Top-K High Utility Episode Mining from a Complex Event Sequence

Sonam Rathore
IIIT-Delhi, India
sonam13108@iiitd.ac.in

Siddharth Dawar
IIIT-Delhi, India
siddharthd@iiitd.ac.in

Vikram Goyal
IIIT-Delhi, India
vikram@iiitd.ac.in

Dhaval Patel
IIT Roorkee, India
patelfec@iitr.ac.in

ABSTRACT

Utility episode mining has emerged as an interesting and challenging research topic in data mining. It finds applications in anomaly detection, biomedical data analysis, predicting stock trends etc. The number of high-utility episodes that can be extracted from a sequence depends upon the value of minimum utility threshold. It is often difficult for a user to find a suitable threshold value which fits their purpose. The sequence can generate many high-utility episodes at low threshold value and very few episodes at higher threshold values. In order to relieve the user from this tedious task, we propose an algorithm for mining top-k high utility episodes from a complex event sequence. The parameter k can be set by the user according to his/her needs. We also propose effective strategies for raising the threshold value in order to prune the search space effectively. We conduct extensive experiments on real and synthetic datasets and the experimental results demonstrate the effectiveness of our proposed strategies in terms of total execution time and the number of candidate episodes generated.

1. INTRODUCTION

Frequent pattern mining finds patterns from a database, which have frequency no less than a given minimum support threshold. Frequent pattern mining finds applications in market-basket analysis, mining association rules [23], plagiarism detection and biomedical data analysis [30]. As types of data vary from one application to another, researchers have developed various frequent pattern mining algorithms. For example, frequent itemset mining [16] deals with transaction data, sequential pattern mining [27] operates on sequence data, frequent episode mining [24] works on long event sequence, and frequent pattern mining in data streams [19]. The algorithms developed for mining frequent patterns have mostly employed the monotone/anti-monotone property to prune the exponential search space effectively. The monotone property states that the subsets of a frequent pattern are also frequent and the anti-monotone property states that the supersets of an infrequent pattern are also infrequent.

However, the frequent patterns extracted can be of low-profit value. The concept of *high-utility pattern mining* was introduced to capture the notion of utility, which has been observed in real life. *High-utility pattern mining* finds patterns from a database which have their utility value no less than a given minimum utility-threshold. The utility function measures the importance of a pattern and varies according to the application. For example, in a retail store domain, a utility function can measure the profit made by the store by selling the items in the itemset together over a period of time. High-utility pattern mining has wide range of applications in cross-marketing in retail stores [8, 20], web-click stream analysis [18], medicine [25] etc. High-utility mining has also been applied with other mining techniques like high-utility sequential pattern mining [34, 31], high-utility pattern mining from a transaction database [5, 7, 10], mining high-utility patterns from a data stream [3], mining utility-frequency skyline pattern [11] and high-utility episode-pattern mining [32].

Mining high-utility episodes from a customer shopping sequence may discover patterns which may help in discovering the purchasing behaviour of customers. In this scenario, the items purchased by a customer in a transaction can be represented as events occurring at a time point. High-utility episode mining can also be used for stock prediction or investment [21, 22]. Some algorithms [26, 14] have been proposed to mine high-utility episodes from simple event sequences, where only one event occurs at a point in time. However, complex event sequences often occur in real life and has many applications as mentioned by Wu et al. [32]. Recently, Wu et al. [32] proposed an algorithm UP-Span to mine high-utility episodes from a complex event sequence. The number of high-utility episodes which can be extracted depends upon the value of the chosen utility threshold and the characteristics of the database. It is possible to extract many episodes at lower threshold values and very few episodes at higher threshold values. The user must analyze the distribution of items, utility value and density of the database in order to choose an appropriate utility value. In summary, a user's engagement while mining high-utility episodes is not a desirable solution.

In this paper, we propose a solution for mining top-k high-utility episodes from a complex event sequence. Our aim is

to relieve a user from the task of analyzing the database and choosing a utility threshold, which is often a difficult task for any user. The parameter k is the number of high-utility episodes to be extracted from the database. A naive approach for extracting top-k high-utility episodes can be to set the minimum utility threshold to zero and apply any high-utility episode mining algorithm to find the complete set of high-utility episodes. Top-k episodes can be then chosen from the result set. However, this approach is computationally very inefficient as the search space is exponential in the number of different items. In order to improve the efficiency, we propose effective strategies to raise the minimum utility threshold from zero as quickly as possible.

Our research contributions can be summarized as follows:

- We propose an algorithm to mine top-k high-utility episodes from a complex event sequence.
- We develop effective strategies to raise the minimum utility threshold quickly during the mining process in order to reduce the search space effectively.
- We conduct extensive experiments on real as well as synthetic datasets and the experiment results demonstrate the effectiveness of our approach.

The paper is organized as follows. Section 2 reviews the related work and background knowledge is explained in Section 3. We propose our algorithm and our strategies for improving the performance in Section 4. The experimental results are presented in Section 5 and Section 6 concludes the paper.

2. RELATED WORK

In this section, we briefly discuss some related work done in this field. We divide the work into two lines of research:

2.1 Frequent Episode Mining

Frequent Episode Mining (FEM) was first introduced by Mannila et al. [24]. Two algorithms WINEPI and MINEPI were proposed in this paper. In WINEPI, the events were sampled regularly over a sequence of events. An episode was considered interesting if it fits into a window width which is defined by the user. The support was computed by counting the number of sliding windows in which episode appeared. However, the algorithm could not avoid the double counting of occurrence of an episode. In order to resolve the issue, the concept of minimal occurrence was introduced. The minimal occurrence of an episode α is a time interval $[t_s,\ t_e]$ where the episode α occurs and it does not occur in any proper subinterval of $[t_s,\ t_e]$. In order to find the support, the algorithm counted the number of minimal occurrences of an episode.

However, the algorithms generate a large of candidate episodes. Several methods have been proposed to improve the performance of existing *FEM* algorithms. However, a majority of the studies are devoted towards mining frequent episodes in simple event sequences [1, 2, 4, 6, 15]. Mining frequent episodes from a complex event sequence were only considered by Huang et al. [17].

2.2 Utility Episode Mining

The *FEM* framework assumes that all the events are of same importance. Therefore, it may report many episodes of low revenue and miss high revenue but low-frequency episodes. For solving this issue, the concept of utility was introduced in episode mining by Guo et al. [13]. However, this paper only considered the external utility of an episode and mining was performed in a simple event sequence. Wu et al. [32] addressed this problem and proposed an algorithm UP-Span to discover high utility episodes in the complex event sequence. The events were associated with internal utility (quantity) and external utility (profit). The authors also proposed two strategies, namely Discarding Global unpromising Events (DGE) and Discarding Local unpromising Events (DLE), to discard unpromising events and reduce the search space. To speed up the UP-Span algorithm, Guo et al. [12] presented a prefix tree structure and tighter upper bounds for candidate episodes utility.

From the above-related work, we can conclude that only preliminary work is done in mining high utility episodes. Many algorithms have been proposed for top-k high-utility pattern mining from transaction [33, 29] and sequential databases [35]. However, the existing top-k utility mining algorithms cannot be directly applied to top-k high utility episode mining on a very large complex event sequence. If we try to transform a complex event sequence into a set of transactions or a set of sequences to make a sequential database, it is not straightforward. Further, this makes the existing algorithms inefficient for top-k utility episode mining from a complex event sequence.

3. BACKGROUND

In this section, we present some definitions given in the earlier works [32] and describe the problem statement formally.

3.1 Episode Mining

Definition 1. ***(Event)*** *An event is defined by the pair (e, t) where e is the event type and $t \in N^+$ is the time at which event occurs.*

Definition 2. ***(Complex event sequence)*** *A complex event sequence $CES = <(SE_1, t_1), (SE_2, t_2), \ldots, (SE_n, t_n)>$ is an ordered sequence of simultaneous event sets, where each simultaneous event set is associated with a time point $t \in N^+$ and $t_i<t_j$, for all $1 \leq i<j \leq n$.*

Let us consider the complex event sequence as shown in Figure 1. $(\langle(D)\rangle, 5)$ is an event which occurs at t_5.

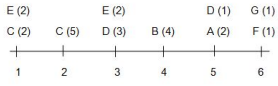


Figure 1: A Complex Event Sequence

A	B	C	D	E	F	G
1	4	2	1	5	3	2

Table 1: External Utility

Definition 3. ***(Simultaneous event set)*** *A simultaneous event set is composed of a set of events, where each event occurs at the same time point t.*

For example, $(\langle(DA)\rangle, 5)$ is a simultaneous event set which occurs at t_5.

Definition 4. ***(Episode containing simultaneous event sets)*** *An episode* $\alpha = <(SE_1), (SE_2), \ldots, (SE_n)>$ *is a non-empty totally ordered set of simultaneous events, where* SE_i *appears before* SE_j *for all* $1 \leq i < j \leq n$.

For example, $\langle (EC), (D) \rangle$ is an episode.

Definition 5. ***(Occurrence)*** *For an episode* $\alpha = <(SE_1, t_1), (SE_2, t_2)\ ,\ldots,\ (SE_n, t_n)>$, *the time interval* $[T_s, T_e]$ *is called the occurrence of episode if* α *occurs in* $[T_s, T_e]$ *and the first simultaneous event* SE_1 *of* α *occurs at time* T_s, *while the last simultaneous event set* SE_k *of* α *occurs at time* T_e.

For example, the occurrence set of episode $\langle (E), (D) \rangle$ is $occ(\langle (E), (D) \rangle) = [1, 3], [3, 3], [1, 5], [3, 5]$.

Definition 6. ***(Minimal Occurrence)*** *The occurrence time interval* $[T_s, T_e]$ *of an episode is called a minimal occurrence if there exists no sub-interval of* $[T_s, T_e]$ *in occurrence of* α. *Consider two time intervals* $[T_s, T_e]$ *and* $[T_s\ ',\ T_e\ ']$, $[T_s\ ',\ T_e\ ']$*is the sub-interval of* $[T_s,\ T_e]$ *if* $T_s \leq T_s\ '$ *and* $T_e\ ' \leq T_e$.

For example, the minimal occurrence set of episode $\langle (E), (D) \rangle$ is
$minOcc(\langle (E), (D) \rangle) = [1, 3], [3, 3], [3, 5]$.

Definition 7. ***(Support of an episode)*** *The support of an episode is defined as the number of minimal occurrences of an episode.*

For example, the support of episode $\langle (E), (D) \rangle$ is 3.

Definition 8. ***(Internal and External Utility)*** *In mining high utility episodes, each event* e_i *is associated with a positive number* $p(e_i)$, *called as the external utility. Each event* e_i *in a simultaneous event set* SE_j *at the time point* t_j *is associated with a positive number* $q(e_i, t_j)$, *called as internal utility.*

For example, the external utility of events is shown in Table 1.

Definition 9. ***(Utility of an event at a time point)*** *The utility of an event* e_i *at a time point* t_j *is* $u(e_i, t_j) = p(e_i, CES) \times q(e_i, t_j)$.

For example, the utility of event $\langle (F) \rangle$ at time t_6 is $1 \times 3 = 3$.

Definition 10. ***(Utility of a simultaneous event set at a time point)*** *The utility of a simultaneous event set* $SE = \langle (e_1, t_1), (e_2, t_2), (e_3, t_3), \ldots, (e_k, t_k) \rangle$ *at a time point* t_i *is defined as* $u(SE, t_i) = \Sigma_{i=1}^{k} u(e_i, t_i)$.

For example, utility of simultaneous event $\langle (GF) \rangle = (2 \times 1) + (3 \times 1) = 5$.

Definition 11. ***(Utility value of an episode w.r.t. its minimal occurrence)*** *Let* $mo(\alpha) = [T_s, T_e]$ *be a minimal occurrence of the episode* $\alpha = \langle (SE_1), (SE_2), \ldots, (SE_n) \rangle$, *where each simultaneous event set* $SE_i \in \alpha$ *is associated with a time point* T_i. *The utility of the episode* α *w.r.t* $mo(\alpha)$ *is defined as* $u(\alpha, mo(\alpha)) = \Sigma_{i=1}^{k} u(SE_j, t_i)$.

Definition 12. ***(Utility of an episode in a complex event sequence)*** *Let* $moSet(\alpha) = [TI_1, TI_2, \ldots, TI_k]$ *be the set of all minimal occurrences of the episode* α, *where* TI_i *is a minimal occurrence of* α *for* $1 \leq i \leq k$. *The utility value of the episode* α *in a complex event sequence* CS *is defined as* $uv(\alpha, CS) = \Sigma_{i=1}^{k} u(\alpha, TI_k)$.

For example, $u(\langle (E), (D) \rangle, [1, 3]) = 10 + 3 = 13$, and $u(\langle (E),$
$(D) \rangle, CES) = u(\langle (E), (D) \rangle, [1, 3]) + u(\langle (E),$
$(D) \rangle, [3, 3])\ + u(\langle (E), (D) \rangle, [3, 5]) = 13 + 13 + 11 = 37$.

Definition 13. ***(High Utility Episode(HUE))*** *If the utility of an episode is not less than the minimum utility threshold, then it is called as a high utility episode.*

Definition 14. ***(Maximum Time Duration)*** *Maximum Time Duration (abbreviated as MTD) is a user specified window. A minimal occurrence* $mo(\alpha) = [T_s, T_e]$ *satisfies the maximum time duration constraint iff,* $(T_e - T_s + 1) \leq MTD$.

Definition 15. ***(Simultaneous and Serial concatenations)*** *Let* $\alpha = \langle (SE_1), (SE_2), \ldots, (SE_x) \rangle$ *and* $\beta = \langle (SE'_1), (SE'_2), \ldots, (SE'_y) \rangle$ *be episodes. The simultaneous concatenation of* α *and* β *is defined as:* $simulconcat(\alpha, \beta) = \langle (SE_1), (SE_2), \ldots, (SE_x \bigcup SE'_1), (SE'_2), \ldots, (SE'_y) \rangle$.
The serial concatenation of α *and* β *is defined as:* $serialconcat(\alpha, \beta) = \langle (SE_1), (SE_2)\ , \ldots, (SE_x), (SE'_1), (SE'_2), \ldots, (SE'_y) \rangle$.

For example, Let $\alpha = (\langle (B) \rangle, [4, 4])$ and $\beta = (\langle (D) \rangle, [5, 5])$. The new episode formed by serial concatenation of α and β is $\gamma = \langle (B), (D) \rangle, [4, 5]$. Let now $\alpha = (\langle (A) \rangle, [5, 5])$. The simultaneous concatenation of γ and α is $\langle (B), (DA) \rangle, [4, 5]$.

Since downward closure property doesn't hold in utility mining, the authors [32] proposed the concept of Episode-weighted Utilization (EWU). The EWU satisfies the downward closure property i.e. if the EWU of an episode is less than the minimum utility threshold, all its super episodes will have low utility.

Definition 16. ***(Episode-Weighted Utilization of an episode w.r.t a minimal occurrence)*** *Let* $mo_j(\alpha) = [T_s, T_e]$ *be a minimal occurrence of the episode* $\alpha = \langle (SE_1), (SE_2), \ldots, (SE_x) \rangle$, *where each simultaneous event set* $SE_i \in \alpha$ *is associated with a time point* $T_i (1 \leq i \leq k)$ *and* $mo_j(\alpha)$ *satisfies* MTD. *The episode-weighted utilization of* α *w.r.t* $mo(\alpha)$ *is defined as* $EWU(\alpha, moj(\alpha)) = \Sigma_{i=1}^{k-1} u(SE_i, t_i) + \Sigma_{i=e}^{s+MTD-1} u(tSE_i, t_i)$ *where* tSE_i *is the simultaneous event set occurring at time point* T_i *in* CES.

Definition 17. ***(Episode-Weighted Utilization of an episode)*** *Let* $moSet(\alpha)$
$= [TI_1, TI_2, \ldots, TI_k]$ *be a set of minimal occurrences of the episode* $\alpha = \langle (SE_1), (SE_2), \ldots,$
$(SE_x) \rangle$, *where each simultaneous event set* $SE_i \in \alpha$ *is associated with a time point* $T_i (1 \leq i \leq k)$ *and* $mo_j(\alpha)$ *satisfies* MTD. *The episode weighted utilization of* α *in complex event sequence* CES *is defined as* $EWU(\alpha) = \Sigma_{i=1}^{k} u(\alpha, TI_i)$.

For example, let the MTD set by the user be 3.The EWU of $\langle (E), (D) \rangle$ w.r.t a minimal occurrence is
$EWU(\langle (E), (D) \rangle, [1, 3]) = 10 + 13 = 23$.

The EWU of $\langle(E),(D)\rangle$ in the CES is $EWU(\langle(E),(D)\rangle) = [EWU(\langle(E),(D)\rangle),[1,3]]+[EWU(\langle(E),(D)\rangle),[3,5]] = 23+13 = 36$.

Definition 18. ***(High Weighted Utilization Episode (HWUE))*** *An episode is called High Weighted Utilization Episode its EWU is no less than the minimum utility threshold $min\_utility$.*

4. EFFICIENT MINING OF TOP-K HIGH UTILITY EPISODES

In this section, we present our proposed algorithm, TUP-Basic (Top-k Utility episode mining), for mining top-k high utility episode mining in a complex event sequence. First, we propose a basic algorithm to find the top-k high utility episodes. Then, we propose some strategies to effectively raise the minimum utility threshold during the mining process.

4.1 TUP-Basic

In this subsection, we present a basic algorithm for discovering top-k high-utility episodes from a complex event sequence (See Algorithm 1). The algorithm takes as input a complex event sequence and a parameter k. The algorithm maintains a sorted list of size K dynamically, which contains the top-k high-utility episodes. The minimum utility threshold ($min\_utility$) stores the utility of the current k^{th} episode. The minimum utility threshold is initialized to zero before the start of mining process as the top-k buffer is empty.

First, the database is scanned once to find all 1-length episodes (Line 1). The minimal occurrence, utility and EWU of 1-length episodes are calculated. If the EWU of an episode is greater than the minimum threshold (Line 4), the algorithm explores the search space of high utility episodes with the current episode as the prefix. The algorithm follows a depth-first search strategy for finding new episodes.

To explore the search space, an episode is concatenated simultaneously (Line 5) and serially (Line 7) to events. Let the occurrence of an episode α be $[T_s, T_e]$. The events occurring at T_e are simultaneously concatenated to the episode α. While the events occurring between T_e+1 to $T_s+MTD-1$ are serially concatenated to episode α according to the definition 15.

Each new episode, β, formed by concatenation, calls Insert_Episode(.) which updates the list of Top-k episodes (See Algorithm 2). The input episode is added to the list if the size of the list is less than user-defined k or its EWU is no less than the minimum utility threshold. Once k candidates are discovered the minimum utility is raised to k^{th} highest utility in the list i.e. to the least utility in the list. Further, only episodes satisfying minimum utility is inserted in the list and the $(k+1)^{th}$ episode is removed from the buffer. After the algorithm terminates, top-K list contains the desired output of top-k high utility episode mining in the complex event sequence.

The algorithm returns the correct result as if the EWU of an episode is less than the utility of the current kth episode, it is guaranteed that no super-set of that episode can be in the top-k buffer due to downward closure property. The top-k episodes for k=3 and MTD=2 is: $\{\langle(ED),(B)\rangle : 29, \langle(E),(B)\rangle : 26, \langle(EC),(C)\rangle : 24\}$

Algorithm 1 TUP-Basic(CES, MTD, k)

Input: A complex event sequence CES,desired number of episodes k
Output: The complete set of top-k high utility episodes

```
1: Scan CES to find all one length episodes (oneLengthEpiSet).
2: min_utility=0.
3: for episode epi in oneLengthEpiSet do
4:     if EWU(epi) ≥ min_utility then
5:         Simultaneous Concatenation(epi, minOcc(epi),
6:         min_utility, k).
7:         Serial Concatenation(epi, minOcc(epi)
8:         , min_utility, k).
9:     end if
10: end for
```

Algorithm 2 Insert_Episode($epi, min\_utility, k$)

Input: an episode epi,minimum utility $min\_utility$, desired number of episodes k
Output: Top-K List: Top-k high utility episodes among the candidates

```
1: if size(Top − KList)<k then
2:     Add epi to Top-k List.
3: else
4:     if utility(epi)>min_utility then
5:         Remove k^th high utility episode i.e. episode
6:         having least utility.
7:         Add epi to the List.
8:         Sort the list in decreasing order of utility values
   of episodes.
9:         min_utility = least utility in Top-K List.
10:    end if
11: end if
```

4.2 Pre-insertion Strategy

The TUP-Basic algorithm generates many candidates since the minimum threshold start from zero. We try to raise the minimum threshold before mining high-utility episodes from a complex event sequence. The idea is to pre-insert the simultaneous event sets, i.e. the event sets occurring at the same time point, in the top-K list after the initial scan of the database. We call this strategy as the pre-insertion strategy. We will illustrate the effectiveness of this approach with an example.

Let us consider the example sequence as shown in Figure 1 and utility of events as per Table 1. Let the value of k be 3. First, the event set, $\langle(EC)\rangle : 14$, occurring at time point 1 is inserted. Similarly, the utilities of other simultaneous event sets are calculated and the event sets are inserted in the top-k list.After the simultaneous event sets are inserted, the top-k list is $\{\langle(B)\rangle : 16, \langle(EC)\rangle : 14, \langle(ED)\rangle : 13\}$. As seen from the example, the pre-insertion strategy increases the minimum utility threshold from zero to 13 before starting the mining process.

4.3 EWU Strategy

The EWU associated with every episode captures the frequency as well as utility aspects nicely. The probability of an episode to be of high-utility increases with its EWU value. The idea is to start exploring those episodes first which have higher EWU compared to others. We know that the min-

imum utility threshold in high-utility mining remains fixed and the order in which paths are explored does not affect the efficiency. But, in Top-k the order of path taken does matter because the minimum utility threshold is dependent on the candidates list. So in this strategy, we process those episodes first which have a higher EWU compared to other episodes. we sort the episodes w.r.t their EWU's before concatenating them serially or simultaneously with other episodes. Our hypothesis is that the EWU strategy will work better on dense datasets compared to sparse datasets as dense datasets generate a lot of high EWU episodes.

We illustrate the working of EWU strategy with an example. The EWU of single episodes is shown in Table 2. Since episode (E) has the highest EWU, it is processed first. Let the value of MTD and k be 2 and 3 respectively. The

E	C	D	B	A	G	F
53	47	37	19	8	5	5

Table 2: EWU of single episodes

simultaneous episode $\langle(EC)\rangle : 14$ is generated and inserted into the top-k buffer. Since no events can be simultaneously concatenated with $\langle(EC)\rangle$, the method for serial concatenation is called. The serial episode $\langle(EC),(C)\rangle : 24$ with utility 24 is generated and added to the top-k buffer. Since, no episodes can be serially or simultaneously concatenated with $\langle(EC),(C)\rangle$, the execution returns to episode $\langle(E)\rangle$ and the simultaneous episode $\langle(ED)\rangle : 13$ is generated and added to the buffer. The minimum utility threshold is set to 13. If we randomly select a path or use lexicographical order the minimum utility threshold may not increase so fast. For example, if we first process the episodes of path starting with episode (A) then the minimum threshold raises to value 3 as the Top-k list consists of episodes $\{\langle(AD)\rangle : 3, \langle(A,G)\rangle : 4, \langle(A,F)\rangle : 5\}$ after traversing this path. So, the strategy 2 helps in raising the minimum threshold efficiently and hence it prunes the search space faster.

5. EXPERIMENTS AND RESULTS

In this section, we evaluate the effectiveness of our proposed strategies. We also study the performance of combining pre-insertion and EWU strategies, which we refer as TUP-Combined. We conduct experiments on various real and synthetic datasets. The description of the real datasets is shown in Table 3. A transaction database can be considered as a complex event sequence by considering each item as an event and the items in a transaction as a simultaneous event. We implemented all the algorithms in Java with JDK 1.7 on a Windows 8 platform.

Table 3: *Characteristics of Real Datasets*

Dataset	#T_x	Avg. length	#Items	Type
ChainStore Small	10,000	7.2	46086	Sparse
Retail Small	10,000	5.2	16470	Sparse
Accidents Small	10,000	10	468	Dense
Mushroom	8,124	10	119	Dense

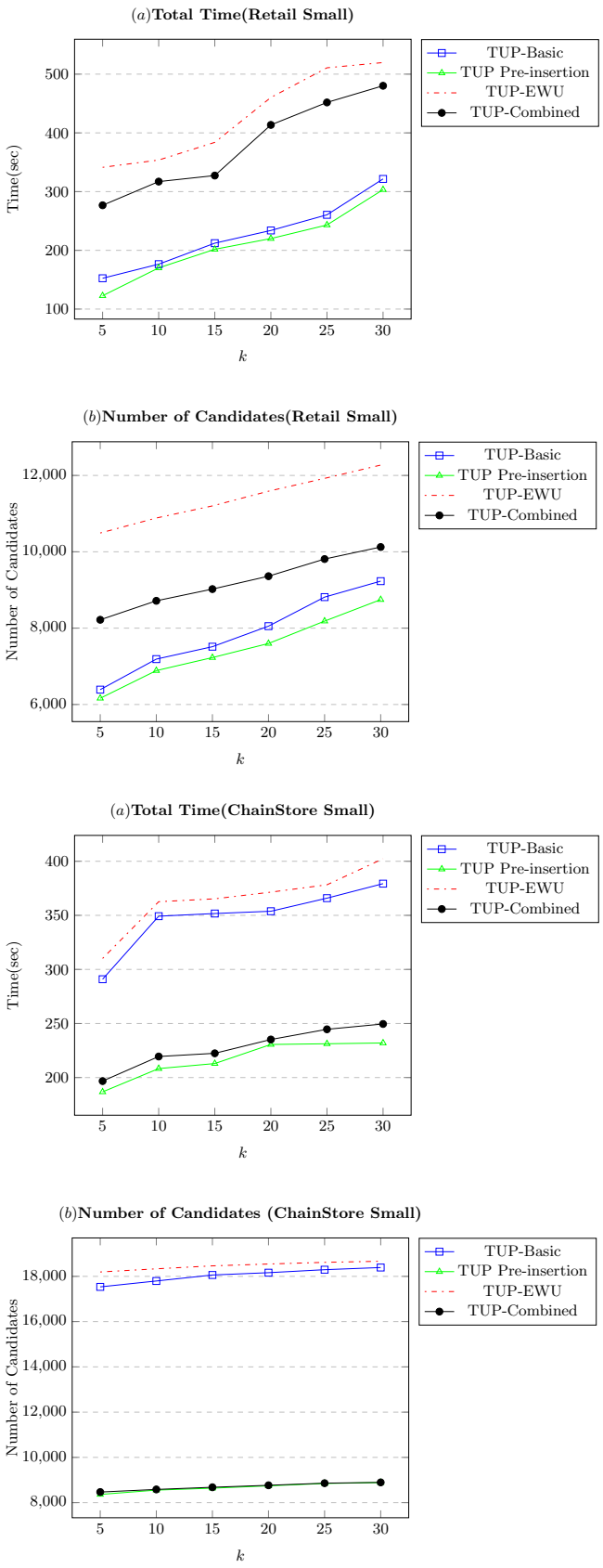
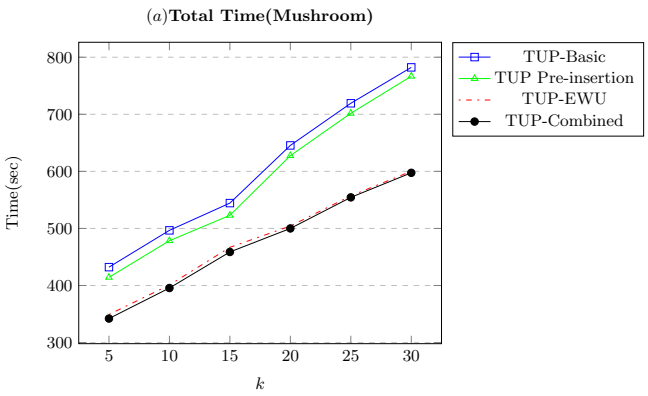
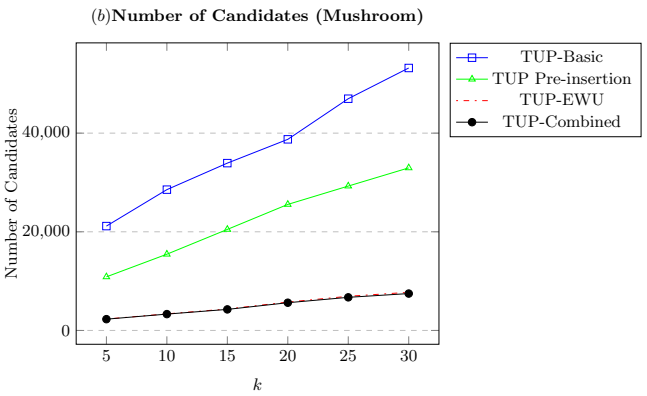
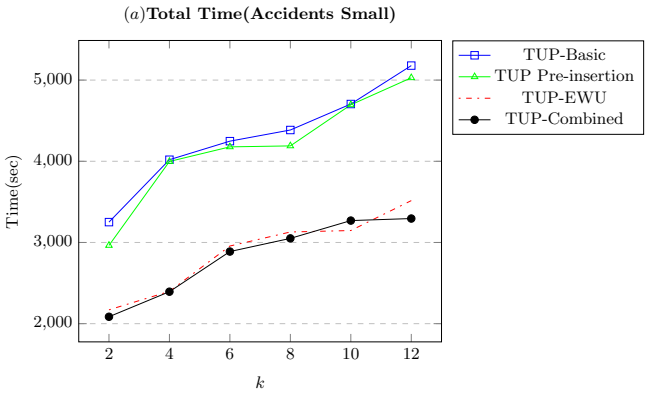


Figure 2: Performance Evaluation on Sparse Datasets







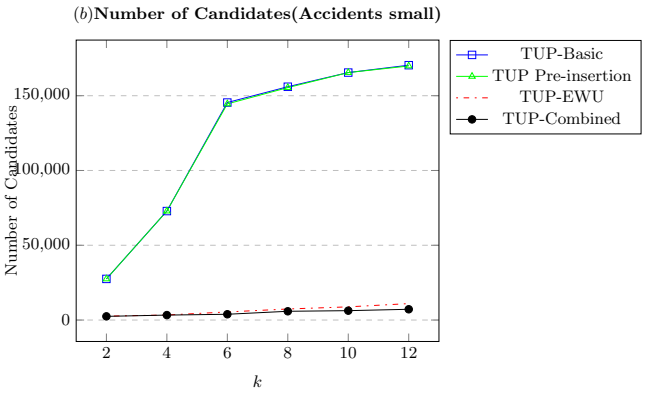
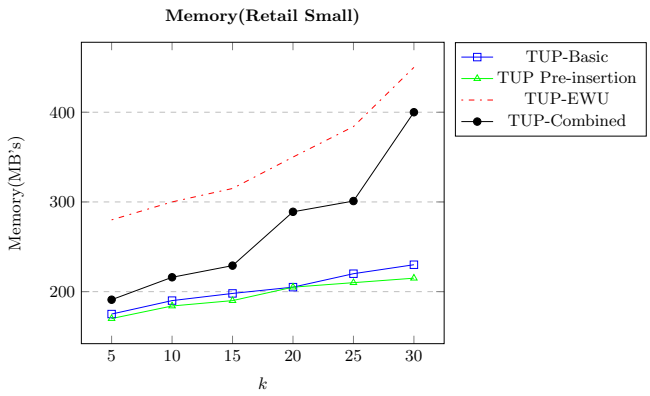
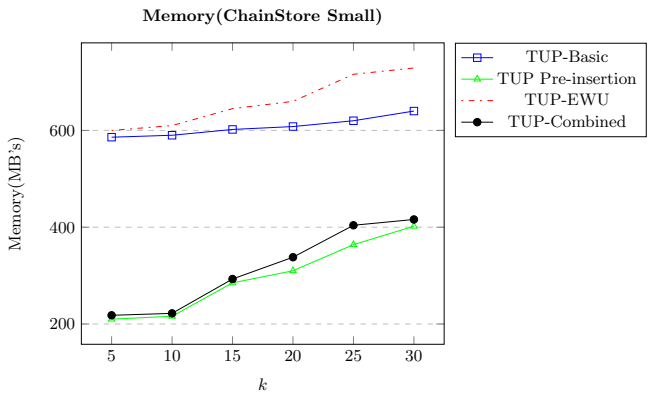
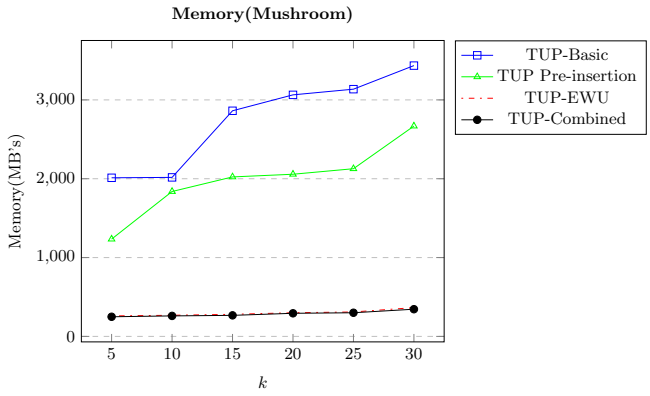


Figure 3: Performance Evaluation on Dense Datasets







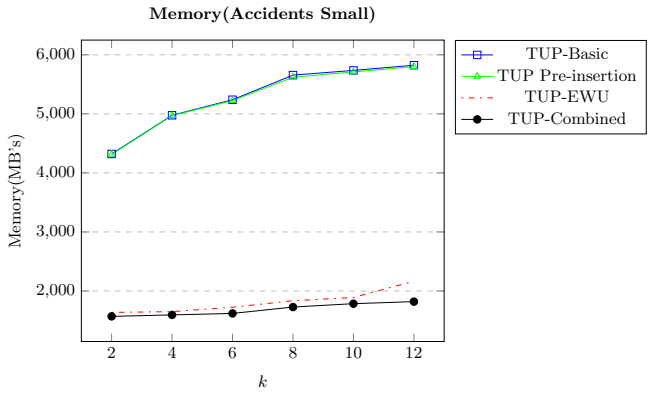


Figure 4: Memory Consumption on Real Datasets

The experiments were performed on an Intel Xeon(R) CPU=26500@2.00 GHz with 64 GB RAM. All real datasets except ChainStore were obtained from FIMI Repository [9]. The ChainStore dataset was obtained from Nu-MineBench 2.0 repository [28]. The quantity information for items was chosen randomly from 1 to 5. The external utility values were generated between 1 to 1000 using log-normal distribution.

We compared the performance of the algorithms on the basis of total execution time as well as the number of candidates. For datasets except Mushroom, we only take the first 10,000 transactions as it takes a lot of time to run experiments on a bigger sequence. Only ChainStore dataset has utility values associated with each item in the database. For other datasets, the utility values are generated between 1 to 5 using log-normal distribution. The quantity values are generated randomly between 1 to 5. We fix the maximum time duration(MTD) parameter to 2 for our experiments.

The results on sparse datasets are shown in Figure 2. The graphs show that pre-insertion strategy beats the other strategies in terms of total execution time and number of candidate episodes generated. The *EWU* strategy performs the worst as sparse datasets usually generate few short high-utility episodes.

The results on dense datasets are shown in Figure 3. The result shows that the performance of TUP-EWU and TUP-Combined strategy outperforms the performance of other strategies. Processing those episodes first which have higher *EWU* is effective in dense datasets as the high-utility episodes usually have high support value and are of longer length compared to sparse datasets.

We further analyzed the memory consumption of the algorithms. The results are shown in Figure 4. The results show that TUP−EWU and TUP−Combined consume less memory on dense datasets as they generate less number of candidate episodes.

6. CONCLUSION

High-utility Episode mining in a complex event sequence is an emerging topic in data mining and has many applications in real world. In this paper, we propose an efficient algorithm for mining top-k high utility episodes in a complex event sequence. We further proposed effective strategies to prune the search space effectively by raising the minimum utility threshold. We conducted extensive experiments on various datasets and the results demonstrate the effectiveness of our proposed strategies.

7. REFERENCES

[1] A. Achar, I. A, and P. S. Sastry. Editorial: Pattern-growth based frequent serial episode discovery. *Data Knowl. Eng.*, 87:91–108, Sept. 2013.

[2] A. Achar, S. Laxman, and P. S. Sastry. A unified view of the apriori-based algorithms for frequent episode discovery. *Knowl. Inf. Syst.*, 31(2):223–250, May 2012.

[3] C. F. Ahmed, S. K. Tanbeer, B.-S. Jeong, and H.-J. Choi. Interactive mining of high utility patterns over data streams. *Expert Systems with Applications*, 39(15):11979 – 11991, 2012.

[4] M. Atallah, W. Szpankowski, and R. Gwadera. Detection of significant sets of episodes in event sequences. In *Data Mining, 2004. ICDM '04. Fourth IEEE International Conference on*, pages 3–10, Nov 2004.

[5] R. Bansal, S. Dawar, and V. Goyal. An efficient algorithm for mining high-utility itemsets with discount notion. In N. Kumar and V. Bhatnagar, editors, *Big Data Analytics*, volume 9498 of *Lecture Notes in Computer Science*, pages 84–98. Springer International Publishing, 2015.

[6] G. Casas-Garriga. Discovering unbounded episodes in sequential data. In N. Lavra, D. Gamberger, L. Todorovski, and H. Blockeel, editors, *Knowledge Discovery in Databases: PKDD 2003*, volume 2838 of *Lecture Notes in Computer Science*, pages 83–94. 2003.

[7] S. Dawar and V. Goyal. Up-hist tree: An efficient data structure for mining high utility patterns from transaction databases. In *Proceedings of the 19th International Database Engineering & Applications Symposium*, IDEAS '15, pages 56–61, 2014.

[8] A. Erwin, R. Gopalan, and N. Achuthan. Efficient mining of high utility itemsets from large datasets. In T. Washio, E. Suzuki, K. Ting, and A. Inokuchi, editors, *Advances in Knowledge Discovery and Data Mining*, volume 5012 of *Lecture Notes in Computer Science*, pages 554–561. 2008.

[9] B. Goethals and M. Zaki. the fimi repository, 2012.

[10] V. Goyal, S. Dawar, and A. Sureka. High utility rare itemset mining over transaction databases. In W. Chu, S. Kikuchi, and S. Bhalla, editors, *Databases in Networked Information Systems*, volume 8999 of *Lecture Notes in Computer Science*, pages 27–40. 2015.

[11] V. Goyal, A. Sureka, and D. Patel. Efficient skyline itemsets mining. In *Proceedings of the Eighth International C* Conference on Computer Science & Software Engineering*, C3S2E '15, pages 119–124, 2008.

[12] G. Guo, L. Zhang, Q. Liu, E. Chen, F. Zhu, and C. Guan. High utility episode mining made practical and fast. In X. Luo, J. Yu, and Z. Li, editors, *Advanced Data Mining and Applications*, volume 8933 of *Lecture Notes in Computer Science*, pages 71–84. 2014.

[13] T. Guo, S. Lin, Y. Wang, and J. Qiao. A new framework for detecting high-utility episodes in event sequence. In *IEEE International Conference on Oxide Materials for Electronic Engineering (OMEE)*, pages 370–373, 2012.

[14] R. Gwadera, M. Atallah, and W. Szpankowski. Reliable detection of episodes in event sequences. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 67–74, Nov 2003.

[15] R. Gwadera, M. J. Atallah, and W. Szpankowski. Reliable detection of episodes in event sequences. *Knowl. Inf. Syst.*, 7(4):415–437, May 2005.

[16] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. *SIGMOD Rec.*, 29(2):1–12, May 2000.

[17] K.-Y. Huang and C.-H. Chang. Efficient mining of frequent episodes from complex sequences. *Information Systems*, 33(1):96 – 114, 2008.

[18] H.-F. Li, H.-Y. Huang, and S.-Y. Lee. Fast and memory efficient mining of high-utility itemsets from data streams: with and without negative item profits.

Knowledge and Information Systems, 28(3):495–522, 2011.

[19] H.-F. Li and S.-Y. Lee. Mining frequent itemsets over data streams using efficient window sliding techniques. *Expert Systems with Applications*, 36(2, Part 1):1466 – 1477, 2009.

[20] Y.-C. Li, J.-S. Yeh, and C.-C. Chang. Isolated items discarding strategy for discovering high utility itemsets. *Data and Knowledge Engineering*, 64(1):198 – 217, 2008.

[21] Y.-F. Lin, C.-F. Huang, and V. S. Tseng. A novel episode mining methodology for stock investment. *Journal of Information Science and Engineering*, 30(3):571–585, 2014.

[22] Y.-F. Lin, C.-W. Wu, C.-F. Huang, and V. S. Tseng. Discovering utility-based episode rules in complex event sequences. *Expert Systems with Applications*, 42(12):5303 – 5314, 2015.

[23] B. L. W. H. Y. Ma. Integrating classification and association rule mining. In *international Conference on Knowledge Discovery and Data Mining*, 1998.

[24] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovering frequent episodes in sequences (extended abstract). In *In 1st Conference on Knowledge Discovery and Data Mining*, pages 210–215, 1995.

[25] F. Medici, M. I. Hawa, A. Giorgini, A. Panelo, C. M. Solfelix, R. Leslie, and P. Pozzilli. Antibodies to gad65 and a tyrosine phosphatase-like molecule ia-2ic in filipino type 1 diabetic patients. *Diabetes Care*, 22(9):1458–1461, 1999.

[26] A. Ng and A.-c. Fu. Mining frequent episodes for relating financial events and stock trends. In K.-Y. Whang, J. Jeon, K. Shim, and J. Srivastava, editors, *Advances in Knowledge Discovery and Data Mining*, volume 2637 of *Lecture Notes in Computer Science*, pages 27–39. 2003.

[27] J. Pei, J. Han, B. Mortazavi-asl, H. Pinto, Q. Chen, U. Dayal, and M. chun Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *icccn*, pages 215–224, 2001.

[28] J. Pisharath, Y. Liu, W.-k. Liao, A. Choudhary, G. Memik, and J. Parhi. Nu-minebench 2.0. *Department of Electrical and Computer Engineering, Northwestern University, Tech. Rep*, 2005.

[29] H. Ryang and U. Yun. Top-k high utility pattern mining with effective threshold raising strategies. *Knowledge-Based Systems*, 76:109 – 126, 2015.

[30] W. Shi, F. K. Ngok, and D. R. Zusman. Cell density regulates cellular reversal frequency in myxococcus xanthus. *Proceedings of the National Academy of Sciences*, 93(9):4142–4146, 1996.

[31] B.-E. Shie, H.-F. Hsiao, V. Tseng, and P. Yu. Mining high utility mobile sequential patterns in mobile commerce environments. In J. Yu, M. Kim, and R. Unland, editors, *Database Systems for Advanced Applications*, volume 6587 of *Lecture Notes in Computer Science*, pages 224–238. 2011.

[32] C.-W. Wu, Y.-F. Lin, P. S. Yu, and V. S. Tseng. Mining high utility episodes in complex event sequences. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 536–544, 2013.

[33] C. W. Wu, B.-E. Shie, V. S. Tseng, and P. S. Yu. Mining top-k high utility itemsets. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 78–86, New York, NY, USA, 2012.

[34] J. Yin, Z. Zheng, and L. Cao. Uspan: An efficient algorithm for mining high utility sequential patterns. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 660–668, 2012.

[35] J. Yin, Z. Zheng, L. Cao, Y. Song, and W. Wei. Efficiently mining top-k high utility sequential patterns. In *IEEE 13th International Conference on Data Mining*, pages 1259–1264, Dec 2013.