Available online at www.isr-publications.com/jmcs
J. Math. Computer Sci. 16 (2016), 112–130
Research Article

Online: ISSN 2008-949x

Journal of Mathematics and Computer Science

Journal Homepage: www.tjmcs.com - www.isr-publications.com/jmcs

Constraint fuzzy sequential pattern mining with TOPSIS method

F. Zabihi[a], M. M. Pedram[b,*], M. Ramezan[b]

[a]*Industrial Engineering Department, Kharazmi University, Tehran, Iran.*
[b]*Computer Engineering Department, Kharazmi University, Tehran, Iran.*

Abstract

To maintain profitability, many companies consider effective customer relationship management (CRM) to be one of the critical factors for success. The central objective of CRM is to maximize the lifetime value of a customer to a company and find positive customers. One of the methods that help this is sequential pattern mining. Sequential pattern mining is to discover all sub-sequences that are frequent. The classical sequential pattern mining algorithms do not allow processing of numerical data and require preprocessing of these data into a binary representation, which necessarily leads to a loss of information. Fuzzy sets are used to overcome this problem. In present fuzzy sequential pattern mining algorithms, there isn't any matter of itemset time and sequences are only found based on sequence of happening. In this paper, a novel algorithm about fuzzy sequential pattern mining is proposed with the time-gap constraints confine the time interval between two adjacent elements to a reasonable period while the sliding time window constraint permits elements of a pattern to span a set of transactions within a user-specified window and a fuzzy membership function is considered. Therefore, loss of useful sequences is prevented in the search process. The proposed algorithm searches for a goal sequence within the defined fuzzy sliding window and fuzzy gap functions.

Keywords: Fuzzy sequential pattern mining, constraint, fuzzy gap.

1. Introduction

It has been well recognized that mining in temporal databases is an important data-mining task. Many approaches to temporal data mining have been proposed to extract information, such as time

*Corresponding author
Email address: pedram@khu.ac.ir (M. M. Pedram)

Received 2015-09-15

series analysis [1, 9, 23, 27], temporal association rules mining [4, 24, 25] and sequential pattern discovery [1, 29, 30]. Among them, sequential pattern mining is one of the most well-known methods and has broad applications (see [16, 32, 41] for an overview), including web-log analysis, customer purchase behavior analysis, process analysis of scientific experiments, and medical record analysis [31, 33, 34]. Sequential pattern mining was proposed by Agrawal and Srikant [2] and it can be defined as "Given a set of sequences, where each sequence consists of a list of elements and each element consists of a set of items and given a user-specified min-support threshold, sequential pattern mining is to find all frequent subsequences, i.e., the subsequences whose occurrence frequency in the set of sequences is no less than min-support." Sequential pattern mining is similar to deriving association rules; the greatest difference between them is that a pattern in the association rule is an unordered set of items and sequential pattern mining discusses the relationship between items and the effect of temporal sequence in the transaction database. For mining sequential patterns, many efficient and useful approaches have been developed at a single level. For instance, Agrawal and Srikant [2] proposed the AprioriSome and AprioriAll algorithms to mine sequential patterns from a set of transactions, briefly; AprioriAll is a three-phase algorithm. First all itemsets with minimum support (frequent itemsets) are found, then it transforms the database so that each transaction is replaced by the set of all frequent itemsets contained in the transaction and finds sequential patterns. Pei et al. [28] proposed PrefixSpan, a pattern-growth method that employed the prefix-projection in sequential pattern mining. PrefixSpan is also capable of dealing very large database. To make the database for next pass much smaller and consequently make the algorithm speedier, PrefixSpan mainly employs the method of database projection; also in PrefixSpan there is no need for candidates' generation only recursively project the database according to their prefix.

A family of regular expression based algorithms that can satisfy user-specified regular expressions were presented by Garofalakis et al. [13] and [14]. A mining method, called SPADE [40], decomposed the original problem into smaller subproblems such that it can efficiently discover sequential patterns by using lattice search techniques and simple join operations. Also Han et al. [17] proposed an efficient algorithm to mine the desired number of frequent closed patterns.

When mining a large database, there can be many sequential patterns. More often a user may want some specific subset of sequential patterns instead of all patterns. The constrained sequential pattern mining techniques were used to solve this application requirement [12].

Although efficient algorithms have been proposed, mining a large amount of sequential patterns from large sequence databases is inherently a computationally expensive task. If we can focus on only those sequential patterns of interest to users, we may be able to avoid a lot of computation cost caused by those uninteresting patterns. This opens a new opportunity for performance improvement. For improving the efficiency of sequential pattern mining by focusing only on interesting patterns, constraints are necessary in many data mining applications. In the background of constraint-based sequential pattern mining, Srikant and Agrawal generalized the scope of sequential pattern mining to include time constraints, sliding time windows and user-defined taxonomy [29]. Mining frequent episodes can also be viewed as a constrained mining problem studied by Mannila et al. [26], since episodes are essentially constraints on events in the form of acyclic graphs in a sequence of events. Garofalakis et al. proposed regular expressions as constraints for sequential pattern mining and developed a family of SPIRIT algorithms; members in the family achieve various Degrees of constraint enforcement. The algorithms use relaxed constraints with nice properties (like anti-monotonicity) to filter out some unpromising patterns/candidates in their early stage [13]. Pei et al. proposed a systematic category of constraints and the pattern-growth methods to tackle the constraints [18]. Most of the conventional data mining algorithms for mining sequential patterns can identify the relation between transactions with binary values. However commonly temporal transactions with

quantitative values are seen in real-world applications. The fuzzy theory is a trend for dealing with the problem of sharp boundary. Hong et al. [19] put together the fuzzy set concepts and the AprioriAll algorithm. It first transforms quantitative values in transactions into linguistic terms, then filters them to find sequential patterns by modifying the AprioriAll mining algorithm. Thus, the next quantitative transactions of a customer can be predicted according to his/her previous transactions. Afterward, Hong et al. [7] extended the above fuzzy AprioriAll to able to handle multi-item transactions. However, the main inadequacy of fuzzy AprioriAll is that the way to determine the fuzzy membership function is not explained. The user has to determine them himself/ herself. Chen et al. [21] proposed a fuzzy grids-based sequential patterns mining algorithm to generate all fuzzy sequential patterns from relational database. In this method, each quantitative attribute is viewed as a linguistic variable and can be divided into many candidate 1-dim fuzzy grids. It consists of two phases: (1) generate all large fuzzy grids and (2) generate all fuzzy sequential patterns. Kaya and Alhaji [6] have proposed a novel multi-objective Genetic Algorithm (GA-based) optimization method for optimizing quantitative sequential patterns. The objective measures are support, confidence and a parameter related to the total number of fuzzy sets in the sequence. To discover fuzzy time-interval sequential patterns from databases Yen-Liang and Chen used the concept of fuzzy sets. Two efficient algorithms, the fuzzy time interval (FT-AprioriI) algorithm and the FTI-PrefixSpan algorithm, are developed for mining fuzzy time-interval sequential patterns. The main idea is to utilize the Apriori-like method to find the candidate sequential patterns. The large sequential patterns are found by checking support of each candidate sequential pattern. When the length of sequential patterns is larger than or equal to two, define the fuzzy number to each time. In this simulation results, the second algorithm outperforms the first one, not only in computing time but also in scalability with respect to various parameters [3]. Also Sandra Bringay used sequential patterns with fuzzy gaps to provide the biologists with more precise sequences [5]. The concept of gap used in her paper doesn't relate to time and it is related to the differences between genes.

An important contribution to support the fuzzy ranges of time interval of sequential patterns is the new sequential pattern mining algorithm with fuzzy time intervals (SPFTI) proposed by Chung-I Chang. This algorithm used hierarchical clustering technique and represents the time interval between itemsets as several trapezoid fuzzy numbers. This approach shows not only the order of successive itemsets, but also the fuzzy ranges of time intervals, which is a more flexible way than the FTI-Apriori algorithms. In addition, fuzzy sequential patterns can be used to mine additional knowledge so giving more information than sequential patterns [20]. A novel change mining model was proposed to detect the change in fuzzy time-interval sequential patterns called MineFuzzChange. brick-and-mortar transactional dataset collected from a retail chain in Taiwan and a B2C EC dataset, experiments are carried out to appraise the proposed model. Also the model helps managers to understand the changing behaviors of their customers and to formulate timely marketing and inventory strategies [15].

Guil et al. extended the algorithm introducing a fuzzy set-based technique into more expressive and flexible mining process. The proposed solution consists of the inclusion of a reference fuzzy set in the counting method. Moreover, the algorithm allows setting a user-defined parameter, which defines the granularity of the temporal dimension. They have carried out a series of experiments to show how TSET fuzzy-Miner algorithm behaves with both, synthetic and real-life datasets [22]. Lan et al. [8] introduced a new fuzzy utility measure with the fuzzy minimum operator to evaluate the fuzzy utilities of itemsets. Besides, an effective fuzzy utility upper-bound model based on the proposed measure was designed to provide the downward-closure property in fuzzy sets, thus reducing the search space of finding high fuzzy utility itemsets.

Cheng proposed a novel model to discover an FQSP with both multiple minimum supports and

adjustable membership functions. Experiments using synthetic and real datasets demonstrated the computational efficiency, scalability and effectiveness of the model [35].

Also in [36] was worked on fuzzy constrained sequential pattern mining. Six novel algorithms are proposed. In these algorithms in addition to fuzzifying quantity of items, the fuzzy membership functions are considered for sliding window constraint and time gap constraint. Therefore not only we will be able to process quantitative data but also prevent from deleting efficient sequences that occurred in crisp sequential pattern algorithms and generate relevant patterns based on the defined time constraints. One of the algorithms is proposed in [37] which a pattern will be found if the fuzzy membership of every item in itemsets is greater equal than zero and sliding window constraint was considered. Since the time span of buying is important for pattern, another algorithm would be proposed in [38] used time gap constraint in addition to sliding window constraint. After that in [39] more complete algorithm was proposed in which pattern would be found with the best fuzzy membership and searching is done for the entire database. Further in [10] was worked on one of a fuzzy sequential pattern algorithm to mine fuzzy sequential patterns from the Blood Transfusion Service Center data set.

As said, a lot of works have been done in sequential pattern mining and some other works consider constraints to users are allowed to specify their focus in the mining process through various time constraints and then relevant patterns are generated. Researchers have recognized that frequency is not the best measure to use in determining the significance of a pattern in many applications. Because of the importance of time constraint in sequential pattern mining approach and the capability of fuzzy sequential pattern in quantitative data, the combination of these two approaches are taken into account. A novel algorithm is proposed and described in Subsection 5.1. In this algorithm in addition to fuzzify quantity of items, a fuzzy membership function is considered for sliding window and time gap constraints. Also for showing the efficiency of the novel algorithm, two different algorithms are compared with it described in Subsection 6.1.

2. Sequential Pattern Mining

Definition 2.1. Sequential pattern is a sequence of itemsets that frequently occurred in a specific order and all items in the same itemsets are supposed to have the same transaction time value or within a time gap. Usually all the transactions of a customer are together viewed as a sequence, usually called customer-sequence, where each transaction is represented as an itemsets in that sequence and all the transactions are listed in a certain order with regard to the transaction-time.

Definition 2.2. Let D be a database of customer transactions, $I = I_1, I_2, \cdots, I_m$ be a set of distinct attributes called items, T be a transaction that includes customer-id, transaction-time, item-purchased, s_i be an itemset, which contains a set of items from I and S be a sequence that consists of an ordered list of item sets $< s_1, s_2, \cdots, s_n >$. No customer has more than one transaction with the same transaction-time.

Example 2.3. If a customer c_1 purchases products $1, 2, 3, 4$ and 5 called items according to the sequence $< (1)(2, 3)(4)(5) >$, then all items of the sequence were bought separately, except products 2 and 3 which were purchased at the same time. Also product 2 and 3 called itemsets. Suppose product 1 was bought in 20 Apr. and products 2 and 3 were bought in 16 May and respectively products 4 and 5 were bought in 14 Jul. and 15 Aug., a transaction T is shown in Table 1.

Table 1: An example of transaction

costumer	Transaction	product	Date
	1	1	20 Apr
	2	2, 3	16 May
C_1	3	4	14 Jul
	4	5	15 Aug

Definition 2.4. Contain: a sequence $< a_1, a_2, \cdots, a_n >$ is contained in another sequence $< b_1, b_2, \cdots, b_m >$, if $i_1 < i_2 < \cdots < i_n$ such that $a_1 \subseteq b_{i1}, a_2 \subseteq b_{i2}, \cdots, a_n \subseteq b_{in}$.

Example 2.5. The sequence $< (3)(6,7,9)(7,9) >$ is contained in $< (2)(3)(6,7,8,9)(7)(7,9) >$, since $(3) \subseteq (3), (6,7,9) \subseteq (6,7,8,9), (7,9) \subseteq (7,9)$. However, sequence $< (2)(3) >$ is not contained in sequence $< (2,3) >$, since the former sequence means 3 is bought after 2 being bought, while the latter represents *item* 2 and 3 being bought together. A sequence is maximal if it is not contained in any other sequences.

Definition 2.6. Support: a customer support a sequence s if s is contained in the corresponding customer-sequence, the support of sequence s is defined as the fraction of customers who support this sequence.

$$\text{Support(s)} = \frac{\text{Number of support customers}}{\text{Total number of customers}} \tag{2.1}$$

Sequential pattern mining is the process of extracting certain sequential patterns whose support exceeds a predefined minimal support threshold. Since the number of sequences can be very large and users have different interests and requirements, to get the most interesting sequential patterns, usually a minimum support is pre-defined by users. By using the minimum support we can prune out those sequential patterns of no interest, consequently make the mining process more efficient.

Note that *items* are processed using a binary evaluation – present or not present.

3. Constrained sequential pattern

In most of real world problems, especially pattern study for managerial decision support, it is important to impose time interval constraint in the sequential pattern mining task. Mining complete set of sequential patterns is not always effective or efficient. Efficiency and effectiveness can be improved by focusing only on interesting patterns. Also with pushing constraint in the algorithm, the user can easily retrieve their favorite patterns and the cost of mining patterns from database is reduced.

We now recall the definition of frequent sequences when handling time constraints [30].

Definition 3.1. given a user-specified time gap and a time window size (*window Size*), a data sequence $d = < d_1 \cdots d_m >$ is said to support a sequence $S =< s_1 \cdots s_n >$ if there exist integers $l_1 \leq u_1 < l_2 \leq u_2 < \cdots < l_n \leq u_n$ such that:

1. s_i is contained in $U_{k=l_i}^{u_i} d_k$, $1 \leq i \leq n$;

2. transaction-time (d_{u_i}) - transaction-time (d_{l_i}) $\leq$*windowsize*, $1 \leq i \leq n$;

3. transaction-time (d_{l_i}) - transaction-time $(d_{u_{i-1}})$ $>$*minGap*, $2 \leq i \leq n$;

4. transaction-time (d_{u_i}) - transaction-time $(d_{l_{i-1}})$ $\leq$*maxGap*, $2 \leq i \leq n$.

Example 3.2. Let us now consider the sequence $S =< (a, b, c, d)(e, f, g) >$ and time constraints specified as *windowSize*=3 and *minGap*=0 and *maxGap*=5, $l_1 = 1$, $u_1 = 2$, $l_2 = 3$ and $u_2 = 5$ (see Figure 1). Products a, b, c and d are bought in time span 1 to 2 and products e, f and g are bought in time span 3 to 5.

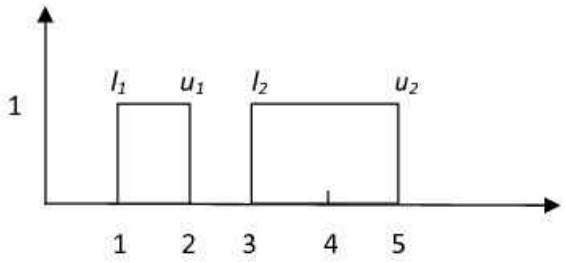

Figure 1: Illustration of the time constraints

The time-gap constraints confine the time interval between two adjacent elements to a reasonable period while the sliding time window constraint permits elements of a pattern to span a set of transactions within a user-specified window.

Mining sequences with time constraints allows a more flexible handling of the transactions, insofar as the end user is provided with the following advantages:

- To group together *itemsets* when their transaction times are rather close via the *window Size* constraint. For example, it does not matter if *itemsets* in a sequential pattern were present in two different transactions, as long as the transaction times of those transactions are within some small window.

- To regard *itemsets* as too close or distant to appear in the same frequent sequence with time gap constraint. For example, the end user probably does not care if a client bought 'Star Wars Episode IV' followed by 'Star Wars Episode III' two years later. [30]

4. Fuzzy Sequential Patterns

In order to mine fuzzy sequential patterns, the universe of each quantitative *item* is partitioned into several fuzzy sets.

Definition 4.1. A fuzzy *item* is the association of one *item* and one corresponding fuzzy set. It is denoted by $[x, a]$ where x is the *item* (also called attribute) and a is the associated fuzzy set.

Example 4.2. [milk, lot] is a fuzzy *item* where lot is a fuzzy set defined by a membership function on the quantity universe of the possible values of the *item* milk.

Definition 4.3. A fuzzy *itemset* is a set of fuzzy *items*. It can be denoted as a pair of sets (set of *items*, set of fuzzy sets associated to each *item*) or as a list of fuzzy *items*.

We will note: $(X, A) = ([x_1, a_1], \cdots, [x_p, a_p])$ where X is a set of *items*, A a set of corresponding fuzzy sets and $[x_i, a_i]$ are fuzzy *items*.

Example 4.4. (X, A) = ([milk, lot][butter, little]) is a fuzzy *itemset* and can be also denoted by ((milk, lot)(butter, little)).
One fuzzy *itemset* contains only one fuzzy *item* related to one single attribute. For example, the fuzzy *itemset*([milk, lot][milk,little]) is not a valid fuzzy *itemset* because it contains twice the attribute milk.

Definition 4.5. A g-k-sequence $S =< s_1 \cdots s_g >$ is a sequence constituted by g fuzzy *itemsets* $s = (X, A)$ grouping together k fuzzy *items* $[x, a]$.

Example 4.6. The sequence < ([milk, little][butter, lot]) ([cheese,lot]) > groups together 3 fuzzy *items* into 2 *itemsets*. It is a fuzzy $2-3-$sequence.

Definition 4.7. The frequency of a fuzzy *itemset* is computed as the number of objects supporting this fuzzy *itemset* compared to the total number of objects in the database:

$$Freq(X, A) = \frac{\sum_{o \in O}[F(o, (X, A))]}{|O|}, \tag{4.1}$$

where the frequency *Deg*ree *Freq(X, A)* shows whether the object o supports the fuzzy *itemset* (X,A) or not. O is representation of entire objects in database [11].

5. The proposed constrained fuzzy sequential pattern

The combination of time constraint in sequential pattern mining approach and the capability of fuzzy sequential pattern in quantitative data are taken into account. To considering time of accruing *items*, time sliding window and time gap constraints are entered in the proposed algorithm. Mining sequence are efficiently with pushing time sliding window and time gap constraints. In proposed algorithm a fuzzy function is considered for time constraints. The time is spanned by fuzzifying and the best membership *Deg*ree of *item* is searched by the algorithm.

Based on this, a novel algorithm is proposed that is described in Subsection 5.1. With fuzzy membership function for time constraints not only we will be able to process quantity data but also prevent from deleting efficient sequences and generate relevant patterns based on the defined time constraint.

Definition 5.1. The *item* and *itemset* concepts have been redefined relative to classical sequential patterns.

Itemset: one or more *items* with the same time stamp

curIS: current *Itemset* which is processed

curItem: current *Item* of *Itemset*

gS: (goal sequence) sequence contains one or more *Itemset*

T: fuzzified transactions set

FSW: Fuzzy Sliding Window

Win: sliding window membership function

wS: sliding window span

FG: fuzzy time gap

gap: time gap membership function

found: membership *Deg*rees of goal sequence

5.1. The proposed constrained fuzzy sequential pattern mining algorithm

First, the values of transactions are fuzzified by the membership functions defined by user. This process will be done for each quantity of *items* in transaction set. Then various sequences are generated and searched in each transaction. The proposed algorithm searches goal sequence based on fuzzy sliding window and fuzzy time gap shown in Figure 2 and the best sequence must be found based on three indexes: membership *Deg*ree of *item*, membership *Deg*ree of sliding window and membership *Deg*ree of time gap. But choosing best sequence with three indexes is not easy. So, for choosing best solution, one method of Multiple Attribute Decision Making (MADM) called *TOPSIS* [1] is used. Algorithm searches all customers and average of fuzzy membership *Deg*rees of each customer is computed, then sequence support is achieved.

The novel proposed algorithm is made from three smaller algorithms called *FindConstTotallySeq* algorithm, *FindTotallyIS* algorithm and Optimization algorithm. Also TOSIS method is used in *FindTotallyIS* algorithm. For better description, this paper is structured as follows: Section 5.1.1 presents an introduction to *Find Const Totally Seq* algorithm, *FindTotallyIS* algorithm is described in Section 5.1.2, while Section 5.1.3 reviews *TOPSIS* method and *Optimization* algorithm is described in Section 5.1.4.

5.1.1. Find Const Totally Seq algorithm

The algorithm gets goal sequence as input and searches it in a fuzzy transaction set and produces various *path*s, then with *FindConstTotallyIS* algorithm and pushing fuzzy sliding window constraint and time gap constraint, useful patterns are retrieved. Inefficient *path*s would be deleted by ***Optimization*** algorithm. Finally the best goal sequence (based on the fuzzy membership *Deg*ree) in the transaction is found and the fuzzy membership *Deg*ree is retrieved as output.

Input: *gS*, candidate *g-k*-sequence;
R, record set to run;

Output: *m*, frequency *Deg*ree of the best *gS* representation instantiated in the record set *R*;

Paths: list of *path*s → (*seq, curIS, curDeg*);
ND, list of found;
*Items*et(s) *memDeg*, *fwinDeg, gapDeg*;
W, attribute weight for *TOPSIS*;

Paths← *Path*(∅, *gS*.first, 0);
For each $r \in R$ **do**
  For each *pth* ∈ *Paths, not updated at r* **do**
    If (*pth* not closed) **Then**
      cIS ← *FindTotallyIS* (*pth.preIS*, *gS*);
      If (not empty *cIS*) **Then**
        pth.curDeg← *pth.curDeg* $T_{[x,a]} \in$ *pth*.curIS $\alpha_a(r[x])$;
        Update (*pth*);
      End If
    End If
    For *j* from 2 to *pth. curIS*– 1**do**
    cIS ← *FindTotallyIS* (*gS*.get(*j*));
    If (not empty*cIS*) **Then**
      npth.curIS ← gS.get(*j*);
      ND[*1*] = [*pth.curIS.memDeg*, *pth.curIS.FSWDge*, *pth curIS.FGDeg*];
      ND[*2*] = cIS.*memDeg, cIS.FSWDge, cIS.FGDeg*;
      If(*TOPSIS*(*ND*, *W*) = 2 **Then**
        For *I* **from** 1 **to** *j*-1 **do**
          npth.seq ← *npth.seq gS*.get(*i*);
          npth.curDeg← *npth.curDeg pth.curDeg*;
        End For
        npth.curDeg ← *npth.curDeg* $T_{[x,a]}$ (*gS*.get(*j*)($_a$(*r*[*x*]);
        Paths ← *Paths* ∪ *Update(npth)*;
      End If
    End If
  End For
End For
If ((*gS.first*∈ *r*)&(not(*FirstPass*)) **Then**
  pth← *Path*(∅, *gS*.first, $T_{[x,a]}$ (*pth*.first ($_a$(*r*[*x*]);
  Paths← *Paths*∪ *pth*;
  Update(*pth*);
End If
Paths. Optimize();
End For
For each *pathpthPath s* **do**
  If (*pth* not closed) **Then**
    delete(*pth*);
  End If
End For
m← *Agg* (*pth.curDeg*);
return *m*;

5.1.2. Find Totallyis algorithm

This algorithm gives an *items*et as input and searches each its *items* in the given fuzzified sliding window. Because searching is in several transactions more than one *item* is retrieved, so more than one *items*et as the same as goal *items*et present in sliding window. We consider three fuzzy membership *Deg*rees for each *items*et: first is the fuzzy membership *Deg*ree of *items* that is minimum fuzzy membership *Deg*ree of *items* in *items*et and second is the fuzzy membership *Deg*ree of transaction in *FSW* that is minimum fuzzy membership *Deg*ree of transactions which *items* of *items*et is found in it and finally fuzzy membership *Deg*ree of time gap that is difference between time of current *items*et and time of last *items*et in gap membership function. Therefore all of the *items*et should be considered and best *item* should be chose based on these three parameters. Therefore we confront MADM. In this algorithm one of the methods of MADM called *TOPSIS* is used. With computation of closeness of each *items*ets to ideal solution can get better *items*et based on defined indexes and this closeness will be returned.

Furthermore if fuzzy membership of found *items*et in time gap membership function is not greater than zero because of long time duration with last *items*et, current *items*et could not be as result and is deleted. If there is not any *items*et, the *path* will not be continued.

Input: *gS*, candidate *g-k*-sequence;
curIS, current *Item* Set
R, record set to run

Output: selectedIS, selected *Item* Set

FIS, list of *Items*et and its *Deg*rees
ND, list of found *Items*et(s) *memDeg, fwinDeg, gapDeg*
W, attribute weight for *TOPSIS*

$FIS \leftarrow \{fIS \mid (fIS \in FSW, fIS \in FG, fIS.memDeg > \omega)\}$; i=1;
Foreach $fIS \in FIS$ **do**
$ND[i] = [fIS.memDeg, fIS.FSWDge, fIS.FGDeg]$;
i ++;
End For
SelectedIS =FIS $Topsis(ND, W)$];
Return selected *IS*;

5.1.3. TOPSIS Method

TOPSIS *(Technique for Order Preference by Similarity to Ideal Solution) is one of the useful Multi Attribute Decision making techniques that are very simple and easy to implement, so that it is used when the user prefers a simpler weighting approach. The method is calculated as follows:*

Input: *ND*, list of found *Items*et(s) *memDeg*, *fwinDeg*, *gapDeg*;
W, attribute weight for *TOPSIS*;

Output: *top*, best;

$WNN \leftarrow$ diagonal of W;

$$V = N_D.W_{n\times n} = \begin{vmatrix} V_{11} & \cdots & V_{1j} & \cdots & V_{1n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ V_{m1} & \cdots & V_{mj} & \cdots & V_{mn} \end{vmatrix},$$

$$A^+ = \{(\max_i V_{ij} | j \in J), (\min_i V_{ij} | j \in J^{'}) | i = 1, 2, , m\} = \{V_1^+, V_2^+, , V_j^+, , V_n^+, \},$$

$$A^- = \{\left(\min_i V_{ij} j \in J\right), (\max_i V_{ij} | j \in J^{'}), | i = 1, 2, , m\} = \{V_1^-, V_2^-, , V_j^-, , V_n^-, \},$$

$$d_{i+} = \{ \sum_{j=1}^{n} \left(V_{ij} - V_j^+\right)^2 \}^{0.5} ; i = 1, 2, , m,$$

$$d_{i-} = \{ \sum_{j=1}^{n} \left(V_{ij} - V_j^-\right)^2 \}^{0.5} ; i = 1, 2, , m,$$

$$cl_{i+} = \frac{d_{i-}}{(d_{i+} + d_{i-})} \; ; \quad 0 \leq cl_{i+} \leq 1 \; ; \quad i = 1, 2, , m,$$

$max = \mathrm{cl}_1$;
$top = 1$;
For i=1 to m **do**
  If (cl_i>max) **Then**
    $max = cl_i$;
    $top = i$;
  End If
End For
Return *top*;

5.1.4. Optimization algorithm

This algorithm gives *path*s as input and search. Then every *path* which their *curIS* is equal, with *TOPSIS* method and comparison of current membership *Deg*ree, the *path* which its closeness to ideal solution is more is kept and delete other *path*s.

Input: *Paths*, list of *paths* to reduce

For each *pth*∈*Paths* **do**
  While (*Paths*.hasnext()) **do**
    pth'←*Paths*.next();
    If (*pth'.curIS*=*pth.curIS*) **Then**
      ND[1] = [*pth.memDeg, pth.FWINDge,pth.FGapDeg*];
      ND[2] = [*pth'.memDeg, pth'.FWINDge, pth'.FGapDeg*];
      top= *TOPSIS*(*ND*,*W*);
      If (*top* = 1) **Then**
        delete(*pth'*);
      Else
        delete(*pth*);
      End If
    End If
  End While
End For

Example 5.2. An example is considered in this section to understand how the proposed algorithm worked. Consider the dataset given by Table 2. Fuzzified data are shown in Table 3 based on given membership function for the *items* in Fig. 1. The proposed algorithm has been run for the given data, as well as the sliding window and time gap membership functions shown in Fig. 2. The search approach based on proposed algorithm is given in Table 4.

Table 2: data related to transactions of customers

Customer	Transaction	Milk	Butter	Cheese	Pencil	Eraser	Date
C_1	1	0	3	2	4	2	2
	2	1	2	2	2	1	16
	3	2	0	1	0	1	22
C_2	1	3	1	1	2	4	1
	2	5	3	4	3	2	4
	3	1	1	5	4	2	19
	4	4	4	1	3	4	23
	5	1	2	3	1	2	25
C_2	1	1	1	0	2	1	4
	2	4	0	1	0	0	11
	3	0	0	3	0	2	18
	4	2	1	4	1	0	22
	5	2	0	2	3	1	28

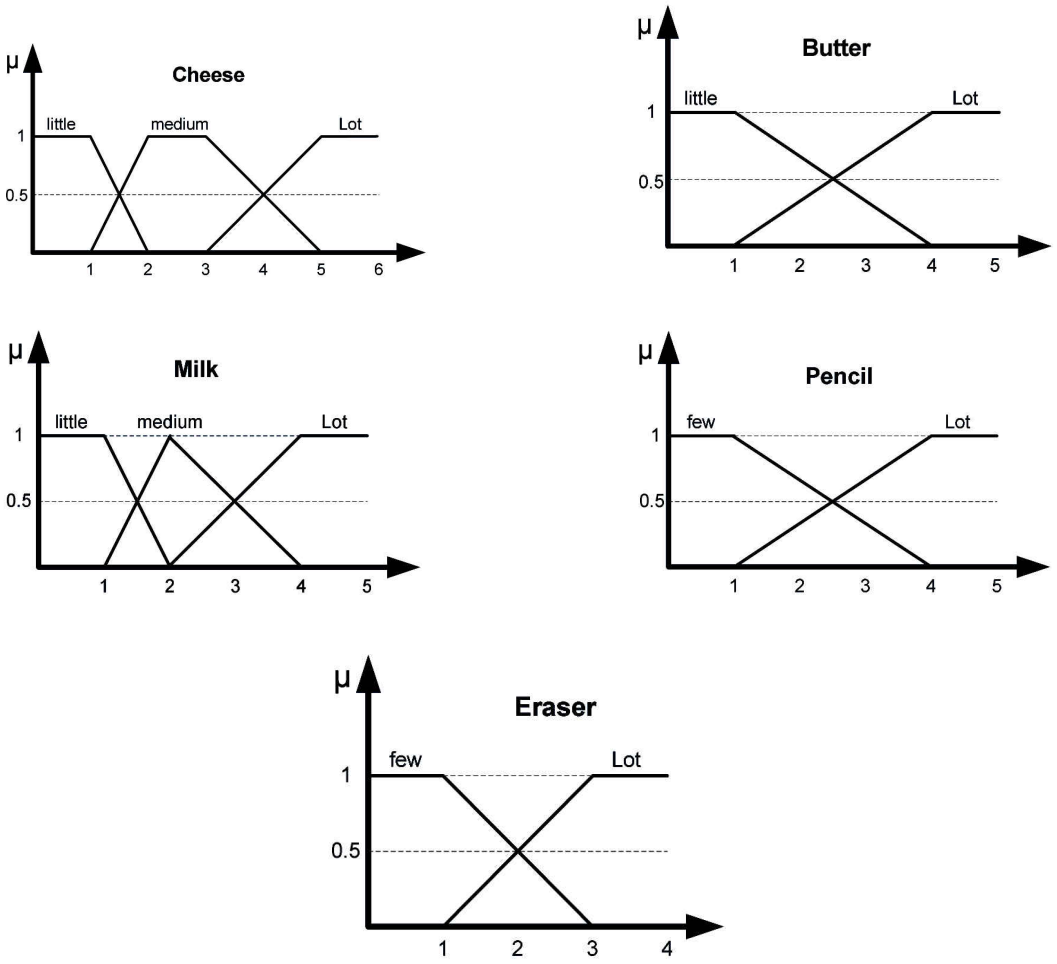

Figure 2: membership functions of *items*

Table 3: fuzzified data related to transactions of customer C2

	Milk			Butter		Cheese			Pencil		Eraser		
Transaction	li	m	L	li	L	li	m	L	F	L	f	L	Date
1	0	0.5	0.5	1	0	1	0	0	0.667	0.333	0	1	1
2	0	0	1	0.333	0667	0	0.5	0.5	0.333	0.667	0.5	0.5	4
3	1	0	0	1	0	0	0	1	0	1	0.5	0.5	19
4	0	0	1	0	1	1	0	0	0.333	0.667	0	1	23
5	1	0	0	0.667	0.333	0	1	0	1	0	0.5	0.5	25

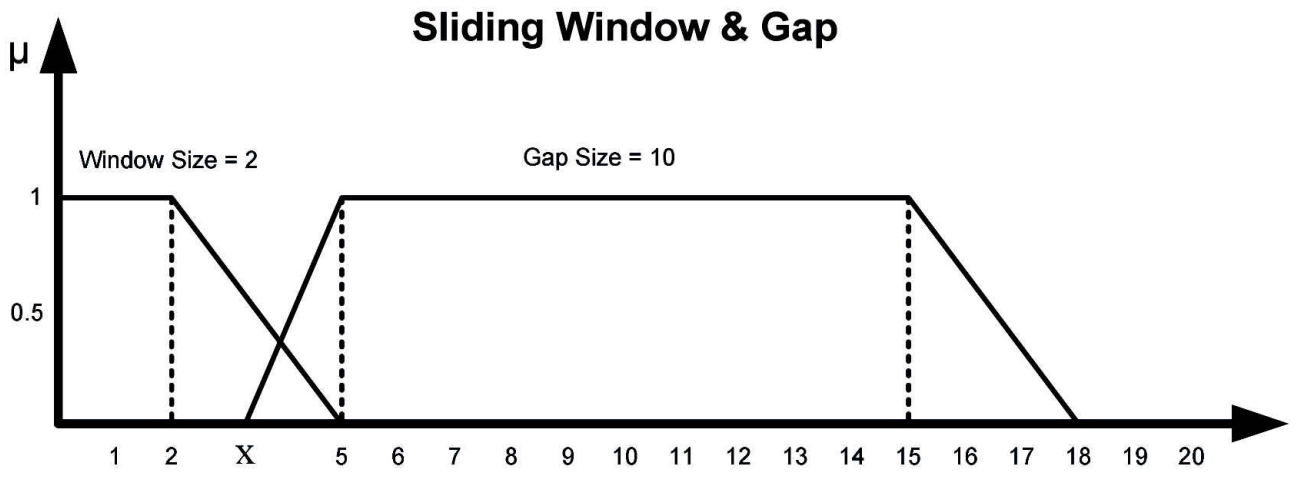

Figure 3: membership function related to sliding window and time gap

Sliding window span is given as 2 and gap length is given as 10 and their fuzzy membership functions are considered as shown in Figure 3.

Assumption: $\omega = 0.20$, gS=< ([Milk Lot],[Butter little])([Milk little])>, *path1*:(∅ , < ([Milk Lot], [Butter little])>, 0, 0, 0), Attributes = {*item* membership *Deg*ree in transaction, transaction membership *Deg*ree in *FSW*, transaction membership *Deg*ree in *FG* }, $W = \{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\}$

TOPSIS: As it is seen four *items*ets are found in current step of sliding window in Table 3. For achieving best *items*et, TOPSIS method is done. *items*ets are ordered with computed $cl_{i+}$and the result is:< ([T2: Milk Lot], [T1: Butter little])>, *itemDeg*: 1,*winDeg*: 0.667, *gapDeg*: —. This solution is chosen as best one and is added to *path*. The chosen *items*et is shown bold at each step. For illustration the relation of each product to transactions, T symbol is used. For instance [T2: Milk Lot] means Milk is bought in transaction 2.

Output: Chosen *path*: *Path*3 :(<([T4: Milk Lot], [T3: Butter little]), ([T5: Milk little])>,∅, 1, 0.667, 1), Output of *FindTotallySeq* algorithm: (1, 0.667, 1).

This algorithm is run for the entire customers in the database. Then the support of goal sequence will be computed by Equation (4.1).

Date	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	Iteration 1
Transactions	T_1			T_2															T_3				T_4		T_5	
$T_i \in FSW$	T_1, T_2																									
Found *itemsets*	< ([T1: Milk Lot], [T1: Butter little])>, *itemDeg*:0.5, *winDeg*:1, *gapDeg*:--- < ([T1: Milk Lot], [T2: Butter little])>, *itemDeg*:0.333, *winDeg*:0.667, *gapDeg*:--- **< ([T2: Milk Lot], [T1: Butter little])>, *itemDeg*:1, *winDeg*:0.667, *gapDeg*:---** < ([T1: Milk Lot], [T2: Butter little])>, *itemDeg*:0.333, *winDeg*:0.667, *gapDeg*:---																									
Paths	*Path*1 :(<([T2: Milk Lot], [T1: Butter little])>, < ([Milk little])>, 1, 0.667, ---)																									
Events	*Path*1: updated																									
Date	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	Iteration 2
Transactions	T_1			T_2															T_3				T_4		T_5	
$T_i \in FSW$	T_2																									
Found *itemsets*	**< ([T2: Milk Lot], [T2: Butter little])>, *itemDeg*:0.333, *winDeg*:1, *gapDeg*:---**																									
Paths	*Path*1 :(< ([T2: Milk Lot], [T1: Butter little])>, < ([Milk little])>, 1, 0.667, ---) *Path*2 :(<([T2: Milk Lot], [T2: Butter little])>, < ([Milk little])>, 0.333, 1, ---)																									
Events	*Path*1:not changed, *Path*2: created																									
Date	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	Iteration 3
Transactions	T_1			T_2															T_3				T_4		T_5	
$T_i \in FSW$	T_3,T_4																									
Found *itemsets*	For updating *Path*2: **< ([T3: Milk little])>, *itemDeg*:1, *winDeg*:1, *gapDeg*: 0.333** For creating *Path*3: **<([T4: Milk Lot], [T3: Butter little])>, *itemDeg*:1, *winDeg*: 0.333, *gapDeg*:---**																									
Paths	*Path*1 :(<([T2: Milk Lot], [T1: Butter little])>, < ([Milk little])>, 1, 0.667, ---) *Path*2 :(<([T2: Milk Lot], [T2: Butter little]), ([T3:Milk little])>, ∅, 0.667, 1, 0.333) *Path*3 :(<([T4: Milk Lot], [T3: Butter little])>, < ([Milk little])>, 1, 0.333, ---)																									
Events	*Path*1: deleted, *Path*2: updated, *Path*3:created																									
Date	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	Iteration 4
Transactions	T_1			T_2															T_3				T_4		T_5	
$T_i \in FSW$	T_4,T_5																									
Found *itemsets*	**<([T4: Milk Lot], [T5: Butter little])>, *itemDeg*:0.667, *winDeg*:1, *gapDeg*:---**																									
Paths	*Path*2 :(<([T2: Milk Lot], [T2: Butter little]), ([T3:Milk little])>, ∅, 0.667, 1, 0.333) *Path*3 :(<([T4: Milk Lot], [T3: Butter little])>, < ([Milk little])>, 1, 0.333, ---) *Path*4 :(<([T4: Milk Lot], [T5: Butter little])>, < ([Milk little])>, 0.667, 1, ---)																									
Events	*Path*2: not changed, *Path*3: not changed, *Path*4:created																									
Date	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	Iteration 5
Transactions	T_1			T_2															T_3				T_4		T_5	
$T_i \in FSW$	T_5																									
Found *itemsets*	**< ([T5: Milk little])>, *itemDeg*:1, *winDeg*:1, *gapDeg*: 1**																									
Paths	*Path*2 :(<([T2: Milk Lot], [T2: Butter little]), ([T3:Milk little])>, ∅, 0.667, 1, 0.333) *Path*3 :(<([T4: Milk Lot], [T3: Butter little]), ([T5:Milk little])>, ∅, 1, 0.667, 1) *Path*4 :(< ([T4: Milk Lot], [T5: Butter little])>, < ([Milk little])>, 0.667, 1, ---)																									
Events	*Path*2: not changed, *Path*3: updated, *Path*4: not changed																									
Date	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	Iteration 6
Transactions	T_1			T_2															T_3				T_4		T_5	
$T_i \in FSW$	∅																									
Found *itemsets*	∅																									
Paths	*Path*2 :(<([T2: Milk Lot], [T1: Butter little]), ([T3:Milk little])>, ∅, 0.667, 1, 0.333) ***Path*3 :(<([T4: Milk Lot], [T3: Butter little]), ([T5:Milk little])>, ∅, 1, 0.667, 1)** *Path*4 :(< ([T4: Milk Lot], [T5: Butter little])>, < ([Milk little])>, 0.667, 1, ---)																									
Events	*Path*3: selected																									

Table 4: search approach of goal sequence based on proposed algorithm

6. Experimental result

6.1. Experiments on a real dataset

For experiments, 452 transactions of 45 customers are considered. The novel algorithm is compared to two different algorithms. *TotallyFuzzy* [11] and *Totally Fuzzy-FSW* [38] are used for comparison. The number of frequent sequences is computed for minSupp{0.6, 0.7, 0.8, 0.9, 1.0} according to the primary condition given in Table 5. It is expected that more increase in minSupp, more decrease in the number of frequent sequences, as shown in Figure3.

Table 5: assumption data for running the algorithm

algorithm	ω	Sliding Window Size	Sliding Window fuzzy part Size	Gap Size	Gap fuzzy part Size
TotallyFuzzy	0.5	-	-	-	-
TotallyFuzzy-FSW	0.5	2	3	-	-
TotallyFuzzy-FSW-FG	0.5	2	3	10	3

As expected, the number of frequent sequences of all the algorithms decreases when the support is increased. Also it shows that TotallyFuzzy definitely extracts less frequent sequences than TotallyFuzzy-FSW. This is due to the fuzzy sliding window constraint. In fact, the proposed fuzzy algorithm, finds more relevant sequences. So the number of frequent sequences is necessarily increased compared to Totally Fuzzy. More sequences can be recovered by Totally Fuzzy-FSW-FG.

The runtime of the algorithms is shown in Figure 4. As expected, the runtime of mining the frequent sequences for all the algorithms decrease when the support is increased. Also it shows that runtime of Totally Fuzzy definitely is less than Totally Fuzzy-FSW. This is due to the fuzzy sliding window constraint. In fact, Totally Fuzzy-FSW, finds more relevant sequences, while its runtime increases. Obviously the number of frequent sequences and runtime will be increased by adding Gap constraint, thus runtime of Totally Fuzzy-FSW-FG is more than TotallyFuzzy-FSW. Therefore the penalty of achieving more relevant patterns is paying time.

For Figures 5 and 6, Table 6 is considered and TotallyFuzzy-FSW-FG algorithm has been run.

Table 6: Values of parameters for primary setting

ω	0.5
minSupp	0.8
SlidingWindow fuzzy part	3
Gap	10
Gap fuzzy part	3

Figure 5 shows the number of frequent sequences for Totally Fuzzy-FSW-FG algorithm and in Figure 6, runtime of frequent sequences according to the length of Sliding Window is considered. As expected, the runtime and the number of frequent sequences of the algorithm increase when the length of sliding window is increased. But depend on order of transactions in database; complete ascendant manner may not be achieved.

TotallyFuzzy-FSW-FG has been run with information of Table 5 instead of consideration sliding window=2 and the result is shown in Figure 7 and Figure 8. In these diagrams the different number of frequent sequences and runtime are shown when the length of fuzzy part of sliding window is increased. As expected, the number of frequent sequences and runtime increases when the length of fuzzy part of sliding window is increased. But depend on order of transactions in database; complete ascendant manner may not be achieved.

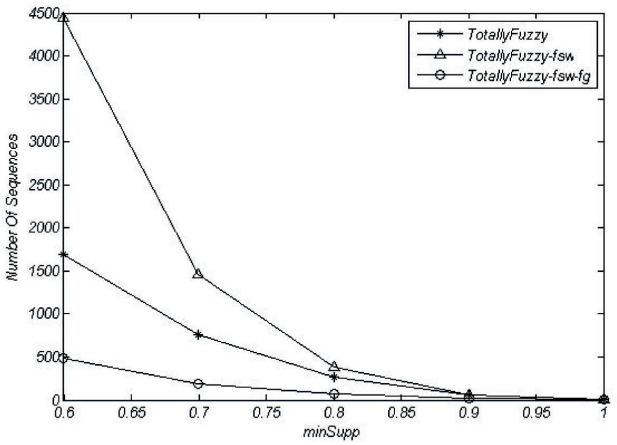

Figure 4: Number of frequent sequences according to minSupp

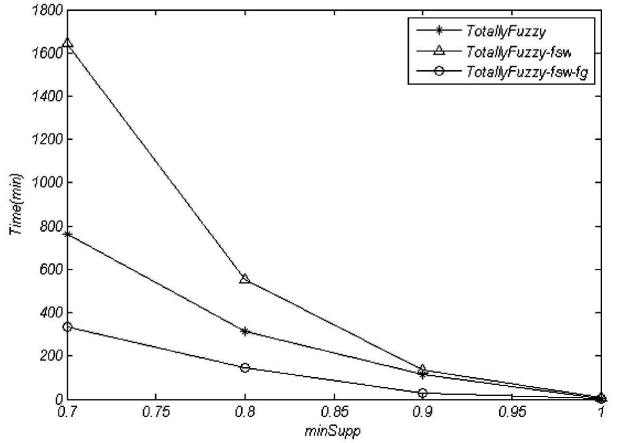

Figure 5: Runtime according to minSupp

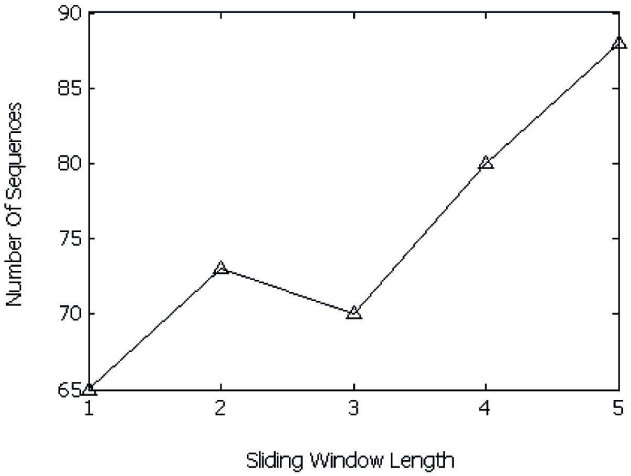

Figure 6: Number of frequent sequences according to the length of Sliding Window

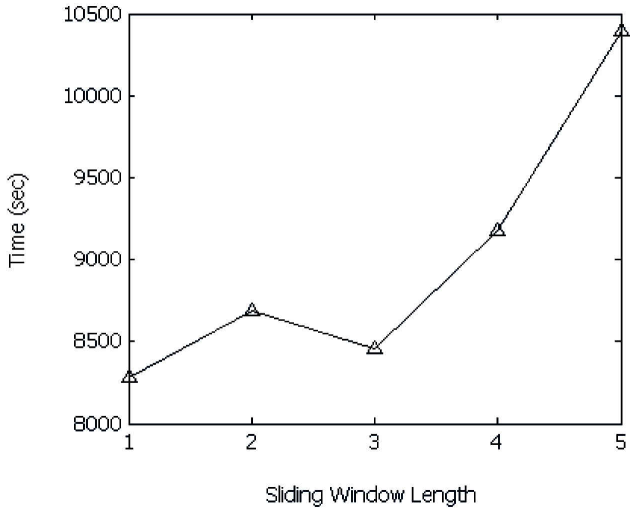

Figure 7: Runtime according to the length of Sliding Window

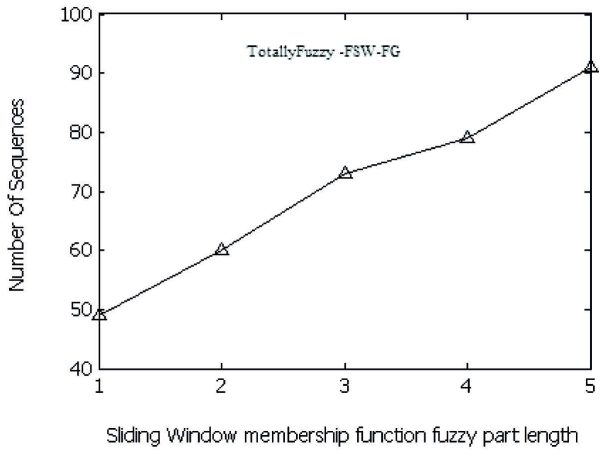

Figure 8: Number of frequent sequences according to the length of fuzzy part of Sliding Window

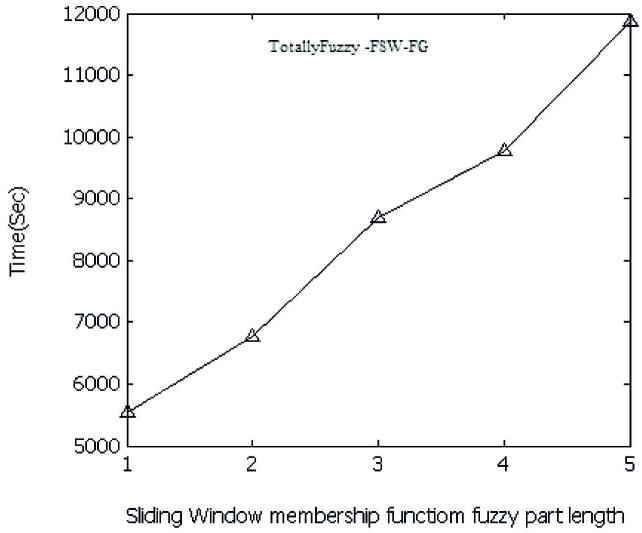

Figure 9: Runtime according to the length of Sliding Window

7. Conclusions

In this paper we proposed constrained fuzzy sequential pattern mining. In the proposed algorithm, sliding window constraint and time gap constraints are applied and a simple fuzzy version of these constraints are used and it was shown effectiveness of the proposed method which prevented from deleting efficient sequences. The algorithm takes into account event time and optimizes search process with fuzzification. Thus patterns which match the defined time constraints will be retrieved. The proposed algorithm works well in data mining for quantitative values; it is promising to be able to include other constraint retrieve useful patterns.

References

[1] R. Agrawal, C. Faloutsos, A. Swami, *Efficient similarity search in sequence databases*, Lecture Notes in Computer Science, **730** (1993), 69–84. 1, 5.1

[2] R. Agrawal, R. Srikant, *Mining sequential patterns*, Proceedings of 1995 International Conference Data Engineering (ICDE'95), (1995), 3–14. 1

[3] S. Bringay, A. Laurent, B. Orsetti, P. Salle, M. Teisseire, *Handling fuzzy gaps in sequential patterns: Application to health*, in FUZZ-IEEE, Korea, (2009). 1

[4] C. Y. Chang, M. S. Chen, C. H. Lee, *Mining general temporal association rules for items with different exhibition periods*, IEEE International Conference on Data Mining, Maebashi, (2002). 1

[5] C. I. Chang, H. Chueh, N. P. Lin, *Sequential patterns mining with fuzzy time-intervals*, in IEEE Sixth International Conference on Fuzzy Systems and Knowledge Discovery, (2009). 1

[6] Y. L. Chen, T. C. K. Huang, *Discovering fuzzy time-interval sequential patterns in sequence databases*, IEEE transactions on systems, man and cybernetics. Part B, Cybernetics, **35** (2005), 959–972. 1

[7] R. S. Chen, G. H. Tzeng, C. C. Chen, Y. C. Hu, *Discovery of fuzzy sequential patterns for fuzzy partitions in quantitative attributes*, Computer Systems and Applications ACS/IEEE International Conference, (2001), 144–150. 1

[8] K. Cheng, T. Huang, *Iscovery of fuzzy quantitative sequential patterns with multiple minimum supports and adjustable membership functions*, Inform. Sci., **222** (2013), 126–146. 1

[9] C. Faloutsos, M. Ranganathan, Y. Manolopoulos, *Fast subsequence matching in time-series databases*, Proceedings of the ACM SIGMOD International Conference on Management of Data, Minneapolis, (1994). 1

[10] C. Fiot, A. Laurent, M. Teisseire, *Motifs sequentiels flous: un peu, beaucoup, passionnement*, e, mesjourneesd'Extraction et Gestion des Connaissances, (2005), 507–518.1

[11] C. Fiot, A. Laurent, M. Teisseire, *From crispness to fuzziness: Three algorithms for soft sequential pattern mining*, IEEE Transactions on Fuzzy Systems, **15** (2007), 1263–1277. 4, 2

[12] G. Dong, J. Pei, *Sequence data mining*, Springer Science& Business Media, Maebashi, (2007). 1

[13] M. N. Garofalakis, R. Rastogi, K. Shim, *SPIRIT: Sequential pattern mining with regular expression constraints*, Proceedings of The 25th International Conference on Very Large Data Bases (VLDB'99), Edinburgh, (1999), 223–234. 1

[14] M. N. Garofalakis, R. Rastogi, K. Shim, *Mining sequential pattern with regular expression constraints*, IEEE Trans. Knowl. Data Eng., **14** (2002), 530–552. 1

[15] F. Guil, A. Bailon, J. Alvarez, R. Marin, *Mining generalized temporal patterns based on fuzzy counting*, Expert Syst. Appl., **40** (2013), 1296–1304. 1

[16] J. Han, M. Kamberl, *Data mining: concepts and techniques*, Academic Press, USA, (2001). 1

[17] J. Han, J. Wang, Y. Lu, P. Tzvetkov, J. Han, *Mining top-k frequent closed patterns without minimum support*, Proceedings of IEEE International Conference Data Mining, Maebashi, (2002), 221–218. 1

[18] T. P. Hong, C. S. Kuo, S. C. Chi, *Mining fuzzy sequential patterns from quantitative data*, Systems, Man and Cybernetics IEEE SMC'99 Conference Proceedings of 1999 IEEE International Conference, (1999), 962–966. 1

[19] T. P. Hong, K. Y. Lin, S. L. Wang, *Mining fuzzy sequential patterns from multiple-item transactions*, IFSA World Congress and 20th NAFIPS International Conference, Joint 9th, (2001), 1317–1321. 1

[20] T. C. K. Huang, *Mining the change of customer behavior in fuzzy time-interval sequential patterns*, Appl. Soft Comput., **12** (2012), 1068–1086. 1

[21] M. Kaya, R. Alhaji, *multi-objective genetic algorithm-based approach for optimizing fuzzy sequential patterns*, Tools with Artificial Intelligence (ICTAI), 16th IEEE International Conference, (2004), 396–400. 1

[22] G. C. Lan, T. P. Hong, Y. H. Lin, S. L. Wang, *Fuzzy utility mining with upper-bound measure*, Appl. Soft Comput., **30** (2015), 767–777. 1

[23] B. Lebaron, S. Weigend, *A bootstrap evaluation of the effect of data splitting on financial time series*, IEEE Trans. Neural Networks, **9** (1998), 213–220. 1

[24] C. H. Lee, M. S. Chen, C. R. Lin, *Progressive partition miner: an efficient algorithm for mining general temporal association rules*, IEEE Trans. Knowl. Data Eng., **15** (2003), 1004–1017. 1

[25] P. Li, Y. Ning, X. S. Wang, S. Jajodia, *Discovering calendar based temporal association rules*, in Proceedings of the 8th International Symposium on Temporal Representation and Reasoning, Cividale, (2001), 111–118. 1

[26] H. Mannila, H. Toivonen, A. I. Verkamo, *Discovery of frequent episodes in event sequences*, Data Min. Knowl. Discovery, **1** (1997), 259–289. 1

[27] K. Mehta, S. Bhattacharyya, *Adequacy of training data for evolutionary mining of trading rules*, Decis. Support Syst., **37** (2004), 461–474. 1

[28] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, M. C. Hsu, *PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth*, Proceedings of International Conference on Data Engineering, Heidelberg, (2001), 215–224. 1

[29] R. Srikant, R. Agrawal, *Mining sequential patterns: generalizations and performance improvements*, in Research Report RJ 9994, IBM Almaden Research Center, San Jose, California, (1995). 1

[30] R. Srikant, R. Agrawal, *Mining sequential patterns: generalizations and performance improvements*, Proceedings of the 5th International Conference on Extending Database Technology, Avignon, (1996). 1, 3, 3

[31] R. Srikant, Y. Yang, *Mining web logs to improve website organization*, Proceedings of the 10th International World Wide Web Conference, Hong Kong, (2001). 1

[32] J. Srivastava, *Mining temporal data*, http://www.cs.umn.edu/research/websift/survey/. 1

[33] X. Yan, J. Han, *CloSpan: mining closed sequential patterns in large datasets*, Proceedings of the 2003 SIAM International Conference on Data Mining (SDM'03), San Francisco, (2003). 1

[34] J. Yang, W. Wang, P. Yu, J. Han, *Mining long sequential patterns in a noisy environment*, Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, (2002), 406–417. 1

[35] F. Zabihi, *Fuzzy constrained sequential pattern mining*, MSc Thesis, Kharazmi University, Tehran, Iran, (2010). 1

[36] F. Zabihi, M. Ramezan, M. M. Pedram, A. Memariani, *Constrained Sequential Pattern Mining*, in The 2nd congress on Data Mining, Tehran, Iran, (2008). 1

[37] F. Zabihi, M. Ramezan, M. M. Pedram, A. Memariani, *A novel algorithm for fuzzy data mining with sliding window and time gap constrains*, in the 2nd joint congress on Fuzzy and Intelligent Syatems, Tehran, Iran, (2008). 1

[38] F. Zabihi, M. Ramezan, M. M. Pedram, A. Memariani, *Fuzzy sequential pattern mining with sliding window constraint*, in 2nd IEEE International Conference on Education Technology and Computer (ICETC), Shanghai, China, (2010). 1, 2

[39] F. Zabihi, M. Ramezan, M. M. Pedram, A. Memariani, *Rule Extraction for Blood donators with fuzzy sequential pattern mining*, J. Math. Comput. Sci., **2** (2011), 37–43. 1

[40] J. M. Zaki, *SPADE: An efficient algorithm for mining frequent sequences*, Machine Learning, **42** (2001), 31–60. 1

[41] Q. Zhao, S. S. Bhowmick, *Sequential pattern mining: a survey, Technical Report*, School of Computer Engineering, Nanyang Technological University, Singapore, (2003). 1