MINING FUZZY TEMPORAL ITEMSETS WITHIN VARIOUS TIME INTERVALS IN QUANTITATIVE DATASETS

M. KADKHODA, M. R. AKBARZADEH-T AND S. M. TAHERI

This paper is dedicated to Professor L. A. Zadeh on the occasion of his 95th birthday and the 50th year of the birth of fuzzy logic

Abstract. This research aims at proposing a new method for discovering frequent temporal itemsets in continuous subsets of a dataset with quantitative transactions. It is important to note that although these temporal itemsets may have relatively high *support* or occurrence within particular time intervals, they do not necessarily get similar *support* across the whole dataset, which makes it almost impossible to extract them using the existing traditional algorithms. This paper directly addresses this problem and introduces a new algorithm called Fuzzy Solid Linguistic Itemset Mining (FSLIM) to discover Solid Linguistic Itemsets (SLIs) within a quantitative dataset. SLI is a new concept introduced here as an essential part of the solution presented in this paper. The proposed method consists of two phases. In the first phase, fuzzy set theory is used to transform each quantitative value to a linguistic item; and in the second phase, all SLIs are extracted. Finally, the efficiency of FSLIM is compared in terms of execution time, scalability and the number of frequent patterns with those of two classic approaches on synthetic datasets. The proposed approach is also applied to an actual Mashhad Urban Traffic dataset in order to illustrate FSLIM's ability in discovering the hidden knowledge that could not be extracted by traditional methods.

1. Introduction

With the explosive increase in the amount of data generated by different sources, extracting valuable knowledge from large datasets has become more important than ever. Among various existing data mining approaches, association rule mining [1] is one of the popular methods for revealing the very interesting correlations between data items in large datasets. Frequent itemset mining is the first stage in discovering these association rules that use a specific threshold value called *Minimum Support* to determine if a given itemset is *frequent* or not. Since the early work of Agrawal et al. [1], several efficient approaches have been proposed, covering a broad range of solutions for the frequent itemset mining problem [22, 23, 7]. However, the factor of time has not been addressed in any of these traditional works. In other words, the times when transactions occur play no part in their mining solutions and they basically extract all the unordered correlations among items of a data collection regardless of the time of their occurrence. This raises a huge challenge

Invited Paper: September 2016

Key words and phrases: Fuzzy data mining, Temporal data mining, Frequent itemset, Temporal quantitative dataset.

in dealing with problems in the real-world, since time is often an essential element that requires specific attention.

Temporal frequent itemset mining adds time constraints (in the form of time points or time intervals) to the frequent itemsets to indicate when they hold. Chen and Petrounias [4] first addressed the concept of temporal frequent itemsets. Ale and Rossi [2] proposed an approach for discovering temporal frequent itemsets by utilizing the lifespan of items, where the lifespan of an item is a period of time which starts from the first appearance of the item to the moment it disappears. Chen et al. [13] introduced an algorithm called Progressive Partition Miner (PPM) in order to mine temporal frequent itemsets in a publication dataset. Working on the same algorithm, other researchers extended PPM to include items with different exhibition time periods [10, 5]. Hong et al. introduced the concept of up-to-date itemsets which concerned the most recent frequent items with an unfixed window size and developed two approaches for discovering them [17, 8]. Although the mentioned approaches extract frequent itemsets within the time shorter than the whole dataset's lifespan, these times are determined by the known lifespan of each item. Hence, there would be no need to have complicated time interval extraction processes.

Besides the normal temporal frequent itemsets mentioned above, market managers are also interested in the frequent itemsets, occurring in specific time intervals to make better decisions. For instance, in a supermarket dataset which records every sale transaction, bread and cheese sales together may only form 2% of all transactions, whereas essentially, the majority of these two items are sold between 8 and 10 am. In fact, we may find that 45% of transactions that occur between 8 and 10 am include both bread and cheese together. Thus, it would not be wrong to conclude that the itemset (bread, cheese) is a frequent itemset during time interval of 8-10 am. This example shows that many frequent itemsets could exist within different time intervals, and these cannot be extracted using traditional algorithms.

There is a wide range of applications for using frequent itemsets within time intervals. In marketing, for instance, these sets of time intervals reveal the behavior of frequent itemsets and the managers can adjust their marketing strategy to improve their business. Similarly, in medical industry, knowing the time interval when an illness (illnesses) has occurred would be highly useful in managing epidemics. Furthermore, collecting information about user behaviours in the websites of on-line stores, on-line conferences and, in general, social networks can provide valuable information for managers in the process of decision-making and offering new services.

This fact has motivated researchers to find new approaches in micro-analysis of datasets to discover this hidden knowledge. Saleh and his colleagues [24] proposed SIM algorithm to extract frequent itemsets in a temporally contiguous subset of a dataset and coined the term "Solid Itemset." According to them, Solid Itemsets (SIs) are not necessarily frequent over the entire dataset, and they used the same threshold value to discover all SIs. Matthews et al. [18] presented an approach that uses genetic algorithm to mine frequent itemsets in time intervals of a dataset. Suvvari et al. [25] introduced the concepts of significant time intervals and maximal

significant time intervals and proposed an algorithm to extract frequent itemsets. In this approach, the *support* of a frequent itemset in a significant time interval must be equal to or higher than its *support* in the entire dataset. In a similar work, Xiao et al. [26] proposed a minsup-minconf-minwin framework for mining association rules with a time-window(ARTW) in temporal transaction datasets. This algorithm is later extended in [27] to determine maximum frequent time window using a variable neighborhood search algorithm.

All the algorithms mentioned above are defined on binary datasets and may not be used for quantitative ones, while in reality a huge number of datasets are quantitative. Fuzzy set theory, due to its simplicity as well as its similarity to human reasoning, is one of the most important techniques that is frequently used in data mining [29, 11]. Despite all the valuable works done in fuzzy temporal frequent itemset mining so far [14, 15, 6, 30, 12], there are a few studies [19, 20] that are close to the problem addressed in this paper, but they are not the same. Matthews and his colleagues in these two papers focus on how the membership functions of quantitative items are defined. For this reason, in these algorithms the membership functions are not fixed at first and they use GA for tuning the lateral displacements of membership functions in random subsets of the dataset. So each rule is specific to its subset.

This paper, conducts a new study on developing an efficient method to mine frequent itemsets within time intervals in a quantitative transaction dataset. Expanding on SIM algorithm, the goal is to find any itemsets that are frequent over specific time intervals in a quantitative dataset but would not be extracted by traditional methods since their *support* is very low over the whole dataset. Size of these time intervals is not known in advance and shown to be optimal by following this algorithm. Hence, the main contributions of this paper are threefold. The first is discovering the optimal time intervals for each frequent itemset in a quantitative transaction dataset. The second is introducing the concept of **Solid Linguistic Itemset**(SLI). And the third is proposing a new efficient approach called **FSLIM** for extracting all SLIs where items in the specified interval at least enjoy a predefined *Minimum support*.

The rest of the paper is organized as the following: In Section 2, several important definitions are addressed and a new SLI definition is introduced. Section 3 explores details of the SLI mining algorithm. Several simulated experiments and the findings are reported in Section 4. And lastly, the paper concludes with a short discussion in Section 5.

2. Definitions and Problem Statement

Discovering frequent itemsets is the basis of association rule mining. Here, we modify the traditional definition of frequent itemsets where each item is not only associated with a quantitative value but has a time stamp as well. More specifically, the following definitions are provided to help explain the problem of fuzzy linguistic solid mining. For easier referencing, Table 1 provides a list of symbols used throughout this paper.

Symbol	Meaning	Symbol	Meaning
D	The temporal quantitative transaction dataset	η	A set of linguistic items in D'
φ	A set of ordered time instances	L_k	A k-linguistic itemset
Ω	List of items in D	T'_r	r^{th} temporal linguistic transaction in D'
m	The number of items in D	D'	A temporal linguistic transaction dataset
n	The number of transactions in D and D'	$\|D'\|$	The number of temporal linguistic transactions in D'
I_j	The j^{th} item in D, $1 \leq j \leq m$	$\mathcal{T}$	A time interval in D'
I'_j	The j^{th} quantitative item in D, $1 \leq j \leq m$	t_s	the start time of $\mathcal{T}$
k	Length of an itemset	t_e	the end time of $\mathcal{T}$
S_k	A k-itemset, $1 \leq k \leq m$	Trans($\mathcal{T}$)	The set of temporal linguistic transactions occurred in $\mathcal{T}$
S'_k	A quantitative k-itemset, $1 \leq k \leq m$	$\mathcal{T}_{\text{all}}$	the set of all potentially possible times intervals in D'
T_r	The r^{th} temporal quantitative transaction in D, $1 \leq r \leq n$, $T_r = \langle r, \varphi_r, S'_{k,r} \rangle$	c_k	A candidate with size k
r	Transaction ID in D and D'	C_k	A set of candidates with size k
φ_r	The time instance when the transaction occurred	$\mathcal{T}_c$	The time interval of c_k
$S'_{k,r}$	The associative quantitative k-itemset in r^{th} temporal quantitative transaction	$knlist$	The list of existing kernels in $\mathcal{T}_c$
v_j^r	The quantitative value of item I_j in T_r	knl	A kernel of a linguistic itemset
μ_{jq}^r	The membership degree of in v_j^r linguistic term l_{jq}	$\mathcal{T}_{knl}$	The time interval of knl
l_{jq}	The q^{th} linguistic terms (fuzzy region) for item I_j	Tfs	The temporal fuzzy support of knl
h	The number of linguistic terms for I_j	Γ	list of linguistic items in D'
M	The number of linguistic items in D'	γ	The user predefined Minimum support
l_i	The i^{th} linguistic item in D', $1 \leq i \leq M$	ξ	The set of predefined membership functions
Sc_k	The number of candidates with size k	SLI_k	list of solid linguistic itemsets with size k
$\mu_{c_k}^{(r)}$	Membership degree of c_k in T'_r	$SLIs$	list of solid linguistic itemsets

TABLE 1. List of Symbols Used in the Proposed Algorithm

Definition 2.1. Let $\Omega = \{ I_1, I_2, \ldots, I_m\}$ be a set of **items**. Consider **k-itemset** $S_k = \{I_1, I_2, \ldots I_k\} \subset \Omega$ where $1 \leq k \leq m$. A **quantitative item** I'_j is a tuple $I'_j =< I_j, v_j >$, where v_j is a positive real number, $\forall\ 1 \leq j \leq m$. A **quantitative k-itemset** S'_k is then defined as $S'_k = \{I'_1, I'_2, \ldots, I'_k\}$.

Definition 2.2. Let $\varphi = \{\varphi_1, \varphi_2, \ldots, \varphi_n\}$ be a set of ordered time instances, where $\forall\ u \leq w,\ \varphi_u \leq \varphi_w$. A **temporal quantitative transaction** T_r is a 3-tuple $T_r =< r, \varphi_r, S'_{k,r} >$ such that r is the transaction ID, φ_r is the time instance when the transaction occurred, and $S'_{k,r}$ is the associated quantitative k-itemset.

For simplicity, since only one transaction is assumed to occur at each time instance here, r is used in place of the time instance, hence $T_r =< r, S'_{k,r} >$. A **temporal quantitative transaction dataset** D is then defined as $D = \{T_1, T_2, \ldots, T_n\}$.

Example 2.3. Table 2 shows a temporal quantitative transaction dataset(D) with ten temporal quantitative transactions. There are six items $\Omega = (A, B, C, D, E, F)$ and each item has an associated quantitative value. For instance, item A in Transaction 1 has value 5. As can be seen, in this dataset, r refers to the time instance of each transaction.

r	T_r
1	$A:5$ $B:9$ $C:12$ $D:6$ $E:14$
2	$A:4$ $C:9$ $D:1$ $E:5$ $F:2$
3	$A:3$ $B:12$ $C:5$ $D:12$ $E:4$
4	$B:8$ $C:8$ $D:14$ $E:6$
5	$B:7$ $F:3$
6	$D:9$
7	$A:2$ $D:4$ $E:5$
8	$A:6$ $C:11$
9	$B:5$
10	$A:13$ $B:3$ $C:13$ $E:1$

TABLE 2. A Temporal Quantitative Transaction Dataset (D)

In order to deal with quantitative values, fuzzy set theory is commonly used to transform these values to linguistic terms. Using a set of membership functions, each value v_j^r, the quantitative value of item I_j in T_r, can be transformed to a fuzzy set: $\left\{\frac{\mu_{j1}^r}{l_{j1}} + \frac{\mu_{j2}^r}{l_{j2}} + \cdots + \frac{\mu_{jh}^r}{l_{jh}}\right\}$, where l_{jq} is the q^{th} linguistic terms (fuzzy region) for item I_j and $\mu_{jq}^r$is the membership degree of v_j^r in linguistic term l_{jq} and h is the number of linguistic terms for I_j.

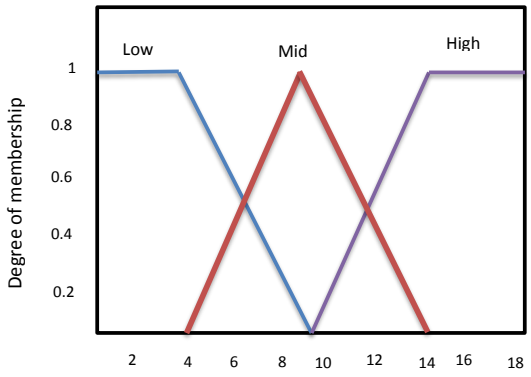


FIGURE 1. The Predefined Membership Function

r	*A*			*B*			*C*			*D*			*E*			*F*		
	L	M	H	L	M	H	L	M	H	L	M	H	L	M	H	L	M	H
1	0.8	0.2	0	0	1	0	0	0.4	0.6	0.6	0.4	0	0.8	0.2	1	0	0	0
2	1	0	0	0	0	0	0	1	0	1	0	0	0.8	0.2	0	1	0	0
3	1	0	0	0	0.4	0.6	0.8	0.2	0	0	0.4	0.6	1	0	0	0	0	0
4	0	0	0	0.2	0.8	0	0.2	0.8	0	0	0	1	0.6	0.4	0	0	0	0
5	0	0	0	0.4	0.6	0	0	0	0	0	0	0	0	0	0	1	0	0
6	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
7	1	0	0	0	0	0	0	0	0	1	0	0	0.8	0.2	0	0	0	0
8	0.6	0.4	0	0	0	0	0	0.6	0.4	0	0	0	0	0	0	0	0	0
9	0	0	0	0.8	0.2	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0.2	0.8	1	0	0	0	0.2	0.8	0	0	0	1	0	0	0	0	0

TABLE 3. A Temporal Linguistic Transaction Dataset (D')

Example 2.4. Using dataset D as presented in Table 2 and fuzzy membership functions as shown in Figure 1, quantitative items in dataset can be substituted with **linguistic items** as well as membership degrees. The result will be similar to Table 3. For example, 5^{th} transaction in the dataset contains these linguistic items: *B.Low*, *B.Mid* and *F.Low* with membership degrees of 0.4, 0.6 and 1, respectively. In this method, frequent itemset mining algorithm is executed on the new linguistic dataset D'. As a result, each mined frequent itemset is expressed in a linguistic format that is more amenable to human perception.

Definition 2.5. Let $\eta = \{l_1,\ l_2, \ldots,\ l_M\}$ be a set of linguistic items l_i, $1 \leq l_i \leq M$. A **k-linguistic itemset** is defined as $L_k = \{\ l_1,\ l_2, \ldots,\ l_k\} \subset \eta$ where $1 \leq k \leq M$.

Definition 2.6. Let $\varphi = \{\varphi_1,\ \varphi_2, \ldots,\ \varphi_n\}$ be a set of ordered time instances, where $\forall\ i \leq j,\ \varphi_i \leq \varphi_j$. A **temporal linguistic transaction** ${T_r}'$ is a 3-tuple ${T_r}' =< r, \varphi_r, L_{k,r} >$ such that r is the transaction ID, φ_r is the time instance when the transaction occurred, and $L_{k,r}$ is the associated k-linguistic itemset. For simplicity, in this research, ${T_r}'$ is denoted as ${T_r}' =< r,\ L_{k,r} >$ since only one transaction is assumed to occur at each time instance. A **temporal linguistic transaction dataset** D' is then defined as $D' = \{{T'}_1,\ {T'}_2, \ldots,\ {T'}_n\}$.

None of the definitions explained thus far apply to different time intervals within the dataset. Below are a few further definitions introduced in this research that consider time interval as an important factor:

Definition 2.7. Let $\mathcal{T} = [\ t_s, t_e]$ be a time interval, where t_s is the start time and t_e is the end time of this interval. The set of temporal linguistic transactions that occur in this interval is denoted as $Trans(\mathcal{T}) = \{T'_r \big|\ \mathcal{T} \cdot t_s \leq r \leq \mathcal{T} \cdot t_e,\ T' \in D'\}$. $\mathcal{T}_{all}$ is defined as the set of all potentially possible time intervals in D'.

Definition 2.8. A temporal linguistic transaction ${T_r}' =< r, L_{k,r} >$ supports linguistic itemset L_s if $L_s \subseteq L_{k,r}$. The **cover** of a linguistic itemset L_s in time interval $\mathcal{T}$ is the set of transaction ID of temporal linguistic transactions in D' that support L_s.

Definition 2.9. Temporal fuzzy support of a linguistic item l_i in time interval $\mathcal{T}$ is defined as follows:

$$TfSupport\,(l, \mathcal{T}) = Sumdegree(\mathcal{T}, l_i)/|Trans\,(\mathcal{T})\,|$$

Where $Sumdegree\,(\mathcal{T}, l_i) = \sum_{r=\mathcal{T}.t_s}^{\mathcal{T}.t_e} {\mu_{l_i}}^{(r)}$ and ${\mu_{l_i}}^{(r)}$ is the membership degree of linguistic item l_i in ${T_r}'$. In other words, the *temporal fuzzy support* of a linguistic item l_i in the time interval of $\mathcal{T}$ is the ratio of its scalar cardinality to the number of temporal linguistic transactions that occurs in $\mathcal{T}$.

Definition 2.10. Temporal fuzzy support of a k-linguistic itemset L_k is computed as follows:

$$TfSupport\,(L_k, \mathcal{T}) = Sumdegree(\mathcal{T}, L_k)/|Trans\,(\mathcal{T})\,|$$

Where $Sumdegree\,(\mathcal{T}, L_k) = \sum_{r=\mathcal{T}.t_s}^{\mathcal{T}.t_e} {\mu_{L_k}}^{(r)}$ such that ${\mu_{L_k}}^{(r)} = Min_{p=1}^{k}\ {\mu_{l_p}}^{(r)}$ and ${\mu_{l_p}}^{(r)}$ is the membership degree of p^{th} linguistic item of L_k in ${T_r}'$.

This research intends to extract linguistic items frequent in a specific time interval in the dataset where these time intervals are optimal. Therefore, the two core proposed concepts of this paper are temporal linguistic itemset and solid linguistic itemset as defined below.

Definition 2.11. A triple $X = (L, \mathcal{T}, \delta)$ is called a **temporal linguistic itemset** where L is a linguistic itemset, $\mathcal{T}$ is the time interval associated with L and δ is the *temporal fuzzy support* of L over the time interval $\mathcal{T}$. X is referred as a **k-temporal linguistic itemset** if $k = |L|$.

Also, **Solid Linguistic Itemset** (SLI) is proposed as follows:

Definition 2.12. Given a user-predefined *Minimum support* γ, let $X = (L, \mathcal{T}, \delta)$ be a temporal linguistic itemset. X is called a **Solid Linguistic Itemset (SLI)** if the following conditions hold:

(1) $X.\delta \geq \gamma$
(2) $\forall\ \mathcal{T}_2 \in \mathcal{T}_{all}$ such that $X.\mathcal{T} \subseteq \mathcal{T}_2$, $TfSupport\,(L, \mathcal{T}_2) < \gamma$
(3) $\forall\ \mathcal{T}_2 \in \mathcal{T}_{all}$ such that $\mathcal{T}_2 \subset X.\mathcal{T}$, $\text{cover}(L, \mathcal{T}_2) < \text{cover}(L, X.\mathcal{T})$.

The above three conditions guarantee that the size of $\mathcal{T}$ is optimal. The first condition represents that L is frequent in the associated time interval $\mathcal{T}$. The second condition guarantees that if there is a larger time interval, then L is not frequent in this time interval. Therefore, the size of $\mathcal{T}$ is maximal. Finally, because the first and the last transaction in the time interval $\mathcal{T}$ contains L, the cover is reduced in all of subset time intervals of $\mathcal{T}$. Hence, the third condition ensures that the size of $\mathcal{T}$ is minimal.

Example 2.13. Using dataset D from Example 2.3 and assuming threshold value of 0.4 for predefined *Minimum support* ($\gamma = 0.4$), SLIs presented in Table 4 could be identified in D. As can be seen, each time interval or contiguous subset of D is denoted by r's, e.g., $\partial_1 = (A.Low, [1, 8], 0.55)$ is a SLI.

No	Itemset	Time Interval	Minimum support
1	A.Low	[1 ,8]	0.55
2	B.Mid	[1 ,5]	0.8
3	C.Mid	[1 ,4]	1
4	E.Low	[2,10]	0.47
5	(A.Low, C.Mid)	[1 ,3]	0.89
6	(A.Low, E.Low)	[2 ,7]	0.43
7	(C.Mid, E.Low)	[2 ,4]	0.53

TABLE 4. SLIs Found in Dataset D'

3. The Proposed Algorithm

The FSLIM (Fuzzy Solid Linguistic Itemset Mining) approach, proposed in this paper consists of two phases: In the first phase, each quantitative item from temporal quantitative transaction dataset D is substituted with linguistic terms and

their membership degrees to obtain D', a temporal linguistic transaction dataset. In the second phase, itemset candidates are generated; frequent itemsets are identified and finally presented in linguistic terms that are more intelligible for human thinking. The flowchart of the proposed FSLIM algorithm is presented in Figure 2. Although the skeleton of the second phase is similar to the previous approach (SIM) proposed in [24], it should be highlighted that unlike the previous one, this algorithm is developed to work on quantitative transaction datasets. As a result, new concepts such as SLI are introduced and some of the existing definitions are either revised or extended. To give an example, *Temporal fuzzy support* are defined as an extended version of *frequency* used in [24]. The proposed algorithm is based on these new concepts and definitions.

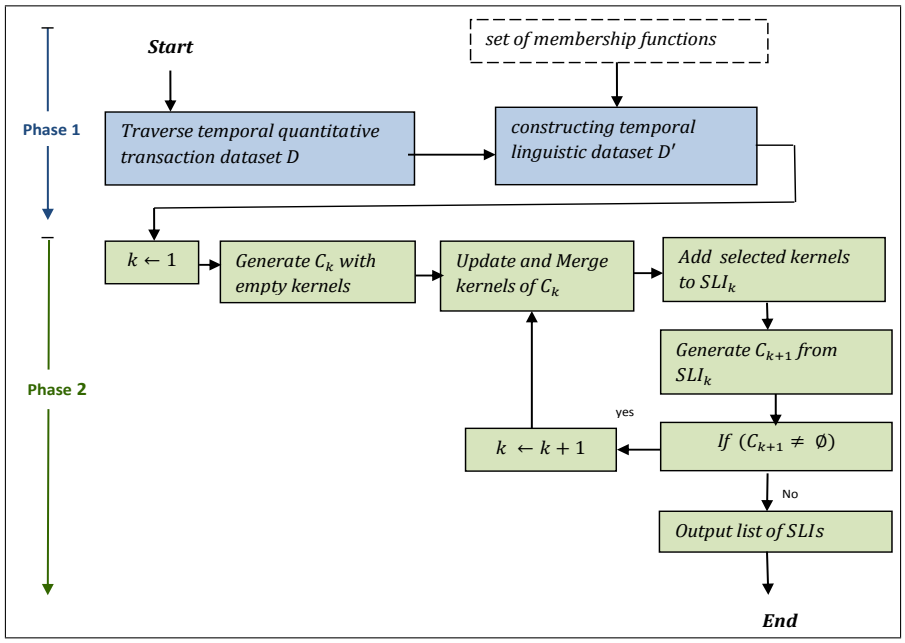


FIGURE 2. The Flowchart of FSLIM

As noted above, candidate generation is the basis for the second phase of the algorithm. It starts with candidates of size 1 and continues to generate candidates with larger size. Whereas the aim is to discover SLIs in the dataset, the structure of each candidate is different from those of traditional algorithms.

Definition 3.1. A **candidate** c_k is denoted by $c_k = (L_k, \mathcal{T}_c, knlist\ )$ where L_k is a k-linguistic itemset, $\mathcal{T}_c$ is the time interval of candidate and $knlist$ is the list of existing *kernels* in this time interval.

Definition 3.2. A **kernel** of a linguistic itemset refers to the longest continuous subset of dataset or time interval where the given linguistic itemset is frequent and the first and last transaction of this time interval support it. Each kernel is

represented here as $knl = (\mathcal{T}_{knl}, Tfs)$ where $\mathcal{T}_{knl}$ is the time interval of knl, and Tfs is the *temporal fuzzy support* of knl.

As mentioned above, each candidate in this approach consists of kernels representing the time intervals in which the candidate is frequent. In fact, kernels are where the main focus of this approach lies. First, based on linguistic items in D', a list of size 1 candidate containing empty kernels is generated. Then, through scanning the dataset using **UpdateKernels** algorithm, all kernels per each candidate are extracted. The details of updating kernels for one candidate is illustrated in

Algorithm 1 UpdateKernels

Input:
c_k : candidate with size k i.e. $c_k = (L_k, \mathcal{T}_c, knlist)$.
T'_r: where $(T'_r.r \in c_k.\mathcal{T}_c)$.
γ: The user predefined *Minimum support*.
Output: updated kernels of candidate C_k.

1) If all kernels in c_k are *closed*
   1-1) If $(c_k.L_k \subset T'_r)$
      1-1)1. create a new open kernel $knl = (\mathcal{T}_{knl}, Tfs)$.
      1-1)2. $Tfs = \mu^{(r)}_{L_k} = \mu^{(r)}_{l_1} \wedge \mu^{(r)}_{l_2} \wedge \cdots \wedge \mu^{(r)}_{l_k}$ where $l_p \in L_k$, $1 \leq p \leq k$ and $\mu^{(r)}_{l_p}$ is the membership degree of linguistic item l_p in T'_r.
      1-1)3. $\mathcal{T}_{knl} = [r, r]$.
2) else if $(c_k.L_k \subset T'_r)$
   2-1) Compute $\mu^{(r)}_{L_k} = \mu^{(r)}_{l_1} \wedge \mu^{(r)}_{l_2} \wedge \cdots \wedge \mu^{(r)}_{l_k}$ where, $l_p \in L_k$, $1 \leq p \leq k$ and $\mu^{(r)}_{l_p}$ is the membership degree of linguistic item l_p in T'_r.
   2-2) $fsup = sumfd/|Trans(\mathcal{T}_{knl}) + 1|$, where $sumfd = \sum_{u=\mathcal{T}_{knl.ts}}^{r} \mu^{(u)}_{L_k}$.
   2-3) If $(fsup > \gamma)$ update the current kernel $knl = (\mathcal{T}_{knl}, Tfs)$ using the following steps:
      2-3)1. $\mathcal{T}_{knl.te} = r$.
      2-3)2. $Tfs = fsupp$.
      else
      2-3)3. Close the current kernel
      2-3)4. create a new open kernel $knl = (\mathcal{T}_{knl}, Tfs)$.
      2-3)5. $\mathcal{T}_{knl} = [r, r]$.
      2-3)6. $Tfs = \mu^{(r)}_{L_k}$.
3) else
   3-1) $\mathcal{T}_{knl.t_e} = r$
   3-2) compute Tfs with considering T'_r
   3-3) If $(Tfs < \gamma)$ close the current kernel (knl) and set $\mathcal{T}_{knl.t_e}$ and Tfs to the previous ones.

End UpdateKernels.

Algorithm 2 MergeKernels

Input:
c_k : candidate with size k i.e. $c_k = (L_k, \mathcal{T}_c, knlist)$.
γ: The user predefined *Minimum support*.
Output: c_k with merged kernels;

1) Mergeable $\leftarrow$ true;
2) deletelist $\leftarrow \varnothing$;
3) $q \leftarrow 0$;
4) if (Mergeable=true)
   4-1) Mergeable $\leftarrow$ false;
   4-2) If $(q < sizeof(knlist) - 1)$
      4-2)1. $r \leftarrow q + 1$;
      4-2)2. $start \leftarrow knlist \cdot q \cdot t_s$
      4-2)3. $end \leftarrow knlist \cdot r \cdot t_e$;
      4-2)4. $f\sup \leftarrow sumfd/|start{-}end{+}1|$, where $sumfd = \sum_{r=start}^{end} \mu_{L_k}^{(r)}$
      4-2)5. if $(f\sup \geq \gamma)$
         4-2)5.1. $knlist \leftarrow knlist + knlist \cdot q \cup knlist \cdot r$.
         4-2)5.2. deletelist $\leftarrow$ deletlist $+q + r$;
         4-2)5.3. Mergeable $\leftarrow$ true;
      4-2)6. if (Mergeable=true)
         4-2)6.1. $knlist$.removeAll(deletlist);
         4-2)6.2. deletlist $\leftarrow \varnothing$;
5) $q \leftarrow q + 1$;
6) goto 4;

End MergeKernels.

Algorithm 1. Next, in order to obtain the longest time intervals in which a given candidate is frequent, the kernels of each candidate are merged using **MergeKernels** algorithm (depicted in Algorithm 2). According to our proposed method, if *temporal fuzzy support* of the linguistic itemset in the resulting kernel is more than the predefined *Minimum support*, the two kernels can be merged. Then, each candidate containing the kernels with a cover more than *Mincover*, is added to SLIs. As it can be seen, *Mincover* is considered as a threshold in this approach, because according to Definition 2.11, explained in the previous section, any linguistic itemset that is supported by at least one transaction can be considered a Solid Linguistic Itemset (SLI). Since this causes problems resulting in a huge number of SLIs, a fixed threshold, called *Mincover* is defined. And prior to adding any candidate to SLIs, it needs to be compared and verified against the *Mincover* threshold. Finally, SLIs are joined to generate candidates for the next step in the same way in which the candidates are generated in *apriori* approach. There are, however, two restrictions: a) two SLIs must have a common time interval greater than zero, and b) two linguistic terms of one item cannot be concurrently in the linguistic itemset

of the generated candidate. These steps are repeated until no further candidates can be generated. The details of the proposed FSLIM algorithm are presented in Algorithm 3.

The following theorem shows that **MergeKernels algorithm** generates optimal SLIs as its output.

Theorem 3.3. *Let $c_k = (L_k, \mathcal{T}_c, knlist)$ such that knlist is the set of kernels of L_k in $\mathcal{T}_c$ with respect to γ. **MergeKernels** finds optimal solid linguistic itemset $SLI_k = (L_k, \mathcal{T}, \delta)$.*

Proof. If we assume that $knl = (\mathcal{T}, \delta)$ is the output of **MergeKernels,** meaning it cannot be merged with any other kernels in *knlist*, then:

(1) $\delta > \gamma$, because in **MergeKernels,** *temporal fuzzy support* of each kernel is checked (step 4-2-5) and must be greater than γ.

(2) $\forall\ \mathcal{T}_p \in \mathcal{T}_{all}, knl \subset \mathcal{T}_p$,
   (a) If $(L_k \in \mathcal{T}_p - knl)$ then $TfSupport\,(L_k, \mathcal{T}_p)$ is smaller than γ. Otherwise, it would have been added to *knl* at first or it would have belonged to another kernel *knl*1 and then would be merged with *knl*.
   (b) If $(L_k \notin \mathcal{T}_p - \ knl)$ then cover($\mathcal{T}_p$) =cover(knl) and

$$TfSupport\,(L_k, \mathcal{T}_p) < TfSupport\,(L_k, knl)\,.$$

   Besides, each increase in *knl* results in L_k is not being supported by the first and last transaction in *knl*.

(3) $\forall\ \mathcal{T}_p \in \mathcal{T}_{all}$, $\mathcal{T}_p \subset knl$, there are two cases:
   (a) If $L_k \subset T_{\mathcal{T}_{p.t_s}}$ and $L_k \subset T_{\mathcal{T}_{p.t_e}}$ even if $TfSupport(\mathcal{T}_p) > \gamma$, it is not the longest continuous subset, hence, not considered as a kernel.
   (b) Otherwise, according to Definition 3.2 it is not a kernel.

Therefore, $SLI_k = (L_k, \mathcal{T}, \delta)$ is an optimal solid linguistic itemset. □

4. Experiments

To evaluate the proposed algorithm, FSLIM is conducted on synthetic and real datasets. The implementation is done in JAVA platform and all simulated tests are run on a personal computer with an Intel(R) Core (TM) i7-2700K CPU (3.5 GHz) and a 8GB RAM. At first, findings of several experiments using synthetic data to evaluate computational efficiency and scalability of its performance are presented. In addition, specific experiments are designed to compare FSLIM algorithm with the more traditional ones. Then, to show the utility of SLIs extracted by the FSLIM algorithm, we conducted an experiment on a real world dataset.

4.1. Synthetic Data Generation.

A considerable number of current works on frequent itemset mining and association rule mining [18, 20, 3, 16, 21] have used the IBM Quest Synthetic Data Generator [18]. Hence, this platform is also chosen here as a benchmark. The output of this approach is a market basket dataset consisting sold items in every shopping basket without any quantities of items. To generate quantitative dataset, in the similar manner to [20], quantities of items sold in a

Algorithm 3 FSLIM

Input:
D : A temporal quantitative transaction dataset.
γ: The user predeined *Minimum support*.
ξ: the set of predefined membership functions.
Output:
SLls: the set of solid linguistic itemsets.
Phase A: Constructing temporal linguistic dataset D′

1) Set $\Gamma \leftarrow \varnothing$. //list of linguistic items in D'
2) for each T_r in D do the following substeps:
   2-1) convert $v_j^{(r)}$ to fuzzy set $\left\{ \frac{\mu_{j1}^r}{l_{j1}} + \frac{\mu_{j2}^r}{l_{j2}} + \cdots + \frac{\mu_{jh}^r}{l_{jh}} \right\}$ using ξ where l_{jq} is the q^{th} linguistic terms (fuzzy region) for item I_j and μ_{jq}^r is the memebership degree of v_j^r in linguistic term l_{jq} and h is the number of linguistic terms for I_j.
   2-2) Insert (l_{jq}, μ_{jq}^r) into D' where $\mu_{jq}^r \neq 0, 1 \leq q \leq h$.
   2-3) Add l_{jq} to Γ where $\mu_{jq}^r \neq 0, 1 \leq q \leq h$.

Phase B: Discovering Solid linguistic itemsets in D′

1) $k \leftarrow 1$ // the number of linguistic items in the current candidastes to be processed.
2) $C_1 \leftarrow \varnothing$; // list of candidastes with lengh 1
3) For each ($w \in \Gamma$) Generate a candidate with empty kernels and add to C_1.
4) $SLI_k \leftarrow \varnothing$; //list of solid linguistic itemsets with size k.
5) Update kernels of each $c_k \in C_k$ according to each transaction T_r' in $D', 1 \leq r \leq n$.
6) Merge kernels of each candidate c_k;
7) Add all *kernels* of each c_k with *cover* greater than *Mincover* to SLI_k.
8) Join solid linguistic itemsets in SLI_k to generate the candidate set C_{k+1} similar to candidate generation in *apriori* considering these restrictions:
   a) two solid linguistic items must have a common time interval greater than zero
   b) two linguistic terms of one item cannot be concurrently in the linguistic itemset of C_{k+1}.
9) $SLIs \leftarrow SLIs + SLI_k$;
10) If $(C_{k+1} \neq \varnothing) k \leftarrow k + 1$ and goto step 4.
11) Output all solid linguistic itemsets in $SLIs$.

End FSLIM

transaction are randomly generated from a uniform distribution in the range 1–18 and assigned to each item in a transaction. The generated transactions do not have any explicit time, but we can assume that the transactions are inserted to the dataset in the order of their occurrence. Since only one transaction is assumed to occur at each time instance here, transaction Id(r) is used in place of time instance

and the generated dataset is considered as a temporal data set. This assumption has also been made in other works focusing on temporal data mining (e.g., [24, 28]). The parameters used in generating datasets are shown in Table 5.

Parameters	**Meaning**
$\|D\|$	Number of transactions
$\|T\|$	Average length of transactions
$\|N\|$	Number of items in dataset
M	Max quantity value of items

TABLE 5. Parameters Used in Dataset Generation

Name	$\|\boldsymbol{N}\|$	$\|\boldsymbol{D}\|$
T5N064D10K	64	10000
T5N2D20K	2000	20000
T5N4D40K	4000	40000
T5N6D60K	6000	60000
T5N8D80K	8000	80000
T5N10D100K	10000	100000
T5N50D500K	50000	500000

TABLE 6. Synthetic Datasets

In generating datasets, average length of each transaction is considered 5 ($|\mathbf{T}|$=**5**) and Max quantity value of items is set to 18 ($\mathbf{M} = 18$). Table 6 summarizes the parameter settings and synthetic dataset generated.

4.2. **Methodology.** There are several algorithms that address frequent itemset mining within the time intervals of a temporal dataset. However, these algorithms cannot be applied to quantitative datasets. Hence, in order to evaluate the efficiency of the proposed algorithm based on accuracy in extraction of knowledge and the execution time of implementation, it is compared with two of the traditional fuzzy frequent pattern mining algorithms that are applicable to quantitative datasets. Several experiments have been also designed to illustrate the scalability of the proposed algorithm based on different thresholds and their impacts in the execution time of algorithm, number of candidates and extracted SLIs. It is important to highlight that the same membership function (shown in Figure 1) is used for all items.

4.3. **The Effect of Thresholds on Execution Time and SLIs.** The first experiment aims at understanding the execution time required to run the algorithm and the impact that the two threshold values *Minimum support* (γ) and *Mincover* could have on this time. For this purpose, four synthetic datasets T5N2D20K, T5N4D40K, T5N8D80K, and T5N10D100K are considered. These datasets are different from each other not only in terms of number of transactions, but also in their total number of items. The algorithm is run several times using different threshold values, and the execution time of each run is measured and results are displayed in Figure 3. Also, Table 7 shows the number of extracted SLIs with different length in all datasets for *Mincover*=3 and $\gamma = 0.01$.

As depicted in Figure 3, values of *Minimum support* and *Mincover* have significant impact on execution time of the algorithm. An increase in the value of *Mincover*, even by one unit, causes a considerable decrease in the execution time. Particularly, when the value of *Minimum support* is low, a small change results in a drastic decrease in execution time. As the value of *Minimum support* gradually increases, any growth or decline in the value of *Mincover* does not lead to substantial exceution time changes. Perhaps the total number of SLIs and consequently, the number of generated candidates could play a role in the execution time as well. In order to investigate this relationship, an experiment is conducted to compare the number of SLIs and generated candidates for different *Mincover* and *Minimum support* values on the same dataset. Figure 4 and Figure 5 illustrate the outcome of this experiment run on dataset T5N2D20K. Looking at the results, a conclusion similar to what is seen in the previous experiment can be derived: any decrease either in *Minimum support* threshold or in *Mincover* value results in an increase in both the extracted SLIs number as well as generated candidates. As expected, there is a direct relation between the execution time on one hand and the number of SLIs and generated candidates on the other hand. Thus, determining these thresholds in the proposed algorithm is crucial.

$\gamma = 0.01 \quad Mincover = 3$					
	T5N2D20K	T5N4D40K	T5N6D60K	T5N8D80K	T5N10D100K
SLI1	9854	10550	9689	9019	8732
SLI2	14	32	77	116	141
SLI3	3	6	31	37	34
SLI4	0	1	10	6	5
SLI5	0	0	1	0	0

TABLE 7. Number of SLIs with Various Length Discovered in Datasets

4.4. **Scalability.** Our next experiment examines the relationship between the size of a dataset (or the number of transactions) and execution time of FSLIM algorithm. To do so, the algorithm is executed on T5N50D500K dataset containing 500,000 transactions with the average length of 5 and 50,000 items. It is done in the following order: initially, the number of transactions is considered 100,000 and the proposed algorithm is run with different values of *Minimum support* ($\gamma = 0.01, 0.02, \ldots, 0.05$) and *Mincover* (Mincover = 4, 5). Then, each time 50,000 transactions are added to the experiment and the execution time is recorded. The process continues up to 500,000 transactions and the results are presented in Figure 6.

For better demonstrating the result, Figure 7 displays the execution time for *Mincover*=5 and various *Minimum support*s. As the figures show, the execution time of FSLIM gradually (linearly) increases with the number of transactions. In Figure 7, each real graph is accompanied with its corresponding linear approximation. Hence, one can conclude that FSLIM still can be used on very large datasets and it is feasible; although, its execution time needs to be amended.

4.5. **Comparing FSLIM with Traditional Algorithms.** As this problem is new in its basic definition, it is difficult to compare FSLIM with other frequent itemset mining approaches. Most of the traditional algorithms extract frequent

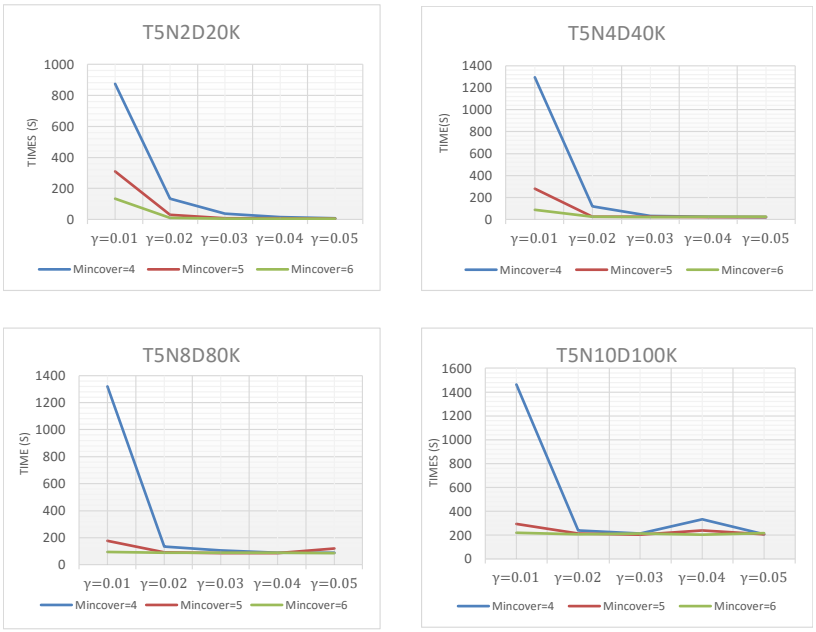


FIGURE 3. Execution Time Along with Various Mincover and Minimum Support

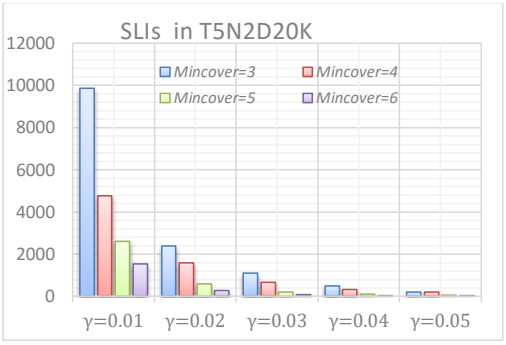


FIGURE 4. Number of SLIs with Various Mincover and Minimum Support

itemsets within the entire datasets. Alternatively, few mining algorithms that deal with frequent itemsets within time intervals (such as [24, 25, 26, 27]) are only applicable to binary datasets and not the quantitative ones. For evaluating the efficiency of FSLIM in this study, it is compared with two traditional algorithms (FAR and FTARM). FAR [9] proposes an approach to discover frequent patterns

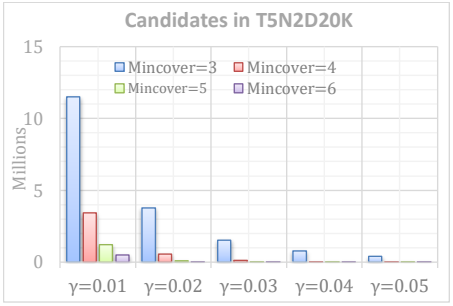


FIGURE 5. Number of Generated Candidates with Various Mincover and Minimum

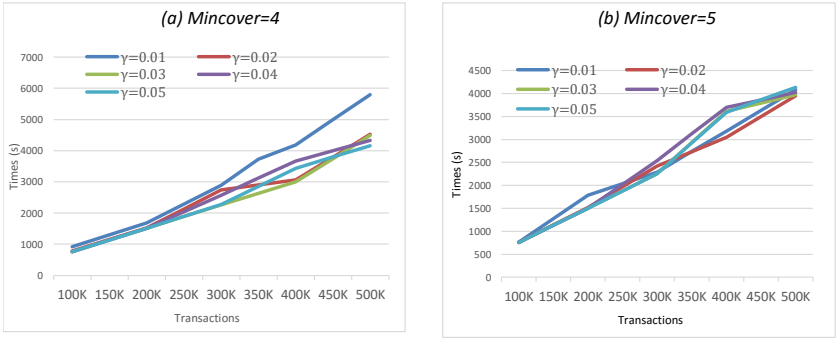


FIGURE 6. Scalability of FSLIM with various Minimum supports and (a) Mincover=4, (b) Mincover=5

in quantitative datasets regardless of the time, i.e. the extracted frequent itemsets are frequent over the entire dataset. FTARM [6] extracts frequent patterns based on lifespan of each item in a temporal quantitative transaction dataset. In this approach, the lifespan of each item is considered a period of time which starts from the first appearance of the item to the end of dataset. In this section, FSLIM is compared with FAR and FTARM based on accuracy in extraction of knowledge and the execution time.

The first experiment is made to compare total number of frequent itemsets extracted using FSLIM and the other approaches. This experiment is conducted on a synthetic dataset (T5N064D10K) consisting of 64 items and 10,000 transactions with average length of 5. In this experiment, the value of *Mincover* is considered 4 and *Minimum support* starts from a low value of $\gamma = 0.005$ and increases gradually. As it can be seen in Figure 8, due to lifespan of items in FTARM, the number of frequent patterns discovered by FTARM is higher than those of FAR. However, the

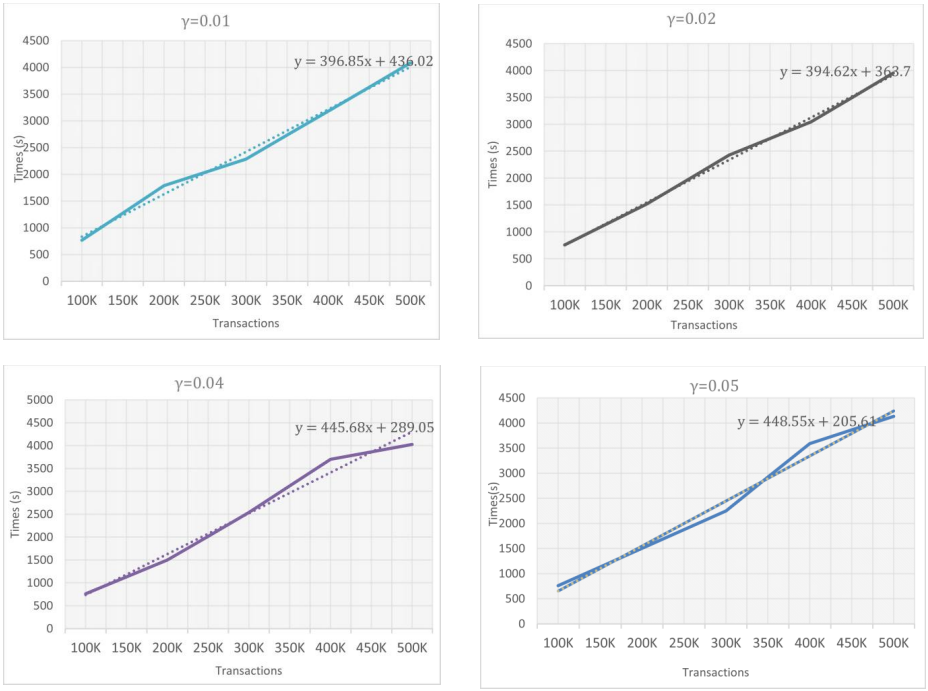


FIGURE 7. Execution time of FSLIM with various number of transactions with Mincover=5

number of SLIs extracted using FSLIM is considerably higher than both of these approaches.

Furthermore, it is important to note that the information extracted using FSLIM, is more detailed as it also contains time intervals within which each itemset occurs frequently. For providing an example, few SLIs extracted by FSLIM with $\gamma = 0.035$ are presented in Table 8. In order to make a better comparison between these three algorithms, the *support* of these itemsets in FAR and FTARM are computed and presented in the table as well. Column r_{first} used in FTARM algorithm, presents the first appearance of itemset in the dataset. As it can be seen, the *support* of itemsets extracted in these two algorithms is much lower than the *support* in FSLIM. In reality neither FAR nor FTARM algorithms could be effective for the high amount of *Minimum supports*. For example, as it is displayed in Figure 8, for any *Minimum supports* greater than 0.1, no frequent itemset is extracted using FAR and FRARM. This is due to the fact that *support* of each itemset in these algorithms is calculated based on the total number of transactions in the dataset. In both of these algorithms, the *support* of itemsets is dependent on the size of dataset, while in FSLIM, SLIs are determined locally and the size of dataset does not play any role in determining the *support* of itemsets. This problem is obviously intensified

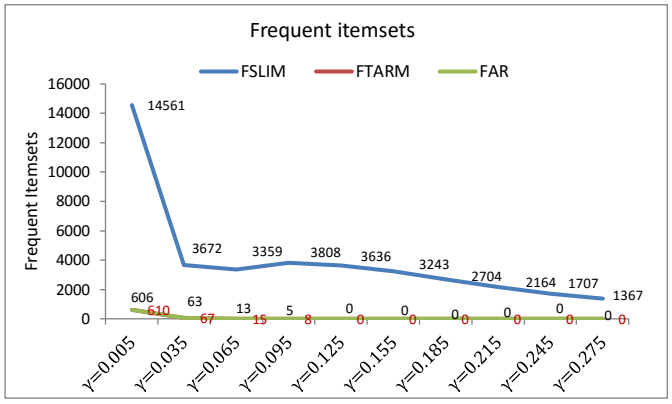


FIGURE 8. Comparing Number of Extracted Frequent Itemsets with Algorithms

in very large datasets. Hence, in order to better comparing these algorithms, the same experiment was done on a dataset consisting of only 10,000 transactions (such as [6]).

Itemset	r_{firat}	FSLIM		FAR	FTARM
		Time Interval	support	support	support
Item10.low	11	[302,542]	0.03568	0.00810	0.00813
Item41.Mid	85	[612,677]	0.07575	0.01088	0.01796
(Item11.Mid , Item36.High)	22	[3351,3387]	0.04864	0.00582	0.00583
		[7305,7334]	0.06000		
(Item14.Low , Item3.Mid)	151	[1481,1524]	0.05	0.00338	0.00344
		[4837,4883]	0.04255		
(Item28.Low , Item38.Mid)	22	[1588,1652]	0.07692	0.00767	0.00768
		[8114,8163]	0.03870		
		[8114,8163]	0.04		
(Item29.Mid , Item63.High)	31	[2857,2906]	0.044	0.00613	0.00615
(Item40.High , Item48.High)	2	[4817,4936]	0.05172	0.00452	0.00452
(Item49.High, Item63.Mid)	111	[7767,7812]	0.10144	0.00312	0.00316
(Item52.High, Item8.High)	90	[6747,6823]	0.03896	0.00215	0.00217
(Item62.Mid, Item8.Mid)	518	[785,846]	0.03548	0.00224	0.00239
(Item63.High , Item8.Mid)	214	[801,851]	0.03529	0.00684	0.00701
		[3259,3317]	0.04067		
		[4182,4227]	0.05217		
(Item17.High, Item28.Low, Item38.Mid)	1349	[1588,1601]	0.14285	0.00066	0.00078
(Item36.High, Item61.Mid, Item63.Mid)	459	[4362,4408]	0.04255	0.00094	0.00100

TABLE 8. Comparing the Support of Few Extracted SLIs with Two Other Algorithms

In order to comparing the response times of FSLIM with the other two algorithms, their execution times are presented in Figure 9. As it is shown, there is not a huge difference between the execution time of FAR and FTARM. Nevertheless, the execution time of FSLIM for the all Minimum support values, as it is expected, is higher than the two other approaches. There are two reasons for this difference: first, the total number of discovered frequent patterns in FSLIM is much higher than those in FAR and FTARM, because FSLIM is able to extract frequent itemset even in smaller time intervals. Second, FSLIM algorithm is more advanced or

complex in comparison to FAR and FTARM. For instance it contains additional functions such as finding and merging kernels of each candidate which results in extracting more detailed knowledge.

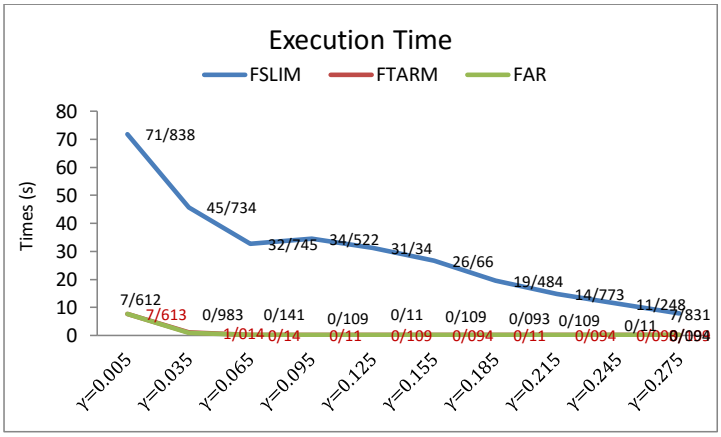


FIGURE 9. Comparing the Execution Time of Algorithms

4.6. **Real Dataset.** Mashad Traffic Database (MTD) provided by Traffic Organization of Mashad contains datasets representing traffic conditions in the junctions of Mashad city. To determine the amount of traffic, SCATSs(Sydney Coordinated Adaptive Traffic System) are installed in each junction and the number of vehicles passed in each 15 minutes are counted by them. This experiment deals with the MTD105 dataset representing traffic data for the junction of Kolahdooz Blvd and Abkooh Street. As it is shown in Figure 10, 11 SCATS are installed in this cross. MTD105 has 4224 transactions and recorded during 46 days from 00:00 8th Nov 2014 to 23:45 24th Dec 2014. Table 9 shows a sample of transactions from the MTD105 dataset.

Each transaction contains a Time item and 11 quantitative items(Si) containing the represented value by SCATSs. At first, with regard to the range of each item, five linguistic terms (*Very Low, Low, Mid, High, Very High*) are defined separately for each quantitative item.

In this experiment, the value of *Mincover* and *Minimum support* are considered 10 and 0.8 respectively. Table 9 represents some SLIs extracted from MTD105 dataset. As in the case of the traffic, the routs with dense traffic is important, in the table 10 some SLI examples in which the traffic is "*High*" or "*Very High*" are presented. For example, in the date 2014/12/24 in the time span of [15:15, 17:30] the traffic in the routes controlled by S2 is *High* and the ones controlled by S7 are *Very High*. As it is seen, while the traditional algorithms are unable to present such information, this algorithm, besides indicating the routes with dense traffic, it indicates the time span of high traffic, too. By obtaining such information, the

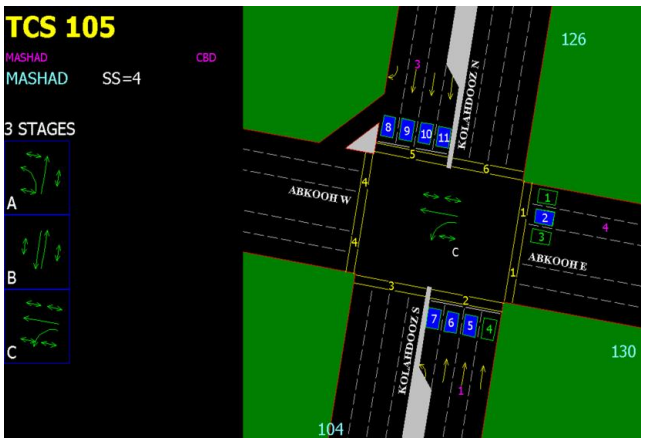


FIGURE 10. Cross of Kolahdooz Blvd and Abkooh Street

Time	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11
...	...	...	...	...	...	...	...	...	...	...	...
2014/11/28 8:30	6	18	14	5	17	16	5	13	21	0	14
2014/11/28 8:45	9	27	14	2	15	26	8	11	14	23	16
2014/11/28 9:00	7	26	15	6	14	15	7	8	22	11	24
2014/11/28 9:15	10	39	18	3	16	18	8	11	18	18	18
2014/11/28 9:30	13	34	33	4	26	26	15	13	19	22	17
2014/11/28 9:45	14	41	35	7	26	22	9	8	18	24	20
2014/11/28 10:00	14	29	25	6	11	14	12	8	14	22	18
2014/11/28 10:15	13	28	22	6	23	24	15	16	24	28	27
2014/11/28 10:30	12	38	28	6	27	32	10	17	17	30	18
2014/11/28 10:45	16	39	35	8	25	25	16	12	19	18	19
2014/11/28 11:00	11	54	41	4	27	31	11	12	20	26	18
2014/11/28 1:15	18	40	37	4	23	24	14	11	22	22	20
2014/11/28 11:30	20	46	47	2	35	37	10	20	21	25	27
2014/11/28 11 :45	17	42	41	5	22	20	14	16	23	29	29
...	...	...	...	...	...	...	...	...	...	...	...

TABLE 9. Some Transactions from MTD105 Dataset

directors will be able to provide more precise traffic plans and find the efficient ways of lessening the traffic density in the city.

5. **Conclusion and Future Works**

In this study, a novel mining approach, called Fuzzy Solid Linguistic Itemset Mining (FSLIM) is proposed. This algorithm is the first attempt to extract all SLIs from a temporal quantitative transaction dataset. SLI refers to the linguistic

Itemset	Time interval	support
(S10.High, S6.High)	2014/12/24 [10:30 ,13:15]	0.82432
(S1.High, S2.High)	2014/12/24 [11:00, 13:15]	0.82958
(S1.High,S9.High)	2014/10/5 [12:15, 14:30]	0.84204
(S2.High, S7.Very High)	2014/12/24 [15:15, 17:30]	0.80580
S1.Very High	2014/11/10 [12:45, 15:00]	0.86667
	2014/12/15 [12:30, 15:00]	0.97980
	2014/12/23 [12:15, 14:45]	0.99091
S3.Very High	2014/11/8 [09:30, 15:00]	0.99510
	2014/11/15 [11:30, 15:00]	0.80222
	2014/12/4 [17:30, 22:45]	0.82879
	2014/12/24 [19:00, 22:30]	0.93074

TABLE 10. Some SLIs Discovered in MTD105 Dataset

frequent itemset within a temporally contiguous subset or time interval of a dataset. The size of each time interval is not known in advance.

As part of the research, several test experiments are conducted on synthetic and real datasets to compare overall performance of this method with traditional algorithms and to demonstrate its actual efficiency. Based on the experimental results, FSLIM is able to discover SLIs in very large datasets and the execution time of this algorithm is linearly proportional to the number of candidates and extracted SLIs. In comparison with the traditional algorithms, FSLIM shows higher accuracy in finding hidden knowledge that cannot be extracted using traditional methods.

Furthermore, all findings confirm that the two thresholds *Mincover* and *Minimum support* play a critical role not only in discovering SLIs, but also in the execution time and number of candidates generated. In the future, we hope to determine these thresholds using evolutionary approaches. We also hope to investigate other algorithms, in place of *apriori* to further increase the efficiency of extraction method. Finally, we assume here that only one transaction occurs at each time stamp. For further research in the future, we would like to propose an approach to discover SLIs in a dataset that consists of multiple transactions at each time stamp.

REFERENCES

[1] R. Agrawal, T. Imieliski and A. Swami, *Mining association rules between sets of items in large databases*, In: ACM SIGMOD Rec., ACM, (1993), 207–216.

[2] J. M. Ale and G. H. Rossi, *An approach to discovering temporal association rules*, In: Proc. 2000 ACM Symp. Appl. Comput.-Vol. 1, ACM, (2000), 294–300.

[3] R. Agrawal and R. Srikant, *Others, Fast algorithms for mining association rules*, In: Proc 20th Int Conf Very Large Data Bases VLDB, (1994), 487–499.

[4] X. Chen and I. Petrounias, *Discovering temporal association rules: Algorithms, language and system*, In: 16th Int. Conf. Data Eng. ICDE, IEEE Computer Society, San Diego, CA, (2000), 306-306.

[5] C. Y. Chang, M. S. Chen and C. H. Lee, *Mining general temporal association rules for items with different exhibition periods*, In: Data Min. 2002 ICDM 2003 Proc. 2002 IEEE Int. Conf. On, IEEE, (2002), 59–66.
[6] C. H. Chen, T. P. Hong and S. B. Lin, *Mining fuzzy temporal knowledge from quantitative transactions*, In: Syst. Sci. Eng. ICSSE 2011 Int. Conf. On, IEEE, (2011), 405–409.
[7] J. Han, J. Pei, Y. Yin and R. Mao, *Mining frequent patterns without candidate generation: A frequent-pattern tree approach*, Data Min. Knowl. Discov., **8** (2004), 53–87.
[8] T. P. Hong, Y. Y. Wu and S. L. Wang,*An effective mining approach for up-to-date patterns*, Expert Syst. Appl., **36** (2009), 9747–9752.
[9] T. P. Hong, C. S. Kuo and S. C. Chi, *Trade-off between computation time and number of rules for fuzzy mining from quantitative data*, Int. J. Uncertain. Fuzziness Knowl.-Based Syst., **9** (2001), 587–604.
[10] J. W. Huang, B. R. Dai and M. S. Chen, *Twain: Two-end association miner with precise frequent exhibition periods*, ACM Trans. Knowl. Discov. Data TKDD., **1** (2007), 8.
[11] H. Ishibuchi and T. Yamamoto, *Rule weight specification in fuzzy rule-based classification systems*, Fuzzy Syst. IEEE Trans. On., **13** (2005), 428–435.
[12] G. C. Lan, C. H. Chen, T. P. Hong and S. B. Lin, *A fuzzy approach for mining general temporal association rules in a publication database*, In: Hybrid Intell. Syst. HIS 2011 11th Int. Conf. On, IEEE, (2011), 611–615.
[13] C. H. Lee, M. S. Chen and C. R. Lin, *Progressive partition miner: an efficient algorithm for mining general temporal association rules*, Knowl. Data Eng. IEEE Trans. On., **15** (2003), 1004–1017.
[14] W. J. Lee and S. J. Lee, *Discovery of fuzzy temporal association rules*, Syst. Man Cybern. Part B Cybern. IEEE Trans. On., **34** (2004), 2330–2342.
[15] W. J. Lee, J. Y. Jiang and S. J. Lee, *Mining fuzzy periodic association rules*, Data Knowl. Eng., **65** (2008), 442–462.
[16] Y. Li, P. Ning, X. S. Wang and S. Jajodia, *Discovering calendar-based temporal association rules*, Data Knowl. Eng., **44** (2003), 193–218.
[17] C. W. Lin and T. P. Hong, *Temporal data mining with up-to-date pattern trees*, Expert Syst. Appl., **38** (2011), 15143–15150.
[18] S. G. Matthews, M. A. Gongora and A. A. Hopgood, *Evolving temporal association rules with genetic algorithms*, In: Res. Dev. Intell. Syst. XXVII, Springer, (2011), 107–120.
[19] S. G. Matthews, M. A. Gongora, A. A. Hopgood and S. Ahmadi, *Web usage mining with evolutionary extraction of temporal fuzzy association rules*, Knowl.-Based Syst., **54** (2013), 66–72.
[20] S. G. Matthews, M. A. Gongora and A. A. Hopgood, *Evolutionary algorithms and fuzzy sets for discovering temporal rules*, Int. J. Appl. Math. Comput. Sci., **23** (2013), 855–868.
[21] S. G. Matthews, M. A. Gongora and A. A. Hopgood, *Evolving temporal fuzzy itemsets from quantitative data with a multi-objective evolutionary algorithm*, In: Genet. Evol. Fuzzy Syst. GEFS 2011 IEEE 5th Int. Workshop On, IEEE, (2011), 9–16.
[22] J. S. Park, M. S. Chen and P. S. Yu, *Using a hash-based method with transaction trimming for mining association rules*, Knowl. Data Eng. IEEE Trans. On., **9** (1997), 813–825.
[23] J. Pei, J. Han, H. Lu, S. Nishio, S. Tang and D. Yang, *H-mine: Hyper-structure mining of frequent patterns in large databases*, In: Data Min. 2001 ICDM 2001 Proc. IEEE Int. Conf. On, IEEE, (2001), 441–448.
[24] B. Saleh and F. Masseglia, *Discovering frequent behaviors: time is an essential element of the context*, Knowl. Inf. Syst., **28** (2011), 311–331.
[25] S. Suvvari and R. B. V. Subramanyam, *An efficient approach for significant time intervals of frequent itemsets*, Int. J. Intell. Syst. Technol. Appl., **13** (2014), 222–243.
[26] Y. Xiao, R. Zhang and I. Kaku, *A new framework of mining association rules with time-windows on real-time transaction database*, Int. J. Innov. Comput. Inf. Control., **7** (2011), 3239–3253.

[27] Y. Xiao, Y. Tian and Q. Zhao, *Optimizing frequent time-window selection for association rules mining in a temporal database using a variable neighbourhood search*, Comput. Oper. Res., **52** (2014), 241–250.
[28] J. S. Yoo and S. Shekhar, *Similarity-profiled temporal association mining*, Knowl. Data Eng. IEEE Trans. On., **21** (2009), 1147–1161.
[29] L. A. Zadeh, *Fuzzy sets*, Inf. Control., **8** (1965), 338–353.
[30] C. Zhuo, L. Jiahui and L. Chen, *A fuzzy calendar-based algorithm for mining temporal association rules and its application*, In: Fuzzy Syst. Knowl. Discov. 2009 FSKD09 Sixth Int. Conf. On, IEEE, (2009), 28–33.

Mahnaz Kadkhoda, Department of Computer Engineering, Center of Excellence on Soft Computing and Intelligent Information Processing, Ferdowsi University of Mashhad, Mashhad, Iran
E-mail address: ma.kadkhoda@stu-mail.um.ac.ir

Mohammad-R. Akbarzadeh-T*, Department of Computer Engineering, Center of Excellence on Soft Computing and Intelligent Information Processing, Ferdowsi University of Mashhad, Mashhad, Iran
E-mail address: akbazar@um.ac.ir

S. Mahmoud Taheri, Faculty of Engineering Science, College of Engineering, University of Tehran, Tehran, Iran
E-mail address: sm_taheri@ut.ac.ir

∗Corresponding author