Mining High Utility Episodes in Complex Event Sequences

Cheng-Wei Wu[1], Yu-Feng Lin[1], Philip S. Yu[2], Vincent S. Tseng[1]
[1]Department of Computer Science and Information Engineering, National Cheng Kung University, Taiwan, ROC
[2]Department of Computer Science, University of Illinois at Chicago, Chicago, Illinois, USA
{silvemoonfox, aorborcord}@gmail.com, psyu@cs.uic.edu, tsengsm@mail.ncku.edu.tw

ABSTRACT

Frequent episode mining (FEM) is an interesting research topic in data mining with wide range of applications. However, the traditional framework of FEM treats all events as having the same importance/utility and assumes that a same type of event appears at most once at any time point. These simplifying assumptions do not reflect the characteristics of scenarios in real applications and thus the useful information of episodes in terms of utilities such as profits is lost. Furthermore, most studies on FEM focused on mining episodes in simple event sequences and few considered the scenario of complex event sequences, where different events can occur simultaneously. To address these issues, in this paper, we incorporate the concept of utility into episode mining and address a new problem of *mining high utility episodes from complex event sequences*, which has not been explored so far. In the proposed framework, the importance/utility of different events is considered and multiple events can appear simultaneously. Several novel features are incorporated into the proposed framework to resolve the challenges raised by this new problem, such as the absence of anti-monotone property and the huge set of candidate episodes. Moreover, an efficient algorithm named *UP-Span* (*Utility ePisodes mining by Spanning prefixes*) is proposed for mining high utility episodes with several strategies incorporated for pruning the search space to achieve high efficiency. Experimental results on real and synthetic datasets show that *UP-Span* has excellent performance and serves as an effective solution to the new problem of mining high utility episodes from complex event sequences.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications — Data Mining

Keywords: Utility mining, episode mining, high utility episodes, complex event sequences

1. INTRODUCTION

Frequent pattern mining (abbreviated as FPM) [1, 3, 4, 12, 24] is a fundamental research topic in data mining, which refers to discovering patterns that appear in a dataset with frequency no less than a user-specified *minimum support threshold*. Many studies have been dedicated to this research, including *frequent itemset mining* [3, 12], *sequential pattern mining* [1, 4, 24] and *frequent episode mining* [2, 9, 11, 16, 19, 20, 21, 22, 23, 30, 31]. However, the classical framework of FPM may discover a large amount of frequent but low revenue patterns and lose the information on valuable patterns having low selling frequencies. Hence, the traditional framework of FPM cannot satisfy the requirement of users who desire to discover patterns with high utilities such as high profits.

To address these issues, *utility pattern mining* (abbreviated as UPM) [5, 6, 7, 8, 13, 14, 15, 17, 18, 25, 26, 27, 28, 29, 32] emerges as an important topic in data mining. In utility pattern mining, each item in the database has a weight (e.g. unit profit) and can appear more than once during a time period (e.g. purchase quantity). The utility of a pattern represents its importance, which can be measured in terms of weight, profit, cost, quantity or other information depending on the user preference. Mining high utility patterns refers to discovering patterns that appear in a dataset with utility no less than a user-specified *minimum utility threshold*. Utility pattern mining is an important task and has a wide range of applications such as website click stream analysis [5, 13, 6], cross-marketing in retail stores [15, 17, 25, 28] and biomedical applications [8].

Although high utility pattern mining is essential to many applications, it is not an easy task because the *downward closure property* [1, 3, 4, 12, 24] in FPM does not hold in UPM. To facilitate the task of high utility pattern mining, most studies [5, 13, 14, 18, 26, 27, 28, 29] incorporate the concept of *TWU* (*Transaction Weighted Utilization*). In the TWU model, a pattern is considered as a candidate or potential high utility pattern (abbreviated as PHUI) if its TWU is no less than the minimum utility threshold, where the TWU of a pattern represents the upper bound of its utility. A general TWU model consists of phase I and phase II. In phase I, all the potential high utility patterns are found. In phase II, high utility patterns are identified from the set of PHUIs by calculating the exact utilities of PHUPs.

Although many studies have been devoted to utility pattern mining, most of them focus on mining *high utility itemsets from transactional databases* [5, 13, 14, 15, 17, 18, 26, 27, 28, 29] or mining *high utility sequential patterns from sequence databases* [6, 7, 25, 33]. The topic of discovering *high utility episodes in complex event sequences* has not been explored so far. An *event sequence* is a long sequence of events. Each *event* is described by its type and a time of occurrence. An *episode* is a set of partially ordered events. The traditional framework of *frequent episode mining* (abbreviated as FEM) [2, 9, 11, 16, 19, 20, 21, 22, 23, 30, 31] is to find episodes that frequently occur in an event sequence. However, the traditional framework of FEM treats all events as having the same weight/utility and assumes that events can only occur at most once at any time point. These simplifying assumptions do not reflect the characteristics of real-life applications. This may result in discovering episodes having low utility (e.g. low profit). Furthermore, most studies on FEM focused on mining episodes in simple event sequences and few considered the scenario of complex event sequences, where different events can occur simultaneously at the same time point.

However, sequences containing such information are often encountered in real-life applications. For instance, in customer behavior analysis, a complex event sequence represents the purchase behavior of a customer. Each time point represents the items bought in a transaction (within a time period) by the customer.

KDD'13, August 11–14, 2013, Chicago, Illinois, USA.

Each purchased item can be regarded as an event having a quantity (internal utility) and a purchase price (external utility). Mining high utility episodes from such sequences can find sequential relationships between sets of items that contribute high profits, which is very valuable for business. Although mining high utility episodes from complex event sequences is desirable for many applications, it is not an easy task to incorporate the concept of utility mining with episode mining. It may pose the following challenges.

First, the utility of an episode is neither monotone nor anti-monotone [22]. In other words, the utility of an episode may be equal to, higher or lower than that of its supersets and subsets. Therefore, many techniques [2, 4, 12, 16, 22, 24, 31] developed in FEM that rely on anti-monotonicity to prune the search space cannot be directly applied to high utility episode mining.

Second, mining episodes from complex event sequences is not a trivial task. In the complex event sequences, different events can occur simultaneously at the same time point. This is substantially different and much more challenging than mining episodes from simple event sequences.

The third challenge is how to incorporate the concept of episode mining with the TWU model [5, 13, 14, 18, 26, 27, 28, 29] to facilitate the mining task. Although the TWU model is widely used in utility pattern mining, it is difficult to adapt this model to high utility episode mining because the dataset to be mined is a single, very long event sequence, which is very different from the transactional database [3, 12, 26] and sequence database [24, 32].

The forth challenge is how to reduce the number of candidates produced in phase I as much as possible if the TWU model can be applied to the high utility episode mining. A large number of candidates produced in phased I may degrade the performance of the mining task in terms of execution time and memory consumption. Therefore, it is important to develop effective strategies to prune the candidates and the search space.

In this paper, we address all of the above challenges by proposing a new framework for mining high utility episodes in complex event sequences. The major contributions of this work are summarized as follows:

First, we incorporate the concept of utility into episode mining and formalize the problem of high utility episode mining. An efficient algorithm named *UP-Span* (*Utility ePisodes mining by Spanning prefixes*) is proposed for mining the complete set of high utility episodes from complex event sequences.

Second, we integrate the concept of TWU model into high utility episode mining and propose *EWU* model (*Episode-Weighted Utilization model*) to efficiently find high utility episodes. Several strategies are proposed to prune the search space and reduce the number of candidates in the mining processes. The proposed strategies improve the overall performance of the mining task. In the experiment, the number of candidates produced by the proposed algorithm is much smaller than that of the baseline algorithm.

Third, we conduct a series of experiments with both synthetic and real datasets. The results show that the proposed framework and the UP-Span algorithm can efficiently discover high utility episodes from large scale data. In particular, the proposed UP-Span algorithm outperforms the baseline algorithm substantially (over two orders of magnitude) and serves as an effective solution to the new problem of mining high utility episodes from complex event sequences.

The remainder of this paper is organized as follows. Section 2 introduces the background for episode mining and utility mining. Section 3 gives the formal definition of high utility episodes and presents the proposed algorithms. Experiments are shown in Section 4. Conclusions and future work are given in Section 5.

2. BACKGROUND

This section introduces the preliminaries related to episode mining and high utility pattern mining.

2.1 Episode Mining

We introduce definitions and properties related to episode mining. For more details about episode mining, readers can refer to [2, 9, 11, 16, 19, 20, 21, 22, 23, 30, 31].

Definition 1 (Simple event sequence). Let $\varepsilon = \{E_1, E_2,\ldots,E_m\}$ be a finite set of *events* and N^+ be a set of time points. A *simple event sequence* $SS = \langle(E_1, T_1), (E_2, T_2),\ldots, (E_n, T_n)\rangle$ is an ordered sequence of events, where each event $E_i \in \varepsilon$ is associated with a time point $T_i \in N^+$ and $T_i < T_j$, for all $1 \le i < j \le n$. For example, Figure 1 shows a simple event sequence $SS = \langle((A), T_1), ((B), T_2), ((C), T_3), ((A), T_5), ((D), T_6), ((C), T_7)\rangle$.

Definition 2 (Simple episode). A *simple episode* α is a non-empty totally ordered set of events of the form $\langle(E_1), (E_2),\ldots, (E_k)\rangle$, where the event E_i appears before the event E_j for all $1 \le i < j \le k$. For example, $\langle(A), (C)\rangle$ is a simple episode.

Definition 3 (Simultaneous event set). A simultaneous event set $SE = (E_1, E_2,\ldots, E_m)$ is composed of a set of events, where each event $E_i \in \varepsilon$ in SE occurs at the same time point t for all $1 \le i \le m$. The length of a SE is denoted by $|SE|$ and is equal to the number of events in SE. Given two simultaneous event sets $SE_1 = (E_1, E_2,\ldots, E_n)$ and $SE_2 = (E_1', E_2',\ldots, E_m')$, where $m \le n$, SE_2 is the subset of SE_1 and SE_1 is the superset of SE_2 iff $SE_2 \subseteq SE_1$.

Definition 4 (Complex event sequence). A *complex event sequence* $CS = \langle(SE_1, T_1), (SE_2, T_2),\ldots, (SE_n, T_n)\rangle$ is an ordered sequence of simultaneous event sets, where each simultaneous event set SE_i is associated with a time point $T_i \in N^+$ and $T_i < T_j$, for all $1 \le i < j \le n$. For example, Figure 2 shows a complex event sequence $CS = \langle((AB), T_1), ((BC), T_2), ((C), T_3), ((AB), T_5), ((CD), T_6), ((C), T_7)\rangle$.

Definition 5 (Episode containing simultaneous events). An episode α is a non-empty totally ordered set of simultaneous events of the form $\langle(SE_1), (SE_2),\ldots, (SE_k)\rangle$, where SE_i appears before SE_j for all $1 \le i < j \le k$. For example, $\langle(AB), (C)\rangle$ is an episode containing the simultaneous event set (AB).

Figure 1. A simple event sequence

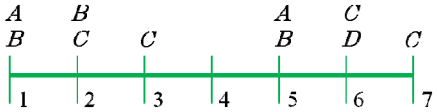


Figure 2. A complex event sequence

Definition 6 (Length and Size). The length of an episode $\alpha = \langle(SE_1), (SE_2),\ldots, (SE_k)\rangle$ is defined as $|\alpha| = \sum_{i=1}^{k} |SE_i|$ and is equal to the number of events in α. An episode α of length k is called k-episode. The size of α is defined as the number of simultaneous event sets in α. For example, $\langle(AB), (C)\rangle$ is a 3-episode of size 2.

Definition 7 (Occurrence). Given an episode $\alpha = \langle(SE_1), (SE_2),\ldots, (SE_k)\rangle$, the time interval $[T_s, T_e]$ is called the *occurrence of* α if (1) α occurs in $[T_s, T_e]$, (2) the first simultaneous event set SE_1 of α occurs at time T_s and the last simultaneous event set SE_k of α occurs at time T_e. The set of all occurrences of α is denoted as *occSet*(α). For

example, the set of all the occurrences of <(*AB*), (*C*)> in Figure 2 is *occSet*(<(*AB*), *C*>) = {[1, 2], [1, 3], [1, 6], [1, 7], [5, 6], [5, 7]}.

Definition 8 (Minimal occurrence). Given two time intervals [T_s, T_e] and [T_s', T_e'] of occurrences of episode α, [T_s', T_e'] is the *sub-time interval* of [T_s, T_e] if $T_s \leq T_s$' and T_e' $\leq T_e$. The time interval [T_s, T_e] is called a *minimal occurrence of episode* α if (1) [T_s, T_e] is the occurrence of episode α and (2) there is no alternative occurrence [T_s', T_e'] of α such that [T_s', T_e'] is the sub-time interval of [T_s, T_e]. A minimal occurrences of α is denoted as *mo*(α). The complete set of minimal occurrences of α is denoted as *moSet*(α). For example, the time interval [1, 2] is a minimal occurrence of <(*AB*), *C*> and *moSet*(<(*AB*), *C*>) = {[1, 2], [5, 6]}.

Definition 9 (Support of an episode). The *support count of an episode* α is defined as the number of minimal occurrences in *moSet*(α) and denoted as *SC*(α). The *support of* α is defined as the ratio of *SC*(α) to the number of time points in *CS*.

Definition 10 (Frequent episode). An episode is called *frequent*, iff its support is no less than a user-specified *minimum support threshold min_sup*. Otherwise, the episode is *infrequent*.

Definition 11 (Frequent episode mining). Given an event sequence *CS* and a user-specified minimum support threshold *min_sup*, the problem of frequent episode mining is to extract all the episodes having a support no less than *min_sup*.

Definition 12 (Sub-episode and super-episode). Given two episodes $\alpha = \langle SE_1, SE_2, \ldots, SE_n \rangle$ and $\beta = \langle SE_1', SE_2', \ldots, SE_m' \rangle$ where $m \leq n$, the episode β is a *sub-episode* of α iff there exists m integers $1 \leq i_1 < i_2 < \ldots < i_m \leq n$ such that $SE_{ik}' \subseteq SE_k$ for $1 \leq k \leq m \leq n$. In addition, episode α is the *super-episode* of β.

Property 1 (Downward closure property for frequent episode mining). The *downward closure* property states that: (1) For any frequent episode, all its sub-episodes are frequent. (2) For any infrequent episode, all is super-episodes are infrequent.
Proof. The reader is referred to [22] for the proof.

Episode mining is an interesting research topic in data mining with wide range of applications. The topic of mining frequent episodes in simple event sequences was first introduced by Mannila et al. [22]. They proposed two algorithms named *WINEPI* and *MINEPI* to find episodes that frequently occur in a simple event sequence. Although *WINEPI* and *MINEPI* algorithms are the pioneers in episode mining and perform well in some cases, they are *Apriori-based* approaches and employee candidate-generation-and-test mechanisms to find frequent episodes. Therefore, they often generate a large number of candidates during the mining processes, which may degrade the performance of the mining task in terms of execution time and memory consumption. To improve the performance of *MINEPI* algorithm, Ma et al. proposed the *PPS* (Position pairs set) algorithm [31], which efficiently finds frequent episodes without generating any candidate during the mining processes. Based on [22], several studies were proposed for mining various types of significant episodes or episode rules. In addition, episode mining is essential to many applications such as event detection in sensor network [30], occurrences of recurrent illnesses [21, 23] and financial data [2].

Although many studies have been devoted to episode mining, most studies on frequent episode mining focused on mining simple episodes in simple event sequences and few considered the scenario of complex event sequences, where different events can occur simultaneously at the same time point. By considering complex event sequences, the episode containing simultaneous events can be discovered, which provides additional information about the relationships between events. Besides, the traditional framework of frequent episode mining treats all events as having the same importance/utility and assumes that an event appears at most once at any time point. These assumptions do not reflect the characteristics in real scenario of several real-life applications and thus the useful information of episodes with high utilities such as high profits is lost. Although discovering episodes with high utility is desirable for many applications, the topic of high utility episode mining has not been addressed so far. In the next subsection, we study the related works about utility mining.

2.2 Utility Pattern Mining

We introduce the preliminary works related to high utility itemset mining, high utility sequential pattern mining and high utility episode mining. For a recent overview of research on utility mining, readers can refer to [5, 6, 7, 8, 13, 14, 15, 17, 18, 25, 26, 27, 28, 29, 32].

The concept of utility mining was first introduced in [8]. In utility pattern mining, each item in a database is associated with an additional value, called its *external utility*, which can be used to indicate the importance/weight/unit profit of the item. Each item appearing in a record of the database is attached with its *internal utility*, which indicates the quality/appearance/quantity of the item in the record. The *utility of an itemset* (a set of items) can be measured by considering its external utility and internal utility. An itemset is called *high utility* if its utility is no less than a *minimum utility threshold*. Otherwise, the itemset is called *low utility*. Mining high utility itemsets is much more challenging than mining frequent itemsets, because the *downward closure property* [3, 12] in frequent itemset mining does not hold in utility mining.

Several algorithms have been proposed for mining HUIs, including *IHUP* [5], *Two-Phase*, *IIDS* [18], *TWU-Mining* [27], and *UP-Growth* [26]. Most of them utilize the *TWDC* (*Transaction-Weighted Downward Closure*) property and adopt the *TWU* (*Transaction-Weighted Utilization*) model to find high utility itemsets. In general, the general TWU model consists of two phases. In phase I, potential high utility itemsets are found from the database. In phase II, the exact utilities of the potential high utility itemsets are computed by scanning the database and high utility itemsets are identified from the set of potential high utility itemsets.

Although the above studies perform well in many applications, they can only handle itemsets and do not consider the sequential data and the ordering relationships between items. Mining high utility patterns from sequential data is a more challenging task. The integration of utility and sequential pattern mining has taken place very recently. We only found four papers [5, 6, 27, 34] on this topic. Ahmed et al. integrated the concept of utility mining with sequential pattern mining and proposed *US* and *UI* algorithms for mining *high utility sequential patterns* [7]. Shie et al. proposed the UMSP algorithm [25] for mining *high utility mobile sequential patterns* in mobile environment. Ahmed et al. designed an algorithm for mining *high utility access sequence*s from web log data [6]. Recently, Yin et al. argued that the problem definition in [6] is rather specific and they proposed a generic framework for high utility sequence analysis and an efficient algorithm named *USpan* [32] for mining high utility sequential patterns. From the above related works, we can observe that only very preliminary works have been done on mining high utility patterns from sequential data. For the topic of *high utility episode mining*, we found that there is only one related paper in the literature [10]. But it only considers the external utility of the event (e.g. importance/weight/unit profit). It did not consider the case of complex event sequence and the internal utility of the event (e.g. quality/quantity/ appearance count).

3. HIGH UTILITY EPISODE MINING

In this subsection, we first explain how we incorporate the concept of utility mining into episode mining and propose a new framework for *high utility episode mining*. Then we present an efficient algorithm named *UP-Span* (*Utility ePisodes mining by SPANning prefixes*) and effective strategies for mining the complete set of high utility episodes in complex event sequences.

3.1 High Utility Episode Mining

Let N^+ be a set of time points and $CS = <(tSE_1, T_1), (tSE_2, T_2),\ldots, (tSE_n, T_n)>$ be a *complex event sequence* with n time points, where each simultaneous event set tSE_i is associated with a time point $T_i \in N^+$ and $T_i < T_j$, for all $1 \le i < j \le n$. In high utility episode mining, each event $E_i \in \varepsilon$ is associated with a positive number $p(E_i, CS)$, called its *external utility*. Each event E_j in a simultaneous event set tSE_i at the time point T_i is associated with a positive number $q(E_j, T_i)$, called its *internal utility*. For example, Figure 3 shows a complex event sequence with internal utility and Table 1 shows the external utilities of events.

Definition 13 (Utility of an event at a time point). The *utility of an event E_j at a time point T_i* is defined as $u(E_j, T_i) = p(E_j, CS) \times q(E_j, T_i)$. For example, the utility of the event (A) at the time point T_1 is $u((A), T_1) = p((A), CS) \times q((A), T_1) = (1\times2) = 2$.

Definition 14 (Utility of a simultaneous event set at a time point). The *utility of a simultaneous event set $SE = (E_1, E_2,\ldots, E_k)$ at a time point T_i* is defined as $u(SE, T_i) = \sum_{j=1}^{k} u(E_j, T_i)$. For example, the utility of the simultaneous event set (AB) at the time point T_1 is $u((AB), T_1) = u((A), T_1) + u((B), T_1) = (2+2) = 4$.

Definition 15 (Total utility of database complex event sequence). The *total utility of a complex event sequence CS* is defined as $u(CS) = \sum_{i=1}^{n} u(SE_i, T_i)$. For example, complex event sequence depicted in Figure 3 is $u(CS) = u((AB), T_1) + u((BC), T_2) + u((C), T_3) + u((AB), T_5) + u((CD), T_6) + u((C), T_7) = (4 + 8 + 3 + 4 + 18 + 3) = 40$.

Definition 16 (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 $\alpha = <(SE_1), (SE_2),\ldots, (SE_k)>$, 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 $u(\alpha, mo(\alpha)) = \sum_{i=1}^{k} u(SE_i, T_i)$. For example, the utility of $<(AB),(C)>$ w.r.t the $mo(<(AB),(C)>) = [1, 2]$ is $(4 + 6) = 10$.

Definition 17 (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 $1 \le i \le k$. The *utility value of the episode α in a complex event sequence CS* is defined as $uv(\alpha, CS) = \sum_{i=1}^{k} u(\alpha, TI_k)$. The *utility of α* is defined as $u(\alpha) = (uv(\alpha)/ u(CS))$. For example, the utility of the episode $<(AB),(C)>$ is $u(<(AB),(C)>) = (uv(<(AB),(C)>) / u(CS)) = (20/40) = 50\%$.

Definition 18 (High Utility Episode; HUE). An episode is a *high utility episode* (abbreviated as *HUE*), iff its utility is no less than a user-specified *minimum utility threshold min_utility*. Otherwise, the episode is a *low utility episode*.

Problem statement. Given a user-specified minimum utility threshold *min_utility* and a complex event sequence *CS* with external utility and internal utility of events, the problem of high utility episode mining is to discover all the episodes having a utility no less than *min_utility*.

Definition 19 (Maximum time duration). Let *MTD* be a user-specified *maximum time duration* and $mo(\alpha) = [T_s, T_e]$ be a minimal occurrence of the episode α. The minimal occurrence $mo(\alpha)$ is said to satisfy the *maximum time duration constraint* iff $(T_e - T_s + 1) \le MTD$.

Definition 20 (Simultaneous and serial concatenations). Let $\alpha = <(SE_1), (SE_2),\ldots, (SE_x)>$ and $\beta = <(SE_1'), (SE_2'),\ldots, (SE_y')>$ be episodes. The *simultaneous concatenation* of α and β is defined as $simul\text{-}concat(\alpha, \beta) = <(SE_1), (SE_2),\ldots, (SE_x \cup SE_1'), (SE_2'),\ldots, (SE_y')>$. The *serial concatenation* of α and β is defined as $serial\text{-}concat(\alpha, \beta) = <(SE_1), (SE_2),\ldots, (SE_x), (SE_1'), (SE_2'),\ldots, (SE_y')>$.

Definition 21 (Episode-Weighted Utilization of an episode w.r.t a minimal occurrence). Let $mo(\alpha) = [T_s, T_e]$ be a minimal occurrence of the episode $\alpha = <(SE_1), (SE_2),\ldots, (SE_{k-1}), (SE_k)>$, where each simultaneous event set $SE_i \in \alpha$ is associated with a time point $T_i (1 \le i \le k)$ and $mo(\alpha)$ satisfies *MTD*. *The episode-weighted utilization of* α w.r.t $mo(\alpha)$ is defined as $EWU(\alpha, mo(\alpha)) = [\sum_{i=1}^{(k-1)} u(SE_i, T_i) + \sum_{i=e}^{(s+MTD-1)} u(tSE_i, T_i)] / u(CS)$, where tSE_i is the simultaneous event set at the time point T_i in *CS*.

For example, if *MTD* = 4, the *EWU* of the episode $\alpha = <(C), (A)>$ w.r.t $mo(<(C), (A)>) = [3, 5]$ is $EWU(<(C),(A)>, [3, 5]) = [u((C), T_3)] + [u((AB), T_5)) + u((CD), T_6)] = 25$.

Definition 22 (Episode-Weighted Utilization of an episode). Let $moSet(\alpha) = [TI_1, TI_2,\ldots, TI_k]$ be the set of all the minimal occurrences of α, where each minimal occurrence $TI_i \in moSet(\alpha)$ satisfies *MTD* for $1 \le i \le k$. *The episode-weighted utilization of α in a complex event sequence CS* is defined as $EWU(\alpha) = ((\sum_{i=1}^{k} EWU(\alpha, TI_i)) / u(CS))$.

For example, when *MTD* = 3, the *EWU* of the episode $\alpha = <(A),(C)>$ is $EWU(<(A),(C)>) = [u((AB), T_1) + u((BC), T_2) + u((C), T_3)] + [u((AB), T_5) + u((CD), T_6) + u((C), T_7)]/u(CS) = 40/40$.

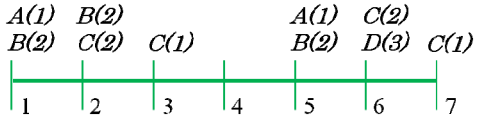


Figure 3. Complex event sequence with internal utility

Table 1. External utilities of events

Event	*A*	*B*	*C*	*D*
External utility	2	1	3	4

Definition 23 (High Weighted Utilization Episode; HWUE). An episode is called *High Weighted Utilization Episode* (abbreviated as *HWUE*) iff its *EWU* is no less than the minimum utility threshold *min_utility*.

Theorem 1 (Episode-Weighted Downward Closure property). Let α and β be episodes, and $\gamma = simult\text{-}concat(\alpha, \beta)$ or $serial\text{-}concat(\alpha, \beta)$. The *Episode-Weighted Downward Closure* (abbreviated as *EWDC*) property states that if $EWU(\alpha) < min\_utility$, γ is a low utility episode.

Proof. Let $moSet(\alpha) = [TI_1, TI_2,\ldots, TI_x]$, $moSet(\gamma) = [TI_1', TI_2',\ldots, TI_y']$. Because $\gamma = simult\text{-}concat(\alpha, \beta)$ or $serial\text{-}concat(\alpha, \beta)$, $|moSet(\alpha)| \ge |moSet(\gamma)|$ [21, 31]. According to the Definition 22, $EWU(\alpha) = ((\sum_{i=1}^{x} EWU(\alpha, TI_i)) / u(CS)) \ge EWU(\gamma) = ((\sum_{j=1}^{y} EWU((\gamma, TI_j)) / u(CS)) \ge u(\gamma)$. If $EWU(\alpha) < min\_utility$, $u(\gamma) < min\_utility$, which yields that γ is low utility (Definition 18).

Table 2. Minimal occurrences, EWUs and utilities of 1-episodes in the complex event sequence of Figure 3

Global Event	Minimal occurrences	EWU	Utility
A	{[1,1], [5, 5]}	40/40	4/40
B	{[1,1], [2,2], [5,5]}	51/40	6/40
C	{[2,2], [3,3], [6,6], [7,7]}	42/40	18/40
D	{[6,6]}	21/40	12/40

Table 3. Minimal occurrences, EWUs and utilities of local 1-episodes in the <(*A*)>-projected database

<(*A*)>-projected database			
Local Event	Minimal occurrences	EWU	Utility
(_*B*)	{[1,1], [5,5]}	40/40	8/40
B	{[1,2]}	13/40	4/40
C	{[1,2], [5,6]}	36/40	16/40
D	{[5,6]}	23/40	14/40

3.2 Efficient Mining of High Utility Episodes

This subsection introduces an algorithm named *UP-Span* (*Utility ePisodes mining by Spanning prefixes*) for efficiently discover high utility episodes in a complex event sequence. The proposed algorithm adopts the prefix-growth paradigm [12, 24]. Following that, two efficient strategies that greatly enhance the performance are introduced.

Pseudo code 1 shows the main procedure of the UP-Span algorithm. The inputs of the UP-Span algorithm are: (1) a complex event sequence *CS*, (2) minimum utility threshold *min_utility* and (3) maximum time duration *MTD*. The algorithm scans the complex event sequence once to find 1-episodes and catching their associated minimal occurrences (Line 1-2). The EWUs and exact utilities of 1-episodes can be calculated according to the Definition 17 and 22. For example, Table 2 shows the minimal occurrences, EWUs and utilities of all 1-episodes in Figure 3 when *MTD* = 3.

For each 1-episode α (also called *global event*), if $EWU(\alpha)$ is no less than *min_utility*, α is identified as a *HWUE* of length one (Definition 23). Then, the algorithm explores the search space of high utility episodes containing α as prefix. The prefix α is spanned by executing the *MiningHUE* procedure (Line 3-5). There are two sub-procedures *MiningSimultHUE* and *MiningSerialHE* in the procedure *MiningHUE*. The sub-procedure *MiningSimultHUE* aims at finding the simultaneous events that are related to α. The sub-procedure *MiningSerialHUE* aims at finding the serial events related to α (Line 7-9).

ALGORITHM: UP-Span

Input: (1) *CS*: complex event sequence;
(2) *min_utility*: minimum utility threshold;
(3) *MTD*: maximum time duration;
Output: *HUE_Set*: The complete set of high utility episodes;
01. Scan *CS* once to find high utility 1-epsiodes and calculate
02. their *EWUs* and catch the associated minimal occurrences;
03. ***for each*** global event α ***do***
04. ***if*** ($EWU(\alpha) \geq min\_utility$) **then**
05. { ***MiningHUE***(α, *moSet*(α), *MTD, min_utility*);}
06.
Procedure *MiningHUE*(*episode* α, *moSet*(α), *MTD, min_utility*)
07. ***MiningSimultHUE***(α, *moSet*(α), *MTD, min_utility*);
08. ***MiningSerialHUE***(α, *moSet*(α), *MTD*, *min_utility*);
09.

Pseudo code 1. Algorithm *UP-Span*

ALGORITHM: *MiningSimultHUE*

Input: (1) α: episode;
(2) *moSet*(α): all minimal occurrences of α
(3) *MTD*: maximum time duration
(4) *min_utility*: minimum utility threshold;
Output: The set of high utility simultaneous episodes w.r.t prefix α;
01. ***for each*** $mo(\alpha) = [T_s, T_e] \in moSet(\alpha)$ ***do***
02. $SES = \{e \mid$ event e occurs at $T_e\}$;
03. ***for each*** *event* $e \in SES$ ***do***
04. $\beta = simult\text{-}concat(\alpha, e)$;
05. Let $occ(\beta) = [T_s, T_e]$;
06. ***if*** ($occ(\beta)$ is a minimal occurrence in $moSet(\beta)$) ***then***
07. { $moSet(\beta) = moSet(\beta) \cup occ(\beta)$;}
08.
09. ***for each*** simultaneous event e in α-projected database ***do***
10. $\beta = simult\text{-}concat(\alpha, e)$;
11. $moSet(\beta)$: = ***Repair_moSet***($moSet(\beta)$);
12. ***if*** ($u(\beta) \geq min\_utility$) ***then*** {$HUE\_Set = HUE\_Set \cup \beta$; }
13. ***if*** ($EWU(\beta) \geq min\_utility$) ***then***
14. { ***MiningHUE***(β, $mo(\beta)$, *MTD, min_utility*); }

Pseudo code 2. Procedure *MiningSimultHUE*

ALGORITHM: *MiningSerialHUE*

Input: (1) α: episode;
(2) *moSet*(α): all minimal occurrences of α
(3) *MTD*: maximum time duration
(4) *min_utility*: minimum utility threshold;
Output: The set of high utility serial episodes w.r.t prefix α;
01. ***for each*** $mo(\alpha) = [T_s, T_e] \in moSet(\alpha)$ ***do***
02. ***for each*** time point t between $[T_e+1, T_s+MTD-1]$ ***do***
03. $NES = \{e \mid$ event e occurs at time point $t\}$;
04. ***for each*** *event* $e \in NES$ ***do***
05. $\beta = serial\text{-}concat(\alpha, e)$;
06. Let $occ(\beta) = [T_s, t]$;
07. ***if*** ($occ(\beta)$ is a minimal occurrence in $moSet(\beta)$) ***then***
08. { $moSet(\beta) = moSet(\beta) \cup occ(\beta)$;}
09.
10. ***for each*** serial event e in projected database of α ***do***
11. $\beta = serial\text{-}concat(\alpha, e)$;
12. $moSet(\beta)$: = ***Repair_moSet***($moSet(\beta)$);
13. ***if*** ($u(\beta) \geq min\_utility$) ***then*** {$HUE\_Set = HUE\_Set \cup \beta$; }
14. ***if*** ($EWU(\beta) \geq min\_utility$) ***then***
15. { ***MiningHUE***(β, $moSet(\beta)$, *MTD*, *min_utility*); }

Pseudo code 3. Procedure *MiningSerialHUE*

Pseudo code 2 shows the procedure of the *MiningSimultHUE*, which is performed as follows. For each minimal occurrence $mo(\alpha) = [T_s, T_e]$ in $moSet(\alpha)$, the algorithm collects all events that occur at the time point T_e into the set *SES* (*S*imultaneous *E*vents *S*et) (Line 1-2). For each event e in the set *SES*, the algorithm performs the simultaneous concatenation of α and e to form an episode β (Line 4). Then, the variable $occ(\beta)$ is set to $[T_s, T_e]$ (Line 5). If $occ(\beta)$ is a minimal occurrence in the set of current minimal occurrences, $occ(\beta)$ is added into the set of minimal occurrence of β (Line 6-7). After that, events that *s*imultaneously occur with α, their minimal occurrences are stored in the *projected database of* α (abbreviated as *α-PB*). For each simultaneous event e in *α-PB*, we perform simultaneous concatenation operation on α and e to form the episode β (Line 11). For each such episode β, the function *Repair_moSet* is called to find the complete set of minimal occurrences of β since the

current *moSet*(β) does not capture the complete set of minimal occurrences of β. After that, all the minimal occurrences of β are collected into *moSet*(β). Given the information contained in *moSet*(β), the utility and *EWU* of β can be calculated according to Definitions 17 and 21. For example, Table 3 shows the minimal occurrences, *EWU* values and utility values of local 1-episodes in the <(*A*)>-projected database when *MTD* = 3. The events in the first row of Table 3 are simultaneous events of the episode <(A)>. After the calculation, if the utility of β is no less than *min_utility*, β is high utility and it is collected into the set *HUE_Set*. If *EWU*(β) is no less than *min_utility*, the procedure *MiningHUE* is called to find high utility episodes w.r.t. the prefix β.

Pseudo code 3 shows the procedure of the *MiningSerialHUE*, which is performed as follows. For each minimal occurrence $mo(\alpha) = [T_s, T_e]$ in *moSet*(α), we collect all events that occur between the time interval $[T_e+1, T_s+MTD-1]$ into the set *NES* (*N*ext *E*vents *S*et) (Line 1-3). For each event *e* in the set *NES*, we perform serial concatenation operation on α and *e* to form an episode β = *simult-concat*(α, *e*) (Line 5). Then, the variable *occ*(β) is set to $[T_s, t]$, where *t* is a time point between the time interval $[T_e+1, T_s+MTD-1]$ (Line 7). If *occ*(β) is a minimal occurrence in the set of current minimal occurrences, *occ*(β) is added into the set of minimal occurrences of β (Line 7-8). After that, events that serially occur after α, and their current minimal occurrences are stored in the *α-PB*. For each serial event *e* in the *α-PB*, the algorithm performs serial concatenation of α and *e* to form an episode β. For each such episode β, the algorithm calls the function *Repair_moSet* to finds the complete set of minimal occurrences of β. After that, all the minimal occurrences of β are collected into the variable *moSet*(β). With the information of *moSet*(β), the utility and *EWU* of β can be calculated according to the Definitions 17 and 22. For example, the last three rows of Table 3 shows minimal occurrences, *EWUs* and utilities of the three serial events of the episode <(A)>. After the calculation, if the utility of β is no less than the *min_utility*, β is a high utility episode and it is collected into the set *HUE_Set*. If the *EWU*(β) is no less than the *min_utility*, the procedure *MiningHUE* is called to find the high utility episodes w.r.t. the prefix β.

Then, we present two effective strategies named *DGE* (*D*iscarding *G*lobal unpromising *E*vents) and *DLE* (*D*iscarding *L*ocal unpromising *E*vents), which are based on the following definitions.

Definition 24 (Promising event). An event *e* is a *promising event* iff $EWU(e) \geq min\_utility$. Otherwise it is an *unpromising event*.

Property 2. Let α be an unpromising event and β be an episode, Any super-episode γ of α such that γ = *simult-concat*(α, β) or γ = *serial-concat*(α, β) is low utility.

Rationale. The property holds by EWDC property (Theorem 1).

Strategy 1 (Discarding Global unpromising Events; DGE). Discard global unpromising events and their exact utilities from the complex event sequence and related EWUs.

Rationale. By the Theorem 1, unpromising events play no role in high utility episodes. Therefore, global unpromising events can be removed from the complex event sequence and their utilities can be ignored in the calculation of the estimated utilities of episodes.

Strategy 2 (Discarding Local unpromising Events; DLE). Discard local unpromising events and their exact utilities from the projected database and related EWUs.

Rationale. By the Theorem 1, local unpromising events play no role in high utility episodes. Therefore, local unpromising events can be removed from the projected database and their utilities can be ignored in the calculation of the estimated utilities of episodes.

4. EXPERIMENTAL EVALUATION

In this section, we evaluate the performance of the proposed algorithms. Experiments were performed on a computer with a 3.40 GHz Intel Core 2 Processor with 4 gigabytes of memory, running on Windows 7. All of the algorithms are implemented in Java. Both synthetic and real datasets are used to evaluate the performance of the algorithms. Synthetic datasets were generated by using the IBM data generator [3]. The parameters of the generator are described as follows: *D* is the total number of time points; *T* is the average size of a simultaneous event set at a time point; *N* is the number of distinct events; *I* is the average size of maximal potential episodes. The internal utility and external utility values are generated using the settings used in [26, 28, 29]. Different types of real world datasets were used in the experiments. *Foodmart*, a small sparse dataset, was acquired from Microsoft foodmart 2000 database [35]; *Retail* was obtained from FIMI Repository [34]. *ChainStore*, a large dataset, was obtained from NU-MineBench 2.0 [36]. Note that these three datasets are sometimes viewed as transaction databases but they can be considered as a single complex sequence by regarding each item as an event and each transaction as a simultaneous event set. The Foodmart and ChainStore already contain unit profits (external utility) and purchased quantities (internal utility). For the Retail dataset, unit profits for items are generated between 1 and 1,000 by using a log-normal distribution and quantities of items are generated randomly between 1 and 5, as in [26, 28, 29]. Table 4 shows the characteristics of the datasets in the experiments. To evaluate the performance of the proposed algorithms, we compare four versions of the algorithm named as follows. The baseline algorithm without strategies DGE and DLE is denoted as UP-Span(Baseline). The algorithm only applying the strategy DGE is denoted as UP-Span(DGE). The algorithm only applying the strategy DLE is denoted as UP-Span(DLE). Lastly, the algorithm UP-Span(DGE+DLE) uses both DGE and DLE strategies.

Table 4. Statistical information about different datasets

Dataset	#Trans	#Items	Avg. Length.
T12I8N1KQ5D10K	10,000	1,000	12
Foodmart	4,141	1,559	4.4
Retail	88,162	16,470	10.3
ChainStore	1,112,949	46,086	7.3

4.1 Evaluation on Synthetic Dataset

We first discuss the performance of the algorithms on the synthetic dataset T12I8N1KQ5D10K. Figure 4 shows the number of candidates and high utility episodes on T12I8N1KQ5D10K under varied minimum utility thresholds when the maximum time duration is set to eight. In Figure 4, there is no high utility episode produced when the minimum utility threshold is lower than 30%.

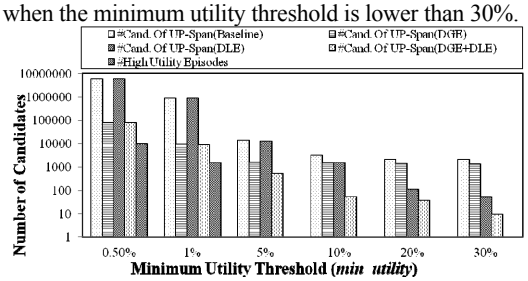


Figure 4. The number of candidates on T12I8N1KQ5D10K dataset under different minimum utility thresholds

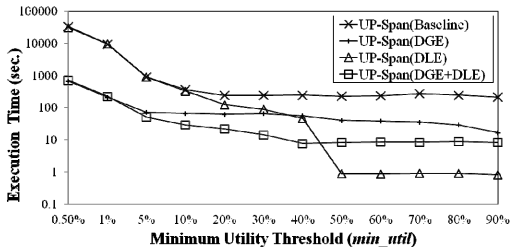


Figure 5. The execution time on T12I8N1KQ5D10K dataset under different minimum utility thresholds

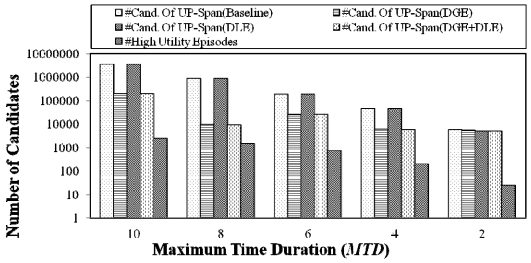


Figure 6. Number of candidates and high utility episodes on T12I8N1KQ5D10K under varied maximum time durations

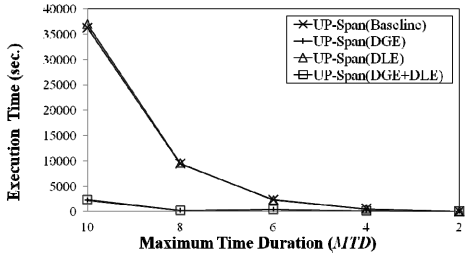


Figure 7. The execution time on T12I8N1KQ5D10K dataset under different maximum time durations

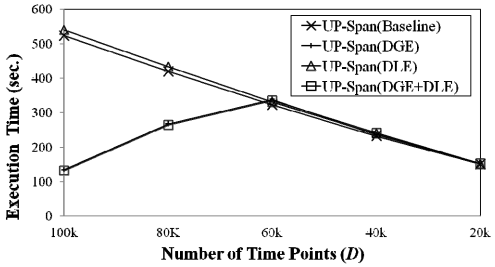


Figure 8. Execution time on T12I8N1KQ5D*x*K dataset (*x* is varied from 20 to 100)

As shown in Figure 4, UP-Span(DGE+DLE) generates much fewer candidates than UP-Span(Baseline). The reason is that strategy DGE effectively reduces the number of candidates by removing global unpromising events and their utilities from the complex event sequence. Although both strategies reduce the number of candidates, the effectiveness of the strategy DGE is better than that of the strategy DLE on this dataset. In the Figure 4, when the minimum utility threshold is less than 1%, the number of candidates generated by UP-Span(DGE+DLE) is about 100 times smaller than the number of candidates generated by UP-Span(Baseline).

Figure 5 shows the execution time on T12I8N1KQ5D10K under varied minimum utility thresholds when the maximum time duration is set to eight. As shown in Figure 5, UP-Span(Baseline) is the worst and UP-Span(DGE+DLE) has the best performance. In Figure 5, UP-Span(DLE) runs faster than UP-Span(Baseline) over 100 times when the minimum utility threshold is higher than 50%. UP-Span(DLE) and UP-Span(Baseline) follow a similar trend when the threshold is less than 10%. It is because the UP-Span(DLE) performs additional processing to decrease the overestimated utilities of the episodes but the there are few local unpromising events in the projected databases. When the threshold is lower than 5%, UP-Span(DGE+DLE) runs faster than UP-Span(baseline) about 10 times. By the above observation, we show that the overall performance of UP-Span(DGE+DLE) outperforms UP-Span(Baseline).

Figure 6 shows the number of candidates and high utility episodes of the algorithms on T12I8N1KQ5D10K under varied maximum time durations. In this experiment, the threshold is set to 1%. As shown in Figure 6, the number of candidates grows rapidly when the maximum time duration increases. In Figure 6, we can see that UP-Span(DGE+DLE) generates much fewer candidates than UP-Span(Baseline). When the maximum time duration is set to ten, UP-Span(DGE+DLE) generates about 10 times less candidates than UP-Span(Baseline). Figure 7 shows the execution time of the algorithms on T12I8N1KQ5D10K under various maximum time durations. As shown in Figure 7, UP-Span(DGE+DLE) and UP-Span(DGE) run about 15 times faster than UP-Span(Baseline) and UP-Span(DLE) because the former two algorithms produce much fewer candidates than the later two algorithms.

Then, we test the scalability of the algorithms on different lengths of complex event sequences. In this experiment, the maximum time duration and the minimum utility threshold are set to four and 10%. The number of time points in the complex event sequence is varied from 20K to 100K. Figure 8 shows the execution time for this experiment. As shown in Figure 8, UP-Span(DGE) and UP-Span(DGE+DLE) have better scalability than UP-Span(Baseline) and UP-Span(DLE) when the number of time points increases. When the number of time points is 100K, UP-Span(DGE) and UP-Span(DGE+DLE) run about 5 times faster than the UP-Span(Baseline) and UP-Span(DLE).

4.2 Evaluation on Real Dataset

In this section, we compare the performance of the algorithms on real datasets. We first show the evaluation on Foodmart, which is a small dataset with 1,559 distinct events. Figure 9 shows the execution time on the Foodmart dataset under different minimum utility thresholds. As shown in Figure 9, all the algorithms have good performance but UP-Span(Baseline) is the slowest and the winner is UP-Span(DLE). On this dataset, the strategy DLE performs better than the strategy DGE. The strategy DLE effectively reduces the number of candidates by removing local unpromising events and their utilities from the projected databases. The execution time of UP-Span(DGE+DLE) is affected by the extra operations performed by the strategy DGE, and thus it runs slower than UP-Span(DLE). When the minimum utility threshold is set to 10%, the execution time of UP-Span(DGE) is close to that of UP-Span(Baseline) since there are few global unpromising events that can be discarded from the complex event sequence.

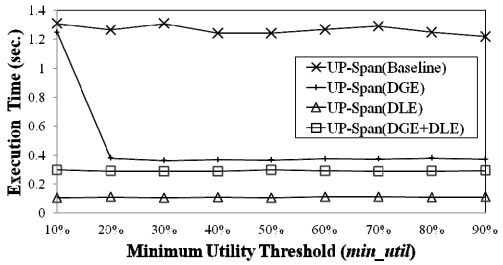


Figure 9. Execution time on Foodmart dataset under different minimum utility thresholds

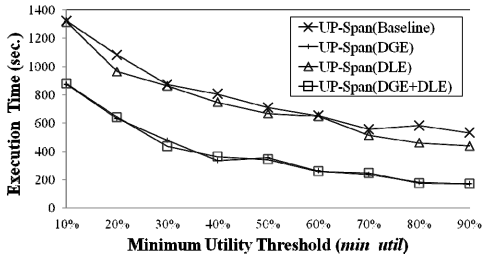


Figure 10. Execution time on Retail dataset under different minimum utility thresholds

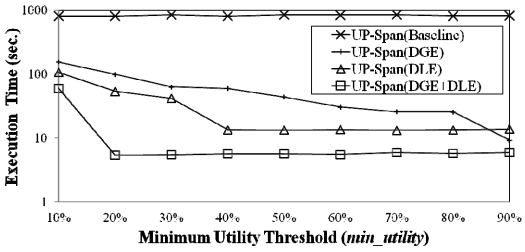
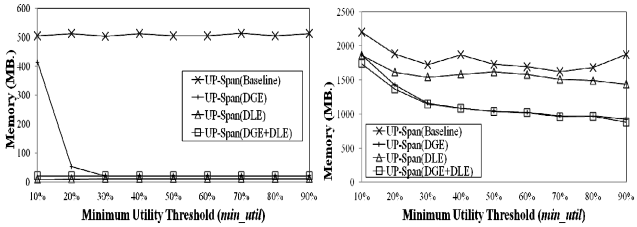


Figure 11. Execution time on ChainStore dataset under different minimum utility thresholds



(a) Foodmart dataset (b) Retail dataset

Figure 12. Memory consumptions of the algorithms

We then evaluate the performance of the algorithms on the Retail dataset. There are 16,470 distinct events in the dataset and the average length of the transactions is longer than that of the Foodmart dataset. Figure 10 shows the execution time of the algorithms on the Retail dataset under different minimum utility thresholds. The results show that UP-Span(DGE+DLE) and UP-Span(DGE) follow a similar trend and they run faster than the UP-Span(Baseline) and UP-Span(DLE).

Figure 11 shows the execution time of the algorithms on the ChainStore dataset under different minimum utility thresholds. In this experiment, the maximum time duration is set to four. As shown in Figure 11, UP-Span(DGE+DLE) is the winner and UP-Span(Baseline) has the worst performance. When the threshold is higher than 20%, UP-Span(DGE+DLE) runs faster than UP-Span(Baseline) over 100 times. When the minimum utility threshold is set to 10%, UP-Span incorporated with strategies run faster than UP-Span(Baseline) over 10 times. Figure 11 also shows that UP-Span incorporated with strategies has good scalability even for large database with large number of events. The overall performance of UP-Span with strategies is better than UP-Span(Baseline).

4.3 Memory Consumption

We evaluate the memory consumption of the algorithms on Foodmart and Retail datasets. Figure 12(a) shows the memory consumption of the algorithms on Foodmart dataset under different minimum utility thresholds. We can observe that UP-Span with strategies uses less memory than UP-Span(Baseline) since the proposed strategies effectively reduce the number of candidates and the number of projected databases. Figure 12(b) shows the memory consumption of the algorithms on the Retail dataset under different minimum utility thresholds. Overall, results show that the best algorithm is UP-Span(DGE+DLE) and the worst one is UP-Span(Baseline).

4.4 Summarization and Discussion

We summarize results of the above experiments and compare characteristics of different algorithms. The experimental results show that our approach outperforms the baseline approach on both real and synthetic datasets. For example, UP-Span(DGE+DLE) runs over 100 times faster than the baseline approach on the ChainStore dataset when the minimum utility threshold is higher than 20%. Depending upon the characteristics of the datasets, the most effective pruning strategy can be different. For example, for the Foodmart dataset, the pruning of local unpromising events (strategy DLE) gives the best performance, while for Retail dataset, it is the pruning of global unpromising events (strategy DGE). UP-Span(DGE+DLE) provides the most consistent and robust performance as it takes both types of pruning strategies into considerations, while UP-Span(DGE) and UP-Span(DLE) perform well only on one of the datasets as it incorporates just one type of pruning strategies. UP-Span(Baseline) always has the worst performance as it does not utilize the DGE and DLE pruning strategies.

There are three reasons why our approach has good scalability and high performance on large databases. First, our approach is not Apriori-based. It discovers patterns by recursively growing patterns one item/event at a time. This avoids well-known drawbacks of Apriori-like approaches: (1) generating too many unnecessary candidates and (2) repeatedly scanning the original database. Second, our approach finds (k+1)-episodes and their occurrences by using minimal occurrences of related k-episodes instead of all the occurrences, which leads to faster calculation and less memory consumption. Third, our approach finds high utility episodes in only one phase, as opposed to most high utility pattern mining algorithms [5, 7, 26, 33], which require collecting candidates and performing an additional database scan to calculate their exact utilities. This facilitates the performance of the mining task in terms of time and space.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we incorporate the concept of utility mining into episode mining and propose a novel framework for mining high utility episodes in complex event sequences, which has not been explored so far. In the proposed framework, we consider the external utility and internal utility of events to measure the utility of episodes. We take the scenario of the complex event sequences into consideration for mining high utility episodes containing simultaneous events, which not only provides users with episodes with high utilities (e.g. high profits) but also more information about the relationships between episodes. We proposed a new algorithm named *UP-Span* (*Utility ePisodes mining by Spanning prefixes*) for efficiently mining the complete set of the high utility episodes. We successfully extend the TWU model to episode mining and propose the *EWU* (*Episode-Weighted Utilization*) model to facilitate the mining task of high utility episode mining. Two effective strategies, namely *DGE* (*Discarding Global unpromising Events*) and *DLE* (*Discarding Local unpromising Events*), are also proposed and incorporated with the UP-Span algorithm, which not only reduce the number of candidates produced in the mining processes but also enhance the performance of them mining task in terms of execution time and memory consumption. Experimental results on both real and synthetic datasets show that UP-Span has good scalability and outperforms the baseline approach substantially, especially under

higher minimum utility threshold (e.g. UP-Span runs faster than the baseline approach over 100 times on ChainStore dataset when the minimum utility threshold is higher than 20%).

Although we first incorporate the concept of utility mining with episode mining and address the problem of *high utility episode mining* in this work, it still leaves ample room for exploration in the future work. For example, in this paper, we only consider serial episodes containing simultaneous events and do not consider other types of episode such as *injective episodes* [22], *parallel episodes* [22], *closed episode*s [19] and so on. In addition, there are many different ways to calculate the occurrence of episode, such as *window-based occurrence* [11, 22], *non-overlapped/overlapped minimal occurrence* ect., which can be addressed in the future work. Mining high utility episodes from event sequences is a novel and challenging problem. Related research topics ranging from problem definition to algorithm improvement and applications are worthwhile to be explored in the future.

ACKNOWLEDGMENTS

This work is supported in part by National Science Council, Taiwan, R.O.C. under grant no. NSC101-2221-E-006-255-MY3 and NSF through grants IIS-0905215, CNS-1115234, IIS-0914934, DBI-0960443, and OISE-1129076, and US Department of Army through grant W911NF-12-1-0066.

REFERENCES

[1] J. Ayres, J. Flannick, J. Gehrke and T. Yiu. Sequential PAttern Mining using a bitmap representation. In Proc. of IEEE Int'l Conf. on Data Mining (ICDM), pp. 429-435, 2002.

[2] A. Ng, and Ada Wai-Chee Fu, Mining Frequent Episodes for Relating Financial Events and Stock Trends, In Proc. of the 7th Pacific-Asia conference on Advances in knowledge discovery and data mining (PAKDD), pp. 27-39, 2003.

[3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In Proc. of the 20th Int'l Conf. on Very Large Data Bases, pp. 487-499, 1994.

[4] R. Agrawal and R. Srikant, Mining Sequential Patterns. In Proc. of Int'l Conf. on Data Engineering. (ICDE), pp. 3-14, 1995.

[5] C. F. Ahmed, S. K. Tanbeer, B.-S. Jeong and Y.-K. Lee. Efficient Tree Structures for High-utility Pattern Mining in Incremental Databases. In IEEE Transactions on Knowledge and Data Engineering, Vol. 21, Issue 12, pp. 1708-1721, 2009.

[6] C. F. Ahmed, S. K. Tanbeer and B. Jeong. A Framework for Mining High Utility Web Access Sequences. In IETE Journal, Vol. 28, Issue 1, pp. 3-16, 2011.

[7] C. F. Ahmed, S. K. Tanbeer and B. Jeong. A Novel Approach for Mining High-Utility Sequential Patterns in Sequence Databases, ETRI Journal, Vol. 32, no.5, pp.676-686, 2010.

[8] R. Chan, Q. Yang and Y. Shen. Mining high-utility itemsets. In Proc. of Third IEEE Int'l Conf. on Data Mining, pp. 19-26, Nov., 2003.

[9] R. Gwadera, M. J. Atallah, and W. Szpankowski. Reliable Detection of Episodes in Event Sequences, Knowledge and Information System, Vol. 7, pp. 415-437, 2005.

[10] T. Guo, S. Lin, Y. Wang and J. Qiao. A new Framework for Detecting High-Utility Episodes in Event Sequence. In Proc. of the IEEE Int'l Conf. on Oxide Materials for Electronic Engineering (OMEE), pp.370-373, 2012.

[11] K.-Y. Huang, and C.-H. Chang, Efficient Mining of Frequent Episodes from Complex Sequences, Information Systems, Vol. 33, pp. 96-114, 2008.

[12] J. Han, J. Pei and Y. Yin. Mining frequent patterns without candidate generation. In Proc. of the ACM-SIGMOD Int'l Conf. on Management of Data, pp. 1-12, 2000.

[13] H.-F. Li, H.-Y. Huang, Y.-C. Chen, Y.-J. Liu, S.-Y. Lee. Fast and Memory Efficient Mining of High Utility Itemsets in Data Streams. In Proc. of the 8th IEEE Int'l Conf. on Data Mining, pp. 881-886, 2008.

[14] Y. Liu, W. Liao, and A. Choudhary. A fast high-utility itemsets mining algorithm. In Proc. of the Utility-Based Data Mining Workshop, 2005.

[15] M. Liu and J. Qu. Mining High Utility Itemsets without Candidate Generation. In Proc. Of the ACM Int'l Conf. on Information and Knowledge Management (CIKM), pp. 55-64, 2012.

[16] S. Laxman, P. S. Sastry, and K. P. Unnikrishnan, A Fast Algorithm for Finding Frequent Episodes in Event Streams, In Proc. of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 410-419, 2007.

[17] J. Liu, K. Wang, and B. C. M. Fung. Direct Discovery of High Utility Itemsets without Candidate Generation. In Proc. of the IEEE Int'l Conf. on Data Mining (ICDM), 6 pages, short paper, 2012.

[18] Y.-C. Li, J.-S. Yeh and C.-C. Chang. Isolated Items Discarding Strategy for Discovering High-utility Itemsets. In Data & Knowledge Engineering, Vol. 64, Issue 1, pp. 198-217, 2008.

[19] N. Tatti, and B. Cule. Mining closed episodes with simultaneous events. In Proc. of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 1172-1180, 2012.

[20] N. Tatti, and J. Vreeken, The Long and the Short of It: Summarizing Event Sequences with Serial Episodes, In Proc. of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2012.

[21] N. Meger, C. Leschi, N. Lucas, and C. Rigotti, Mining episode rules in STULONG dataset, In Proc. of the ECML/PKDD2004 Discovery Challenge, 2004, pp. 1-12.

[22] H. Mannila , H. Toivonen, and A. I. Verkamo, Discovery of Frequent Episodes in Event Sequences, Data Mining and Knowledge Discovery, Vol. 1(3), pp. 259-289, 1997.

[23] D. Patnaik, P. Butler, N. Ramakrishnan, L. Parida, B. J. Keller, and A. Hanauer, Experiences with Mining Temporal Event Sequences from Electroinic Medical Records, In Proc. of ACM SIGKDD conference on Advances in knowledge discovery and data mining (KDD), pp. 360-368, 2011.

[24] J. Pei, J. Han, B. Mortazavi-Asl. J. Wang, H. Pinto, Q. Chen, U. Dayal and M. C. Hsu. PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. In Proc. of the Int'l Conf. on Data Engineering (ICDE), pp. 215-224, 2001.

[25] B. Shie, H. Hsiao, V. S. Tseng and P. S. Yu, Mining high utility mobile sequential patterns in mobile commerce environments, DASFAA 2011, pp.224-238.

[26] V. S. Tseng, C.-W. Wu, B.-E. Shie, and P. S. Yu. UP-Growth: an efficient algorithm for high utility itemset mining. In Proc. of Int'l Conf. on ACM SIGKDD, pp. 253–262, 2010.

[27] B. Vo, H. Nguyen, T. B. Ho, and B. Le. Parallel Method for Mining High-utility Itemsets from Vertically Partitioned Distributed Databases. In KES 2009, Part I, LNAI 5711, pp. 251-260, 2009.

[28] C. Wu, B. Shie, V. S. Tseng, P. S. Yu. Mining top-K high utility itemsets. In Proc. of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 78-86, 2012.

[29] C. Wu, P. Philippe, P. S. Yu and V. S. Tseng. Efficient Mining of a Concise and Lossless Representation of High Utility Itemsets. In Proc. of IEEE Int'l Conf. on Data Mining (ICDM), pp.824-833, 2011.

[30] L. Wan, J. Liao, and X. Zhu. A Frequent Pattern Based Framework for Event Detection in Sensor Network Stream Data, Proc. of the Third International Workshop on Knowledge Discovery from Sensor Data (SensorKDD), pp. 87-96, 2009.

[31] X. Ma, H. Pang, K. Tan. Finding Constrained Frequent Episodes Using Minimal Occurrences, In Proc. of the 8th IEEE Int'l Conf. on Data Mining, pp. 471-474, 2004.

[32] J. Yin, Z. Zheng and L. Cao. USpan: An Efficient Algorithm for Mining High Utility Sequential Patterns. In Proc. of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 660-668, 2012.

[33] M. J. Zaki, SPADE: An Efficient Algorithm for Mining Frequent Sequences. Machine Learning, Vol. 42, pp. 31-60, 2001.

[34] Frequent itemset mining implementations repository, http://fimi.cs.helsinki.fi/

[35] FoodMart2000, Microsoft Developer Network (MSDN), http://msdn.microsoft.com/enus/library/aa217032(v=sql.80).asp

[36] NU-MineBench version 2.0 dataset and technical report, http://cucis.ece.northwestern.edu/projects/DMS/MineBench.html