International Journal of Engineering Trends and Technology
ISSN: 2231 – 5381 /doi:10.14445/22315381/IJETT-V69I4P231

Volume 69 Issue 4, 228-235, April 2021

High Utility-Occupancy Sequential Pattern Mining Algorithm Based On Utility-Occupancy Framework

Saritha Vemulapalli [1, 2, *], Shashi Mogalla [2]

[1] *Dept. of Information Technology, VNR Vignana Jyothi Inst. Of Engg. & Tech., Hyderabad, Telangana, India*
[2] *Dept. of Computer Science & Systems Engineering, Andhra University College of Engineering(A), Visakhapatnam, Andhra Pradesh, India*

[*] saritha_v@vnrvjiet.in, smogalla2000@yahoo.com

Abstract *- Classical sequential pattern mining (SPM) algorithms can not generate patterns that are interesting and potentially useful in all real-world applications due to equal significance for all the items and using only frequency as an interestingness measure. Some real-world applications involve items of different nature, whose significance is measured using different criteria such as utility, risk, profit, weight, time duration, etc. In addition to the utility of items constituting a pattern, the significance of a pattern is also influenced by its occupancy in its supporting sequences. To deal with the above problems, we propose a variant of SPM called high utility-occupancy sequential pattern mining (HUOSPM) to discover more interesting, potentially useful, and dominant patterns. In this paper, the authors devised two compact data structures called seqlist to represent information about each sequence of the quantitative sequence dataset and uolist to maintain candidate patterns information. The authors proposed a novel utility-occupancy framework based HUOSPM algorithm, which discovers the patterns using seqlist and uolist. The authors also proposed search space pruning strategies called pattern extension utility-occupancy, reduced sequence utility-occupancy, and extension upper bound utility-occupancy. Experimentation was carried out on real datasets with varying support threshold and utility occupancy threshold to evaluate the quality of patterns. It is observed from results that the patterns generated by our proposed HUOSPM algorithm are qualitative compared to baseline algorithm prefix span and also it can completely discover HUOSP's.*

Keywords — *Data mining, Pattern discovery, Pattern-Growth, Utility-Occupancy, Variant sequential patterns.*

I. INTRODUCTION

Discovering interesting and potentially useful patterns from diversified datasets is important for decision making in many real world applications. Classical Sequential pattern mining (SPM) intend to find frequent patterns represents time-ordered behavior. Sequential patterns constitute frequently appearing ordered events, wherein each event is made up of an item set, which represents interdependencies exist between the items belonging to events that make up. Initially SPM was implemented in [1], plays an essential role in solving real-world problems in varied domains like web usage mining, retail sector, stock exchange, e-commerce and bioinformatics, etc are of kind time related behavior. For example diagnosing disease based on the chronological order of medical symptoms of a patient, identifying customers buying behavior based on transaction data of retail store, finding telephone call patterns from users telephone calling records, finding weather patterns from weather dataset to predict climate condition (weather forecasting) and finding users navigational patterns from click-stream data for better administration of website.

Classical SPM algorithms discover complete set of patterns by considering all items with equal significance using only frequency as interestingness measure. However some real world applications involve items of different nature and hence the significance of items is assessed in different perspectives measured using various criteria such as utility, risk, profit, weight, time duration, etc. Hence patterns generated by classical SPM algorithms may not be interesting and potentially useful for the users in all contexts. To address the above issue a variant of SPM called High utility sequential pattern mining (HUSPM) [2, 3, 4, 5, 6, 7, 8] is developed to extract patterns of contextual significance by extending the sequence dataset with item significance as an additional characteristic for each item depending on the context. These patterns are the subset of sequential patterns set having utility greater than utility threshold. For example, in supermarket dataset, the significance of an item is measured in terms of the profit earned on it, which increases with the quantity of the item sold and accordingly the significance of the item varies for different occurrences of the item in a sequence. In case of web-usage dataset, the time spent on a web page represents the significance of that web page.

In addition to the utility of items constituting a pattern, the significance of a pattern is also influenced by its occupancy in its supporting sequences. Occupancy implies the proportion of the length of the pattern in their supporting sequences. In many real-world applications, the completeness of the pattern is measured using occupancy. In general sequential patterns have low occupancy, which implies they represent a small portion in their supporting

sequences. To address the above issue variant of SPM called high occupancy sequential pattern mining (HOSPM) [9] is proposed to find out more dominant patterns. However, existing HOSPM algorithms ignore item utility, and on the contrary, HUSPM algorithms ignore pattern occupancy. Hence, to address the above problems, the authors propose a utility-occupancy framework based *HUOSPM* algorithm to discover HUOSP's from a sequence dataset extended with utility information. The proposed algorithm measures the pattern interestingness using both support threshold and utility-occupancy threshold. The structure of the remaining paper is described below. A brief overview of related variant sequential pattern mining algorithms is presented in section II. The problem statement is discussed in section III. THE proposed *HUOSPM* algorithm, notations, and definitions are discussed in section IV. Experimentation is discussed in section V. Section VI encloses conclusions.

II. Related Work

A. Review on HUSPM

Classical SPM algorithms produce patterns by considering equal significance for all the items and frequency as an interesting measure. In some applications of the real world, every item has different nature; hence the significance of items is different and is measured using various criteria such as utility, risk, profit, weight, interest, time, etc. Hence patterns generated by traditional algorithms may not be interesting and potentially useful for the user. To address the aforementioned issue, a variant of SPM called HUSPM is proposed [2, 3, 4, 5, 6, 7, 8] to discover the more interesting, potentially useful patterns. UtilityLevel [2] algorithm discovers utility patterns using candidate generate and test approach, and UtilitySpan [2] algorithm is based on pattern growth strategy. Both the algorithms support downward closure property using *swu*. UtilityLevel [2] algorithm generates high *swu* level-k candidate patterns by joining with level-(k-1) high utility patterns. First, it generates the length-1 candidate patterns having high *swu* value, and then subsequently generates length-2 candidate patterns by joining length-1 high utility patterns and so on until no more candidate patterns are generated with high *swu* value. At each level, candidate patterns are tested by scanning the dataset once to generate high utility sequential patterns. It suffers from more number of candidate generation and dataset scans. To overcome this drawback, a pattern-growth-based strategy is used in UtilitySpan [2] algorithm, which recursively divides the search space using the divide-and-conquer method. The UtilitySpan algorithm read the dataset once to find length-1 high utility patterns. Consequently, it constructs projected databases for length-1 high utility patterns. Then, recursively explore high utility larger patterns by scanning the projected database once using the pattern-growth approach. Therefore, it reduces the runtime significantly. In USpan[3], the authors proposed a generic framework and algorithm for finding HUSP's. In USpan, the search space is represented using *LQS-tree*. It generates candidate patterns by extending the pattern with item extension or sequence extension while exploring the LQS-tree using a depth-first-search strategy. *SWU* measure and *SWDC* properties are used for pruning unpromising sequences in order to improve the mining performance. Utility matrix is used in USpan for data representation, which is complex and memory-consuming. In PHUS [4], the authors proposed a projection-based HUSPM algorithm. It improves the performance by candidate pruning, using projection-based maximum utility measure and *SUUB* measure in generating tight upper bounds. In HuspExt [5], the authors proposed an algorithm using an upper bound called *CRoM*. It prunes unpromising sequences using *PBCG* strategy. However, due to its incorrect upper bound, it can't identify the complete HUSP's. Hence, the author proposed HUS-Span [6] algorithm to find the complete HUSPs using tight utility upper bound measures called *PEU* and *RSU* of prefix-pattern. In HUSP-ULL [7], the authors proposed an algorithm using data structure *UL-list* and look-ahead and irrelevant item pruning strategies for pruning unpromising sequences. The authors proved that the execution time of the HUSP-ULL algorithm is less when compared to the above HUSPM algorithms. In AHUS [8], authors devised array data structure and also pruning mechanism for efficiently pruning non-HUSPs. The author's also proposed AHUS-P [8] algorithm for identifying HUSP's concurrently. The experimental results proved that both the algorithms outperform the HUS-Span with respect to mining time and memory used.

B. Review on HOSPM

However, HUSPM algorithms can discover the more interesting and potentially useful HUSP's. In the majority of real-world applications, the completeness of the pattern is measured using occupancy. In general sequential patterns have low occupancy, which implies they represent a small portion in their supporting sequences. Hence, sequential patterns may not be interesting and potentially useful for the user. To address the aforementioned issue variant of SPM called HOSPM is proposed to discover the more dominant patterns. In DOFRA[9], the authors proposed an algorithm for finding qualified sequential patterns using occupancy-based measures for reducing the search space. It finds the qualified patterns by computing the occupancy of a pattern using support and occupancy thresholds. The algorithm also discovers top-k qualified sequential patterns, which satisfy support and occupancy thresholds.

C. Review on HUOPM

However, HUSPM algorithms can discover the more interesting and potentially useful HUSP's. In the majority of real-world applications, the completeness of the pattern is measured using occupancy. In general, utility sequential patterns have low occupancy, which implies they represent a small portion in their supporting sequences. Even though

HOSPM algorithms can identify the more dominant patterns, but they don't consider item significance into account. Hence, HUSP's and HOSP's may not be interesting and potentially useful for the user. To address both the problems listed above, we propose a variant of SPM called HUOSPM to discover the more interesting, potentially useful, and dominant patterns. The existing HUOPM algorithms [10, 11] are designed for non-sequential datasets (i.e., transaction datasets), where the order of elements is not maintained.

OCEAN [10] algorithm discovers high utility occupancy patterns from transaction datasets. However, it generates incomplete patterns due to the inappropriate computation of utility information, and also, it is not efficient due to inappropriate utilization of the support and utility-occupancy thresholds. In HUOPM [11], the authors proposed an algorithm, compact data structures FU-tree and UO-list. HUOPM algorithm effectively discovers high utility occupancy patterns from transaction datasets using FU-tree and UO-list without generating candidates. The authors proved that it outperforms the OCEAN algorithm.

III. Problem Statement

Let QSD = $\{S_1, S_2, .., S_i, .., S_m\}$ is a quantitative sequence dataset, where each sequence S_i is a sequence of time-ordered quantitative events expressed as $< e_1\ e_2\ ..e_i..\ e_k >$ and each event e_i is a quantitative itemset containing items associated with utility (i.e., e_i is a subset of I). Let I = $\{it_1, it_2, \ldots, it_n\}$ be finite set of distinct items. Items that belong to each item set are organized in lexicographic order. Given the sequences $p =< x_1\ x_2 \ldots x_n >$ and $q =< y_1\ y_2 \ldots y_m >$, then p is a subsequence of q and q contains p denoted as $p \subseteq q$, if $\exists\ 1 \le j_1 < j_2 < \ldots < j_n \le m \in Z$: $x_1 \subseteq y_{j1}$, $x_2 \subseteq y_{j2}, \ldots$, and $x_n \subseteq y_{jn}$. If the sequence 's'= $< e_1\ e_2 \ldots e_j \ldots e_k >$ contains L number of items, then it is called L-length sequence. The sequence length is defined as L =|s|, where $|s| = \Sigma^k_{j=1}|e_j|$. If the sequence 's'= $< e_1\ e_2 \ldots e_j \ldots e_k >$ contains k number of events, then the size of sequence is defined as k. The count of sequences that contain pattern p determines the support of p is denoted as sup(p). Utility refers to the usefulness of an item in terms of time duration, profit, weight, etc. For a given support threshold denoted by α ($0 < \alpha \le 1$) and utility-occupancy threshold denoted by ß ($0 < ß \le 1$), HUOSPM identify interesting patterns from the sequential dataset with utility information that holds a significant amount of support and utility-occupancy by considering item weight, time duration, profit, risk, etc.

Let I = {a,b,c,d,e,f,g} denote the set of distinct items. The *QSD* shown in Table 1 has 4 item set sequences, wherein each item has the quantity associated with it, representing the utility of an item that may vary with the event. For example the third sequence (b[35]c[15]) (a[25]b[45]c[30] consists of three events e1 = (b[35]c[15]) , e2 = (c[20]) and e3 = (a[25]b[45]c[30]). Utility of b is 35 and c is 15 in e1 while the utility of c is 20 in e2 and utility of b is 45, and c is 30 in e3. The pattern is said to occur in a sequence at an index position based on the last matching item of the pattern in the sequence. In sequence 3, pattern (b)(c) occurred at two positions as its last symbol c is contained in 2nd and 3rd events.

Table I. A Sample Quantitative Sequence Dataset

(a[10]b[20])(b[25]c[50]d[10])(d[15]e[58])
(f[23] (a[30]b[40]c[25])(c[20]d[10])
(b[35]c[15])(c[20])(a[25]b[45]c[30])
(d[5]e[10])(a[18]g[10])(b[30]c[20]d[10])(f[10])

IV. Proposed Utility-Occupancy Framework based HUOSPM Algorithm, Notations, and Definitions

For a given support threshold denoted by α ($0 < \alpha \le 1$) and utility-occupancy threshold denoted by ß ($0 < ß \le 1$), the proposed High Utility-Occupancy Sequential pattern mining (HUOSPM) identify interesting patterns from the quantitative sequential dataset; the patterns extracted are in sequential order, with a high frequency of occurrence and high level of utility as well as occupancy. The algorithm explores the pattern space starting from singleton patterns and recursively extending the patterns either by item extension or sequence extension before applying α and ß thresholds to ensure that the extended pattern is eligible for further expansion. An extended pattern with an item appended to the last event of a given pattern sp is called an item extension of the prefix-pattern sp. Similarly, an extended pattern with an item appended as a separate new event to the last event of a given prefix-pattern sp is called sequence extension of the prefix-pattern sp. The complete details of our proposed algorithm are discussed below.

A. ArrayList data structure for representing quantitative sequence dataset

In this paper, the authors devised a compact data structure called *seqlist* to represent information about each sequence of the quantitative sequence dataset. Proposed *HUOSPM* algorithm uses the *seqlist* to keep track of details about items of a specific sequence, their utilities, and suffix sequence utilities for each sequence in the quantitative sequence dataset (QSD).

seqlist is a 2-D list containing 3 rows and a varying number of columns. The number of columns of *seqlist* for a sequence s is equal to the sum of the sequence size and sequence length representing the number of events in s. The first row, *slist* denotes a sequence list, which contains a list of items in sequence *s*. The second row, *ulist*, denotes a utility list, which contains utilities of items in *s*. In the third row, *rulist* denote a remaining utility list, which contains the remaining utilities of the suffix sequences starting at successive indexes in s. Events of a sequence are separated with the same delimiting item as '-1'.

Definition 1: The quantity associated with an item i occurred at index k in sequence s is the utility of i at k in s [8], denoted as u(i, k, s) = { q(i, k, s) }.

Definition 2: The sum of utilities of an item i occurred at

different indices in sequence s is the utility of i in s [8], denoted as u(i, s) = { Σ u(i, k, s) ∀ index k of i in s }.

Definition 3: The sequence utility is defined as the sum of utilities of all items i in sequence s [8], denoted as u(s) = { Σ u(i, s) $_{\forall}$ i ∈ s }.

Definition 4: The utility of a pattern sp occurred at index k in sequence s is defined as the sum of utilities of all items i in sp at index j in s [8], denoted as u(sp, k, s) = {Σ u(i, j, s) $_{\forall}$ i ∈ sp }.

Definition 5: The utility-occupancy of a pattern sp occurred at index k in sequence s is defined as the utility of sp occurred at k divided by the utility of s, denoted as uo(sp, k, s) = u(sp, k, s) / u(s).

Definition 6: The utility-occupancy of a pattern sp in its supporting sequence s is defined as the maximum utility-occupancy of sp in s, denoted as uo (sp, s) = max{ uo(sp, k, s) ∀ index k of sp in s }.

Definition 7: The utility-occupancy of a pattern sp in QSD is *the* ratio of the sum of utility-occupancy of sp in its supporting sequences divided by support count of sp, is denoted as uo(sp) = (Σ $_{sp\subseteq s \wedge s\in QSD}$ uo(sp, s))/ (sup(sp)).

Definition 8: For the specified support threshold denoted by α (0 < α ≤ 1) and utility-occupancy threshold denoted by ß (0 < ß ≤ 1), a pattern sp in QSD is called as HUOSP if sup(sp) ≥ α × |D| and uo(sp) ≥ β.

Definition 9: The remaining utility of an item occurred at index j in remaining utility list is defined as the sum of utilities of items at the index k > j in *ulist*, where j and k doesn't contain delimiting items (dlitem) [8]. It is mathematically expressed as rlist(s) [j] = Σ $_{k > j \wedge ulist(s)[k] != dlitem}$ ulist(s) [k].

B. ArrayList data structure for representing information about promising patterns

A promising pattern may appear at various indices in a sequence with different utility-occupancy values for each occurrence, and hence the pattern's growth needs to be explored at different positions of the sequence. The authors devised another compact data structure known as a *utility-occupancy list* (*uolist*) to keep necessary information about candidate patterns in a specific sequence. THE proposed *HUOSPM* algorithm uses an *uolist* to maintain the occurrence positions, utility-occupancy values, and pattern extension utility-occupancies information about the promising patterns. *HUOSPM* algorithm uses *uolist* of prefix-pattern to generate *uolist* of its extended pattern generated using sequence extension or item extension; thus, *HUOSPM* algorithm can avoid scanning of the complete dataset. *uolist* of pattern sp in sequence s, maintains the occurrence positions, its utility-occupancies, and pattern extension utility-occupancies at those occurrence positions in s. *uolist* is a 2-D list, *uolist[0]* denote occurrence positions of sp in s, *uolist[1]* denote utility-occupancies at those occurrence positions, and *uolist[2]* denote pattern extension utility-occupancies at those occurrence positions.

Definition 10: The utility-occupancy list of a pattern sp at index k in a sequence s is the list of occurrence position of sp, its utility-occupancy, and pattern extension utility-occupancy at that occurrence position k in s, denoted as uolist(sp, k, s).

Definition 11: The utility-occupancy list of a pattern sp in a sequence s is the list of all occurrence positions of sp, its utility-occupancies and pattern extension utility-occupancies at those occurrence positions in s, denoted as uolist(sp, s) = U{uolist(sp, k, s) ∀ index k of sp in s }.

Definition 12: The utility-occupancy list of a pattern sp in QSD is the collection of the utility-occupancy lists of sp in its supporting sequences of QDB, denoted as uolist(sp) = **U** $_{\forall s \in QSD}$ uolist(sp, s).

Definition 13: Consider a pattern sp and its descendant sp′ generated by extending sp with item extension or sequence extension. The utility-occupancy value of sp′ at index k in *s* is denoted and described as uolist(sp′, k, s) = max { uolist(sp, j, s) | ∀ index j of sp in s ∧ j < k} + (ulist(s)[k]/u(s)).

Definition 14: The prefix extension utility [6] of a pattern sp in a sequence s at index k is denoted and described as

peu (sp, k, s) = u(sp, k, s) + rlist(s)[k] , if rlist(s)[k] > 0.
0, otherwise.

Definition 15: The prefix extension utility-occupancy of a pattern sp in a sequence s at index k is denoted and described as peuo (sp, k, s) = peu (sp, k, s)/u(s).

Definition 16: The prefix extension utility-occupancy of a pattern sp in a sequence s is denoted and described as

peuo (sp, s) = max { peuo (sp, k, s) | ∀ index k of sp in s }.

Definition 17: The prefix extension utility-occupancy of a pattern sp in quantitative sequence dataset QSD is denoted and described as

peuo (sp) = (Σ $_{s \in QSD \wedge sp\subseteq s}$ peuo (sp, s)) /sup(sp).

Definition 18: The reduced sequence utility-occupancy of a pattern sp′ generated by extending a pattern sp with item extension or sequence extension in sequence s is denoted and described as

rsuo (sp′, s) = peuo(sp, s) if sp ⊆*s* ∧ sp′⊆*s*.
0, otherwise.

Definition 19: The reduced sequence utility-occupancy (rsuo) of a pattern sp in a given dataset is defined as

rsuo (sp) = (Σ $_{s \in QSD}$ rsuo (sp, s)) / sup(sp).

Definition 20: Extension upper bound utility-occupancy of a pattern sp′ generated by extending a pattern sp with an item i, denoted by euuo(sp′), is defined as euuo (sp′) = uo (sp) + peuo (i).

C. Proposed utility-occupancy framework-based approach for discovering HUOSP's

a) The abstract view of the proposed approach is explained below:

- Pre Process the quantitative dataset for sanitizing it.
  - Read the quantitative sequence dataset to calculate the support count for each distinct item i and find out frequent promising items j such that sup(j) >= α × |D|
  - Remove all unpromising items from the quantitative sequence dataset to sanitize it.
- Extract high utility-occupancy patterns from sanitized quantitative sequence dataset using the proposed HUOSPM algorithm.

b) Proposed HUOSPM algorithm

HUOSPM algorithm starts with initializing promising patterns with frequent promising items list, then constructs the *uolist* for each promising pattern sp. If the utility-occupancy of sp >= ß , then add sp to *HUOSPList*. For each promising one-item pattern in sp, the *HUOSPM* algorithm invokes the *HUOSP Growth* function. If pattern extension utility-occupancy of sp is less than ß, then the *HUOSPGrowth* function doesn't continue its execution further. On the other hand, the algorithm read the projected database (*uolist*) of sp to find possible promising items list *i-extlist* that occur after sp for the item extension and *s-extlist* that occur after sp for the sequence extension. The algorithm adds the items into *i-extlist, s-extlist* if *euuo* (definition 20) of generated pattern sp′ by extending sp with an item i is greater than ß, then the algorithm prune the remaining unpromising items from *i-extlist, s-extlist* using *rsuo* (definition 19). For each item i in *i-extlist*, the *HUOSPM* algorithm generates pattern sp′ by extending sp with an item i to from item extension and construct projected database (*uolist*) of sp′. If the utility-occupancy of sp′ >= ß (definition 7), then add sp′ to HUOSPList. Next, the HUOSPM algorithm recursively calls itself to explore the growth of sp′. Similarly, for each item i in *s-extlist*, the *HUOSPM* algorithm generates pattern sp′ by extending sp with an item i to from sequence extension and construct projected database (*uolist*) of sp′. If the utility-occupancy of sp′ >= ß (definition 7) , then add sp′ to HUOSPList. Next, the HUOSPM algorithm recursively calls itself to explore the growth of sp′.

The proposed HUOSPM algorithm's pseudo-code is described below.

Input: Sanitized QSD, frequent promising items list, α, and ß.
Output: High Utility-Occupancy Sequential Patterns (HUOSPList).

```
for each item sp ∈ frequent promising items list
   construct uolist(p)
        if uo (sp) ≥ ß then
        add sp to HUOSPList
        end if
   HUOPGrowth ( sp , uolist(sp), uo(sp) , peuo (sp))
end for
```

Method:

```
HUOSPGrowth (sp , uolist(sp ), uo(sp ) , peuo(sp))
begin
  if peuo (sp) < ß then return
  Scan the p-projected database and process each sequence as described below
  Generate sp′ by extending a pattern sp with a promising item i,
  if euuo (sp′) ≥ ß then // explore the growth of pattern sp
    identify items that occur after the pattern for the item extension to form i-extlist list
    identify items that occur after the pattern for the sequence extension to form s-extlist list
  end if
  for each item i ∈ i-extlist || s-extlist
    if (rsuo(i) < ß) then remove i from the respective list
  end for
  for each item i ∈ i-extlist
    sp′| ← item extension(sp,i)
    construct uolist(sp′)
    if sup(sp′) >= α then
      if uo (sp′) ≥ ß then
        add sp′ to HUOSPList
      end if
      HUOSPGrowth(sp′,uolist(sp′),uo(sp′),peuo(sp′)) //call
                    HUOSPGrowth on extended pattern sp′.
    end if
  end for
  for each item i ∈ s-extlist
    sp′ ← sequence extension(sp,i)
    construct uolist(sp′ )
    if sup(sp′) >= α then
      if uo (sp′ ) ≥ ß then
        add p^I to HUOSPList
      end if
      HUOSPGrowth(sp′,uolist(sp′),uo(sp′),peuo(sp′)) //call
                    HUOSPGrowth on extended pattern sp′.
    end if
  end for
end
```

c) Pruning strategy

Our proposed algorithm initially prune the search space of candidate promising patterns by calculating upper bound on the utility-occupancy using peuo (definition 17) and the support count, then efficiently prune the remaining search space by pruning sequence extension and item extension candidate items before constructing extended candidate promising patterns using euuo (definition 20), and then prune the remaining unpromising items using rsuo (definition 19).

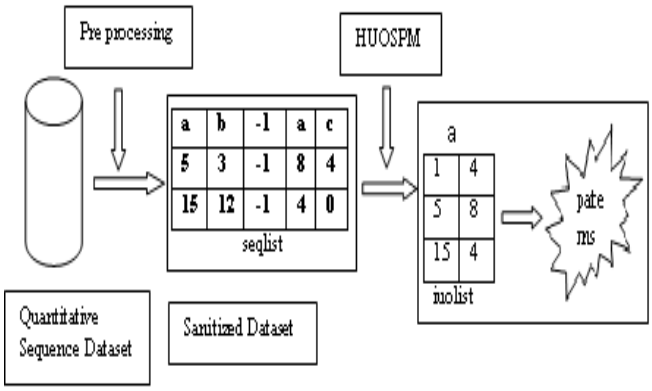


Fig. 1 Abstract view of steps constituting proposed approach

V. Experimentation

Our proposed HUOSPM algorithm is implemented in Java 8. Experimentation was done on a system with Intel(R)Xeon(R) processor E3-1225V5(3.30-GHz), 32GB RAM operating on Windows 10.

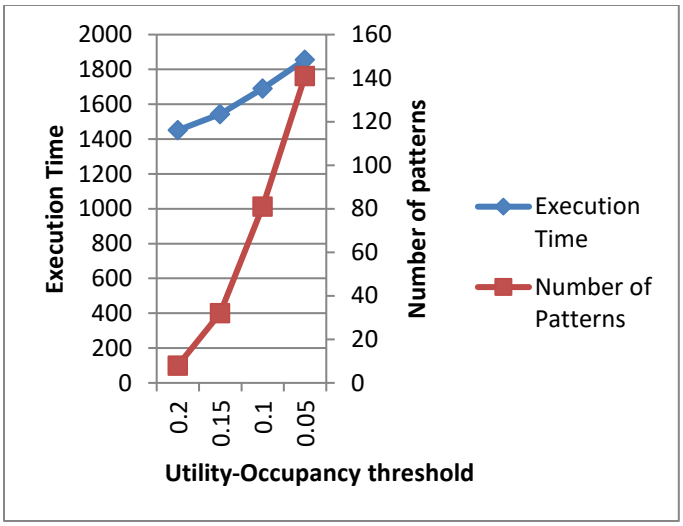
A. SPMF: An open-Source library

SPMF [12] provides open-source java implementations of various data mining algorithms. It provides various algorithms for discovering knowledge using association rules, sequential patterns and their variants, classification, clustering, etc. It also provides real-time & synthetic datasets used for testing the performance of various data mining algorithms.

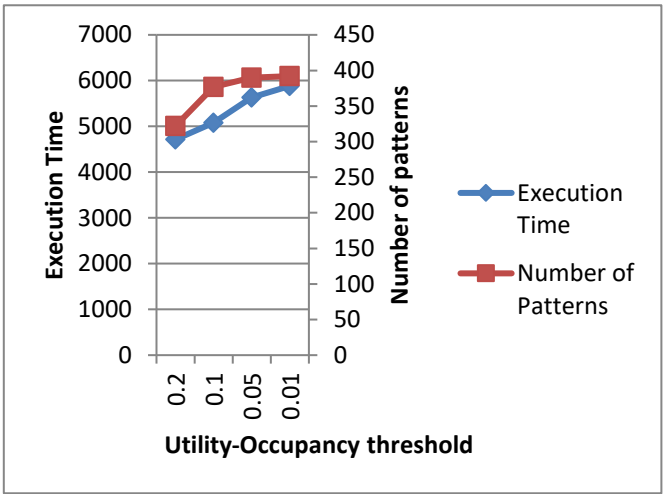
Datasets used in experimentation are collected from SPMF Framework. SIGN_sequence_utility dataset is a real dataset consisting of approximately 800 utterance sequences of sign language. Kosarak10k_sequence_utility dataset is a real dataset consisting of 10,000 click-stream sequences of a Hungarian online news portal.

B. Experimental Results

Experimentation was conducted on SIGN_sequence_utility, Kosarak10k_sequence_utility real datasets with varying support count threshold and utility-occupancy threshold. Fig. 2 shows the evaluation of the HUOSPM algorithm in terms of variation of runtime and number of patterns with respect to variation of utility-occupancy threshold while extracting high utility occupancy patterns from SIGN_sequence_utility dataset with support threshold of 0.5 and Kosarak10k_sequence_utility Dataset with support threshold of 0.01. It is observed from the results that the HUOSPM algorithm can completely discover HUOSP's.



a. SIGN_sequence_utility dataset with support threshold 0.5



b. Kosarak10k_sequence_utility dataset with support threshold 0.01

Fig. 2 Evaluation of runtime and number of patterns

Experimental analysis was also conducted to check the correctness of the proposed algorithm. Average utility occupancy is used as a metric for evaluating the quality of patterns generated by the proposed algorithm. Tables II, III, IV, V & VI represent the percentage of average utility occupancy of patterns and performance under the fixed utility-occupancy threshold and varying support threshold. The quality of patterns is evaluated by calculating the percentage of improvement in average utility occupancy between the prefix span and proposed algorithm under the fixed utility-occupancy threshold and varying support threshold. Fig. 3 and 4 show the comparison of the percentage of improvement in average utility occupancy between the prefix span and the proposed algorithm.

Table II. Evaluation of % of average utility occupancy of patterns on SIGN_sequence_utility Dataset

SIGN_sequence_utility		Utility-occupancy threshold: 0.1	
Support Threshold	Average utility occupancy of prefix span	Average utility occupancy of HUOSPM	% of improvement
0.7	0.14	0.18	28.57
0.6	0.13	0.17	30.76
0.5	0.09	0.14	55.55
0.3	0.07	0.12	71.42
0.2	0.08	0.14	75

Table III. Evaluation of % of average utility occupancy of patterns on SIGN_sequence_utility Dataset

SIGN_sequence_utility		Utility-occupancy threshold: 0.2	
Support Threshold	Average utility occupancy of prefix span	Average utility occupancy of HUOSPM	% of improvement
0.7	0.14	0.24	71.42
0.6	0.13	0.23	76.92
0.5	0.09	0.22	144.44
0.3	0.07	0.2	185.71

Table IV. Evaluation of % of average utility occupancy of patterns on Kosarak10k_sequence_utility

Kosarak10k_sequence_utility		Utility-occupancy threshold: 0.7	
Support Threshold	Average utility occupancy of prefix span	Average utility occupancy of HUOSPM	% of improvement
0.03	0.49	0.76	55.10
0.02	0.44	0.77	75
0.01	0.35	0.74	111.42
0.005	0.24	0.72	200
0.004	0.2	0.71	255
0.003	0.17	0.71	317.64

Table V. Evaluation of % of average utility occupancy of patterns on Kosarak10k_sequence_utility

Kosarak10k_sequence_utility		Utility-occupancy threshold: 0.5	
Support Threshold	Average utility occupancy of prefix span	Average utility occupancy of HUOSPM	% of improvement
0.03	0.49	0.64	30.61
0.02	0.44	0.59	34.09
0.01	0.35	0.56	60
0.005	0.24	0.58	141.66
0.004	0.2	0.58	190
0.003	0.17	0.57	235.29

Table VI. Evaluation of % of average utility occupancy of patterns on Kosarak10k_sequence_utility

Kosarak10k_sequence_utility		Utility-occupancy threshold: 0.3	
Support Threshold	Average utility occupancy of prefix span	Average utility occupancy of HUOSPM	% of improvement
0.03	0.49	0.51	4.08
0.02	0.44	0.49	11.36
0.01	0.35	0.42	20
0.005	0.24	0.38	58.33
0.004	0.2	0.37	85
0.003	0.17	0.37	117.64

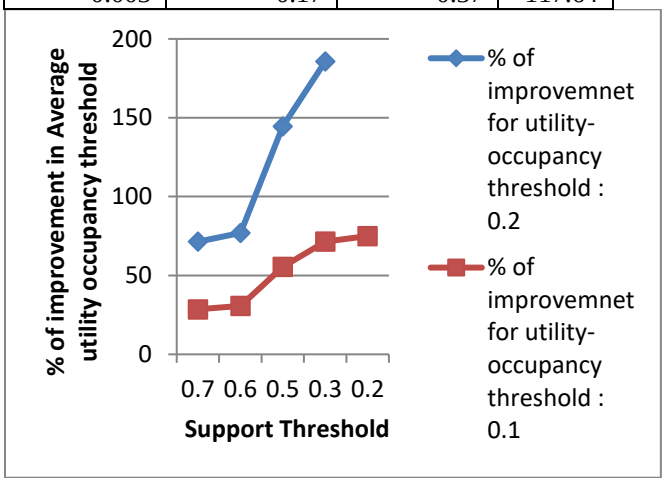


Fig. 3 Evaluation of % of improvement in average utility occupancy of patterns on SIGN_sequence_utility Dataset

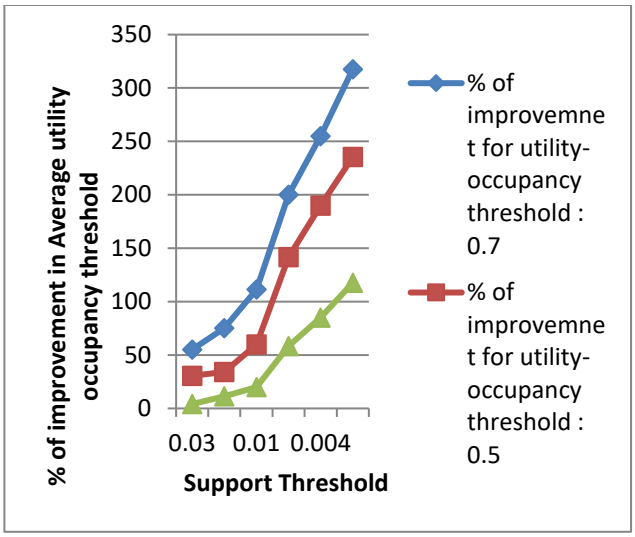


Fig. 4 Evaluation of % of improvement in average utility occupancy of patterns on Kosarak10k_sequence_utility Dataset

VI. CONCLUSIONS

From Fig. 3 and 4, it is clearly observed that patterns generated by the proposed algorithm are qualitative compared to prefix span. It can also be observed that patterns generated by the *HUOSPM* algorithm produce better values for lower support thresholds compared to prefix span.

In this paper, a novel variant of SPM called HUOSPM is proposed by considering item utility, occurrence, and utility-occupancy measures into account. The utility-occupancy measure helps in discovering useful and interesting utility patterns which contribute a major portion of utility in its supporting sequences. The proposed novel generic utility-occupancy framework based *HUOSPM* algorithm measures the pattern interestingness using both support threshold and utility-occupancy threshold. To our knowledge, no algorithm is developed to solve this problem for sequential datasets. The proposed algorithm effectively discovers HUOSP's from QSD using compact data structures *seqlist* and *uolist*. *HUOSPM* algorithm efficiently discovers the patterns using proposed search space pruning strategies called pattern extension utility-occupancy (*peuo*), reduced sequence utility-occupancy (*rsuo*), and extension upper bound utility-occupancy (*euuo*). Experimental evaluations on real datasets with varying support count threshold and utility-occupancy threshold proved that the patterns discovered by the *HUOSPM* algorithm are qualitative compared to prefix span. *HUOSPM* algorithm can completely generate HUOSP's.

REFERENCES

[1] R. Agrawal and R. Srikant, Mining sequential patterns, In Proceedings of 11th International Conference on Data Engineering, IEEE. (1995) 3–14.

[2] Ahmed, C.F., Tanbeer, S.K., Jeong, B.S., A novel approach for mining high-utility sequential patterns in sequence databases, ETRI Journal. 32(5)(2010) 676–686.

[3] J. Yin, Z. Zheng, and L. Cao, USpan: an efficient algorithm for mining high utility sequential patterns, ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p(2012) 660-668.

[4] G. C. Lan, T. P. Hong, V. S. Tseng, and S. L. Wang, Applying the maximum utility measure in high utility sequential pattern mining, Expert Systems with Applications. 41(11)(2014) 5071–5081.

[5] O. K. Alkan and P. Karagoz, CRoM and HuspExt: improving efficiency of high utility sequential pattern extraction, IEEE Transactions on Knowledge and Data Engineering. 27(10)(2015) 2645–2657.

[6] J. Z. Wang, J. L. Huang, and Y. C. Chen, On efficiently mining high utility sequential patterns, Knowledge and Information Systems. 49(2) (2016) 597–627.

[7] W. Gan, J. C. W. Lin, J. Zhang, P. Fournier-Viger, H. C. Chao, and P. S. Yu, Fast utility mining on sequence data, IEEE TRANSACTIONS ON CYBERNETICS. 52(2)(2020) 2168-2267.

[8] Bac Le, Ut Huynh, Duy-Tai Dinh, A pure array structure and parallel strategy for high-utility sequential pattern mining, Expert Systems With Applications, Elsevier, 104(2018) 107-120.

[9] LEI ZHANG, PING LUO, LINPENG TANG, ENHONG CHEN, QI LIU, MIN WANG, and HUI XIONG, Occupancy-Based Frequent Pattern Mining, ACM Transactions on Knowledge Discovery from Data. 10(2) (2015) 14:1- 14:33.

[10] B. Shen, Z. Wen, Y. Zhao, D. Zhou, and W. Zheng, OCEAN: Fast discovery of high utility occupancy itemsets, in Proc. Pac.–Asia Conf. Knowl. Disc. Data Mining. (2016) 354–365.

[11] Wensheng Gan, Jerry Chun-Wei Lin , Philippe Fournier-Viger, Han-Chieh Chao, and Philip S. Yu, HUOPM: High-Utility Occupancy Pattern Mining, IEEE TRANSACTIONS ON CYBERNETICS. 50(3) (2020) 1195-1208.

[12] P. Fournier-Viger, C. W. Lin, A. Gomariz, A. Soltani, Z. Deng, H. T. Lam, The SPMF open-source data mining library version 2, The European Conference on Principles of Data Mining and Knowledge Discovery. (2016) 36-40, URL: http://www.philippe-fournier-viger.com/spmf/.