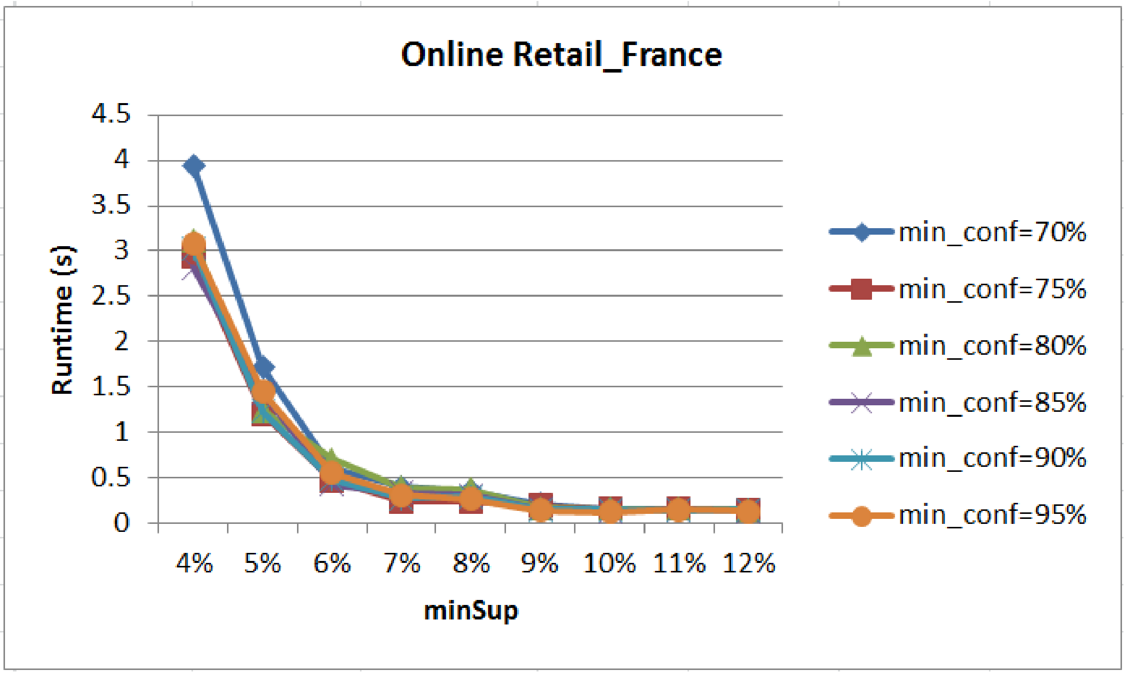

Figure 4. For the sequence database Online Retail_France

4.2. The results

4.2.1. Relationship between the number of valid FCS rules with *minSup* and *minConf*

The relationships between the number of valid FCS rules found out in the two Online Retail_France and QtyT40I10D100K_10K quantitative databases with *minSup* and *minConf*

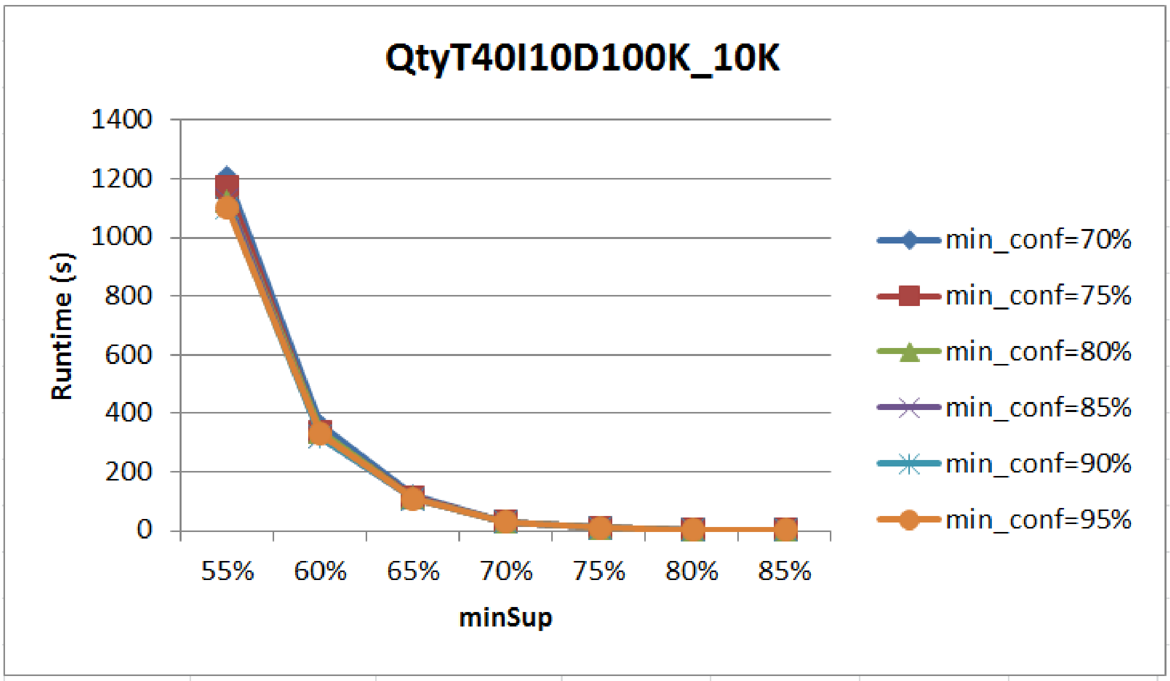

Figure 5. For the sequence database QtyT40I10D100K_10K

are shown in Figures 2, 3 above. These figures reveal that the number of valid FCS rules will decrease sharply if *minSup* and/or *minConf* is increased and the number of these rules will increase if at least one of the two minimum thresholds is decreased and in particular, the number of valid FCS rules will increase very rapidly if the *minSup* is lower than a certain level depending on a specific quantitative sequence database used to discover FCS rules.

4.2.2. Relationship between algorithm executing time with *minSup* and *minConf*

The algorithm executing time for mining valid FCS rules in the two quantitative sequence databases aforementioned depends on *minSup* and *minConf* as shown in Figures 4, 5. It can be seen that the relationship between algorithm executing time with *minSup* and *minConf* is quite similar to the relationship between the number of valid FCS rules with the *minSup* and *minConf* as mentioned above.

4.2.3. Analyzing the valid FCS rules

With *minSup* of 8% and *minConf* of 90%, using the FERMiner algorithm on the Online Retail_France data set, we obtained valid FCS rules described in Table 4.

Table 4. Valid FCS rules found out with *minSup* of 8% and *minConf* of 90%

Valid FCS rules	**The support**	**The confidence**	**The means of the rules**
$702_{\_Average}$, $1116_{\_Average}$ ==> $1545_{\_Small}$	8.85%	96.25%	If a customer buys the stockcode **702** with an ***Average*** number and the stockcode **1116** with an ***Average*** number, then he will also buy later the stockcode **1545** with a ***Small*** number with the support and the confidence of 8.85% and 96.25%, respectively
$1545_{\_Small}$, $110_{\_Large}$ ==> $1545_{\_Small}$	8.74%	96.20%	If a customer buys the stockcode **1545** with a ***Small*** number and the stockcode **110** with a ***Large*** number, then he will also buy later the stockcode **1545** with a ***Small*** number with the support and the confidence of 8.74% and 96.20%, respectively
$1194_{\_Large}$ ==> $1545_{\_Small}$	8.51%	92.50%	If a customer buys the stockcode **1194** with a ***Large*** number, then he will also buy later the stockcode **1545** with a ***Small*** number with the support and the confidence of 8.51% and 92.50%, respectively.
$1545_{\_Small}$, $1116_{\_Average}$ ==> $1545_{\_Small}$	10.00%	91.58%	If a customer buys the stockcode **1545** with a ***Small*** number and the stockcode **1116** with an ***Average*** number, then he will also buy later the stockcode **1545** with a ***Small*** number with the support and the confidence of 10.00% and 91.58%, respectively
$1545_{\_Small}$, $1269_{\_Large}$ ==> $1545_{\_Small}$	9.43%	91.11%	If a customer buys the stockcode **1545** with a ***Small*** number and the stockcode **1269** with a ***Large*** number, then he will also buy later the stockcode **1545** with a ***Small*** number with the support and the confidence of 9.43% and 91.11%, respectively
$110_{\_Large}$ ==> $1545_{\_Small}$	9.43%	91.11%	If a customer buys the stockcode **110** with a ***Large*** number, then he will also buy later the stockcode **1545** with a ***Small*** number with the support and the confidence of 9.43% and 91.11%, respectively

5. CONCLUSION AND RESEARCH IN FUTURE

The main contribution of this paper is to propose and solve the problem of mining FCS rules in quantitative sequence databases. By proposing some new concepts and developing relevant concepts in [9] for the case of fuzzy sequence databases, the algorithm FERMiner was developed based on the ERMiner algorithm in [9]. Experimental results of the FERMiner algorithm show that the number of valid FCS rules and the executing time of the algorithm depend strongly on the minimum support (*minSup*) and confidence (*minConf*) thresholds. This dependence is perfectly suited with theory and reality.

It can be seen that common sequential rules found out until now provide only information about an itemset occuring after an other itemset in a same order in transaction sequences, but in reality, people not only consider the information regarding the order of occurrences of two itemsets but also consider temporal range of their occurrences to use for forecasting goal. The algorithms mining the common sequential rules so far and in this paper are not able to find out rules providing such information. On the other hand at present, the algorithms mining common sequential rules so far as well as in this paper also only find out rules in which the antecedent and consequent parts are contained in a same transaction sequences performed by an object. The algorithms are not able to find common sequential rules in which the antecedent and consequent parts can be contained in sequences performed by different objects, as long as the time point of occurrence of the consequent part must be after the time point of occurrence of the antecedent part. Proposing some other algorithms to solve the two problems aforementioned is our research work in the future.

REFERENCES

[1] 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, pp. 207–216, 1993.

[2] R. Agrawal, R. Srikant, and others, "Mining sequential patterns," in *ICDE*, vol. 95, pp. 3–14, 1995.

[3] H. Cao, Y. Zhang, L. Jia, and G. Si, "A fuzzy sequential pattern mining algorithm based on independent pruning strategy for parameters optimization of ball mill pulverizing system ," *Inf. Technol. Control*, vol. 43, no. 3, pp. 303–314, 2014.

[4] R.-S. Chen, G.-H. Tzeng, C. C. Chen, and Y.-C. Hu, "Discovery of fuzzy sequential patterns for fuzzy partitions in quantitative attributes," in *Proceedings ACS/IEEE International Conference on Computer Systems and Applications*, 2001, pp. 144–150. Doi: 10.1109/AICCSA.2001.933967

[5] Ö. F. Çelebi, E. Zeydan, \.Ismail Ar\i, Ö. Ileri, and S. Ergüt, "Alarm sequence rule mining extended with a time confidence parameter," in *IEEE International Conference on Data Mining (ICDM)*, 2014. [Online]. Available from: http://hdl.handle.net/10679/1957

[6] C. Fiot, A. Laurent, and M. Teisseire, “From crispness to fuzziness: Three algorithms for soft sequential pattern mining,” *IEEE Trans. Fuzzy Syst.*, vol. 15, no. 6, pp. 1263–1277, 2007.

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

[8] P. Fournier-Viger, U. Faghihi, R. Nkambou, and E. M. Nguifo, “CMRules: Mining sequential rules common to several sequences,” *Knowledge-Based Syst.*, vol. 25, no. 1, pp. 63–76, 2012.

[9] P. Fournier-Viger, T. Gueniche, S. Zida, and V. S. Tseng, “ERMiner: sequential rule mining using equivalence classes,” in *International Symposium on Intelligent Data Analysis*, 2014, pp. 108–119.

[10] P. Fournier-Viger, R. Nkambou, and V. S.-M. Tseng, “RuleGrowth: mining sequential rules common to several sequences by pattern-growth,” in *Proceeding SAC ’11 Proceedings of the 2011 ACM Symposium on Applied Computing*, TaiChung, Taiwan, March 21–24, 2011. pp. 956–961. Doi¿10.1145/1982185.1982394

[11] J. Fowkes and C. Sutton, “A subsequence interleaving model for sequential pattern mining,” in *Proceeding KDD ’16 Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, California, USA, August 13–17, 2016. pp.835–844. Doi¿10.1145/2939672.2939787

[12] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.-C. Hsu, “FreeSpan: frequent pattern-projected sequential pattern mining,” in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2001),* Boston, MA, United States, Aug 20–23, 2000. pp. 355–359.

[13] J. Han, J. Pei, and Y. Yin, “Mining frequent patterns without candidate generation,” in *ACM Sigmod Record*, vol. 29, no. 2, pp. 1–12, 2000.

[14] M. Han, Z. Wang, and J. Yuan, “Mining constraint based sequential patterns and rules on restaurant recommendation system ,” *J. Comput. Inf. Syst.*, vol. 9, no. 10, pp. 3901–3908, 2013.

[15] T. Hong, C.-S. Kuo, and S.-C. Chi, “Mining fuzzy sequential patterns from quantitative data,” in *IEEE SMC’99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 99CH37028)*, Tokyo, Japan, Oct. 12–15, 1999. pp. 962–966. Doi: 10.1109/ICSMC.1999.823358

[16] Y.-C. Hu, R.-S. Chen, G.-H. Tzeng, and J.-H. Shieh, “A fuzzy data mining algorithm for finding sequential patterns,” *Int. J. Uncertainty, Fuzziness Knowledge-Based Syst.*, vol. 11, no. 02, pp. 173–193, 2003.

[17] T. C.-K. Huang, “Discovery of fuzzy quantitative sequential patterns with multiple minimum supports and adjustable membership functions,” *Inf. Sci. (Ny).*, vol. 222, pp. 126–146, 2013.

[18] T. Huang, R. Huang, B. Liu, and Y. Yan, “Extracting various types of informative web content via fuzzy sequential pattern mining ,” in *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint Conference on Web and Big Data*, 2017, pp. 230–238. Doi: https://doi.org/10.1007/978-3-319-63579-8_18

[19] T. Kieu, B. Vo, T. Le, Z.-H. Deng, and B. Le, “Mining top-k co-occurrence items with sequential pattern,” *Expert Syst. Appl.*, vol. 85, pp. 123–133, 2017.

[20] (2019, May.) UCI-Machine Learning Repository. [Online]. Available from: https://archive.ics.uci.edu/ml/datasets.php

Received on February 19, 2019
Revised on August 08, 2019