HUE-Span: Fast High Utility Episode Mining

Philippe Fournier-Viger[1], Peng Yang[2],
Jerry Chun-Wei Lin[3], Unil Yun[4]

[1] School of Humanities and Social Sciences,
Harbin Institute of Technology (Shenzhen), China
[2] School of Computer Science and Technology,
Harbin Institute of Technology (Shenzhen), China
[3] Department of Computing, Mathematics and Physics,
Western Norway University of Applied Sciences (HVL), Bergen, Norway
[4] Department of Computer Engineering, Sejong University, Seoul, Republic of Korea
philfv8@yahoo.com, pengyeung@163.com,
jerrylin@ieee.org, yunei@sejong.ac.kr

Abstract. High utility episode mining consists of finding episodes (subsequences of events) that have a high importance (e.g high profit) in a sequence of events with quantities and weights. Though it has important real-life applications, the current problem definition has two critical limitations. First, it underestimates the utility of episodes by not taking into account all timestamps of minimal occurrences for utility calculations, which can result in missing high utility episodes. Second, the state-of-the-art UP-Span algorithm is inefficient on large databases because it uses a loose upper bound on the utility to reduce the search space. This paper addresses the first issue by redefining the problem to guarantee that all high utility episodes are found. Moreover, an efficient algorithm named HUE-Span is proposed to efficiently find all patterns. It relies on a novel upper-bound to reduce the search space and a novel co-occurrence based pruning strategy. Experimental results show that HUE-Span not only finds all patterns but is also up to five times faster than UP-Span.

1 Introduction

Frequent Episode Mining (FEM) [1, 4–6, 12, 15] is a fundamental data mining task, used to analyze a sequence of discrete events. It consists of identifying all episodes having a support (occurrence frequency) that is no less than a user-defined minimum support threshold. An *episode* (also known as serial episode) is a totally ordered set of events (a subsequence). Though FEM has been well-studied, it assumes that each event cannot appear more than once at each time point and that all events have the same importance (e.g. in terms of weight, unit profit or value). But for many real-life applications, this assumption does not hold [17]. For example, for market basket analysis, a sequence of customer transactions may contains non binary item purchase quantities at each time point, and purchased items may have different importance in terms of unit profits. On such data, FEM algorithms may discover a large amount of frequent episodes

that yield a low profit and discard profitable episodes that are less frequent. Hence, to find interesting episodes in sequences, other aspects can be considered such as a measure of importance (e.g. profit).

To address this issue, FEM was generalized as *High Utility Episode Mining* (HUEM) [17] to consider the case where events can appear more than once at each time point and each event has a weight indicating its relative importance. The goal of HUEM is to find *High-Utility Episodes* (HUEs), that is sub-sequences of events that have a high importance as measured by a utility measure (e.g. profit). HUEM has many applications such as website click stream analysis [2], cross-marketing in retail stores [11, 16], stock investment [13] and cloud workload prediction [3]. However, HUEM is more difficult than FEM because the *downward-closure property* used in FEM to reduce the search space using the support measure does not hold for the utility measure. That property states that the support of an episode is anti-monotonic, that is a super-episode of an infrequent episode is infrequent and sub-episodes of a frequent episode are frequent. But the utility of an episode is neither monotonic or anti-monotonic, that is a high utility episode may have a super-episode or sub-episode with lower, equal or higher utility [17]. Hence, techniques designed for reducing the search space in FEM cannot be directly used for HUEM. To mine HUEs in a complex sequence (where some events may be simultaneous) without considering all possible episodes, Wu et al. [17] proposed the UP-Span algorithm. It relies on an upper-bound on the utility that is anti-monotonic, named EWU (*Episode Weighted Utilization*), which is inspired by the TWU measure used in high utility itemset mining [8]. Another HUEM algorithm named T-Span [11] was then proposed but appears to be incomplete and is reported to provide a marginal performance improvement over UP-Span (up to about 25% faster).

Although HUEM is useful, it has two major limitations. First, as this paper will explain, the traditional way of performing utility calculations for episodes, which can underestimate their utility. As a consequence, current HUEM algorithms may miss several HUEs [11, 17]. The reason for this underestimation is that an episode may be supported by multiple timestamps in a time interval but traditional HUEM algorithms only consider the timestamps that follow a specific processing order, and ignore other timestamps where the episode may yield a higher utility. This simplifies the design of HUEM algorithms but can lead to discarding high utility patterns. Second, the state-of-the-art UP-Span algorithm utilizes the loose EWU upper bound on utility. This upper bound is ineffective for reducing the search space of HUEM in large databases.

This paper addresses these limitations by presenting a new framework for mining HUEs in complex event sequences. Contributions are as follows:

- The problem of HUEM is redefined to fix the utility calculation of episodes to guarantee that all high utility episodes are found.
- An efficient algorithm named HUE-Span (**H**igh **U**tility **E**pisodes mining by **Span**ning prefixes) is proposed for mining the complete set of HUEs in complex event sequences.

– The proposed algorithm integrates the concept of *remaining utility* [8, 14] with the EWU model to derive a tighter upper bound on episode utility, named ERU. HUE-Span applies novel search space reduction strategies based on the ERU to reduce the number of candidate episodes.
– To reduce the cost of episode *spanning* (extending an episode with a single event to generate a larger episode), a novel pruning strategy is proposed, named $EEUCP$ (Estimated Episode Utility Cooccurrence Pruning). It can eliminate low utility episodes before a pattern is extended (spanned).
– The performance of the proposed HUE-Span algorithm is compared with the state-of-the-art UP-Span algorithm on both synthetic and real datasets. Results show that the proposed HUE-Span algorithm discovers all HUEs for the redefined HUEM problem, while UP-Span can miss up to 65% of them. Moreover, HUE-Span generates less candidates and is up to five times faster.

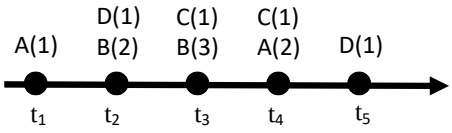
The rest of this paper is organized as follows. Section 2 introduces HUEM and explains why the traditional HUEM model can miss HUEs. Section 3 presents the revised HUEM problem. Then, Sections 4, 5 and 6 respectively presents the proposed HUE-Span algorithm, the experimental evaluation and the conclusion.

2 Frequent and High Utility Episode Mining

The first studies on episode mining focused on finding frequent episodes in a sequence of discrete events with timestamps. FEM is defined as follows [12, 15].

Definition 1 (Complex event sequence). Let $\varepsilon = \{e_1, e_2, \cdots, e_m\}$ be a finite set of events. A complex event sequence $CES = \langle (tSE_{t_1}, t_1), (tSE_{t_2}, t_2), \cdots, (tSE_{t_n}, t_n) \rangle$ is an ordered sequence of simultaneous event sets, where each $tSE_{t_i} \subseteq \varepsilon$ consists of all events associated with a time stamp t_i, and $t_i < t_j$ for any $1 \leq i < j \leq n$.

For example, Fig. 1 (left) shows a complex event sequence $CES = \langle ((A), t_1), ((BD), t_2), ((BC), t_3), ((AC), t_4), ((D), t_5) \rangle$, that will be used as running example. Such sequence can represents various types of data such as customer transactions [1, 4–6], alarm sequences [15], stock data [13] and cloud data [3].



Event	A	B	C	D
Profit	2	1	3	2

Fig. 1: A complex event sequence (left) and external utility values (right)

The goal of FEM is to find all frequent episodes [12, 15]. The support (occurrence frequency) of an episode is measured by counting its minimal occurrences in the input sequence. These concepts are defined as follows.

Definition 2 (Episode containing simultaneous event sets). An episode α is a non-empty totally ordered set of simultaneous events of the form $\langle SE_1, SE_2, \cdots, SE_k \rangle$, where $SE_i \subseteq \varepsilon$ and SE_i appears before SE_j for $1 \leq i < j \leq k$.

Definition 3 (Occurrence). Given an episode $\alpha = \langle SE_1, SE_2, \cdots, SE_k \rangle$, a time interval $[t_s, t_e]$ is called an occurrence of α if (1) α occurs in $[t_s, t_e]$, (2) SE_1 occurs at t_s, and (3) SE_k occurs at t_e. The set of all occurrences of α is denoted as $occSet(\alpha)$.

For instance, the set of all occurrences of $\langle (B), (C) \rangle$ is $occSet(\langle (B), (C) \rangle) = \{[t_2, t_3], [t_2, t_4], [t_3, t_4]\}$.

Definition 4 (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 a *sub-time interval* of $[t_s, t_e]$ if $t_s \leq t'_s \leq t'_e \leq t_e$. The time interval $[t_s, t_e]$ is called a *minimal occurrence* of α if (1) $[t_s, t_e] \in occSet(\alpha)$, (2) there is no alternative occurrence $[t'_s, t'_e]$ of α such that $[t'_s, t'_e]$ is a sub-time interval of $[t_s, t_e]$. The complete set of minimal occurrences of α is denoted as $moSet(\alpha)$.

For example, the episode $\langle (B), (C) \rangle)$ has two minimal occurrences that are $moSet(\langle (B), (C) \rangle) = \{[t_2, t_3], [t_3, t_4]\}$.

Definition 5 (Support of an episode). The support of an episode is its number of minimal occurrences.

For example, the support of episode $\langle (B), (C) \rangle$ is $|moSet(\langle (B), (C) \rangle)| = 2$. FEM consists of finding all episodes that have a minimum support. To consider that events can appear more than once at each time point and that events may have different importance (utility), FEM was generalized as HUEM [17] by introducing the concepts of internal and external utility.

Definition 6 (Internal and external utility of events). Each event $e \in \varepsilon$ of a CES is associated with a postive number $p(e)$, called its *external utility*, representing its relative importance (e.g. unit profit). Moreover, each event e in a simultaneous event set tSE_i at a time point t_i is associated with a positive number $q(e, t_i)$, called its *internal utility* indicating its number of occurrences at time t_i (e.g. purchase quantity).

For example, Fig. 1 (left) shows a complex event sequence with internal utility values. At time point t_2, events B and D have an internal utility (quantities) of 2 and 1, respectively. Fig. 1 (right) indicates that the external utility of A, B, C and D are 2, 1, 3 and 2, respectively. The goal of HUEM is to find high utility episodes, where the utility of an episode is calculated as follows [17].

Definition 7 (Utility of an event at a time point). The *utility* of an event e at a time point t_i is defined as $u(e, t_i) = p(e) \times q(e, t_i)$.

For example,$u(B, t_2) = p(B) \times q(B, t_2) = 1 \times 2 = 2$.

Definition 8 (Utility of a simultaneous event set at a time point). The utility of a simultaneous event set $SE = \{e_{f(1)}, e_{f(2)}, \cdots, e_{f(l)}\}$ at a time point t_i is defined as $u(SE, t_i) = \sum_{j=1}^{l} u(e_{f(j)}, t_i)$.

For example, $u((BD), t_2) = u(B, t_2) + u(D, t_2) = 2 + 2 = 4$.

Definition 9 (Utility of a minimal occurrence of an episode). The utility of a minimal occurrence $[t_s, t_e]$ of an episode $\alpha = \langle SE_1, SE_2, \cdots, SE_k \rangle$ is defined as $u(\alpha, [t_s, t_e]) = \sum_{i=1}^{k} u(SE_i, t_{g(i)})$.

For example, the utility of the minimal occurrence $[t_2, t_3]$ of $\alpha = \langle (B), (C) \rangle$ is $u(\alpha, [t_2, t_3]) = u(B, t_2) + u(C, t_3) = 2+3 = 5$. And $u(\alpha, [t_3, t_4]) = 6$.

Definition 10 (Utility of an episode in a complex event sequence). The *utility of an episode α in a complex event sequence* is the sum of the utility of its minimal occurrences, that is $u(\alpha) = \sum_{mo \in moSet(\alpha)} u(\alpha, mo)$.

For example, the utility of $\langle (B), (C) \rangle$ in the running example is $u(\langle (B), (C) \rangle) = u(\langle (B), (C) \rangle, [t_2, t_3]) + u(\langle (B), (C) \rangle, [t_3, t_4]) = 5 + 6 = 11$.

Definition 11 (High utility episode mining). An episode is a *high utility episode*, if and only if its utility is no less than a user-specified *minimum utility threshold* ($minUtil$). The task of HUEM is to find all HUEs [17].

A major problem with the traditional definition of HUEM is that the utility of an episode is calculated without considering all the timestamps of its minimal occurrences. Hence, utility can be underestimated and it can be argued that some HUEs are missed. This problem is illustrated with an example. Let $minUtil = 9$. The utility of episode $\langle (A), (B), (A) \rangle$ in the CES is calculated as $u(\langle (A), (B), (A) \rangle) = u(\langle (A), (B), (A) \rangle, [t_1, t_4]) = u(A, t_1) + u(B, t_2) + u(A, t_4)$ $= 2 + 2 + 4 = 8$. Thus, this pattern is considered as a low utility episode by HUEM algorithms, and is discarded. But it can be observed that this pattern is a HUE for other timestamps in the same minimal occurrence $[t_1, t_4]$. In fact, $\langle (A), (B), (A) \rangle$ also appears at timestamps t_1, t_3 and t_4 with a utility of $9 \geq minUtil$. The reason why current HUEM algorithms calculate the utility of that episode as 8 in $[t_1, t_4]$ rather than 9 is that the pattern $\langle (A), (B), (A) \rangle$ is obtained by extending $\langle (A), (B) \rangle$ with (A), and that the timestamps of $\langle (A), (B), (A) \rangle$ are obtained by combining those of the minimum occurrence of $\langle (A), (B) \rangle$, i.e. $[t_1, t_2]$ with the timestamp t_4 of (A). In other words, timestamps used for calculating the utility of a minimal occurrence of an episode are determined by the processing order, and other timestamps are ignored. This can lead to underestimating the utility of episodes in time intervals of their minimal occurrences, and hence to discard episodes that should be considered as HUEs.

3 Redefining High Utility Episode Mining

To fix the above issue of utility calculation in HUEM, and ensures that all HUEs are found, this paper proposes to consider all timestamps that match with an

episode in each of its minimal occurrence. This is done by using the maximum of the utility values. Definition 9 is redefined as follows.

Definition 12 (Redefined utility of a minimal occurrence of an episode). Let $[t_s, t_e]$ be a minimal occurrence of an episode $\alpha = \langle SE_1, SE_2, \cdots, SE_k \rangle$, in which each middle simultaneous event set $SE_i \in \alpha$ and $i \in [2, k-1]$ is associated with some (at least one) time points $t_{g(i)1}, t_{g(i)2}, \cdots, t_{g(i)j}$. The *utility* of the episode α w.r.t. $[t_s, t_e]$ is defined as $u(\alpha, [t_s, t_e]) = u(SE_1, t_s) + \sum_{i=2}^{k-1} max\{u(SE_i, t_{g(i)x}) | x \in [1, j]\} + u(SE_k, t_e)$.

For example, the redefined utility of the minimal occurrence $[t_1, t_4]$ of $\langle (A), (B), (A) \rangle$ is $u(\langle (A), (B), (A) \rangle, [t_1, t_4]) = u(A, t_1) + max\{u(B, t_2), u(B, t_3)\} + u(A, t_4) = 2 + max(2, 3) + 4 = 9$. Thus, that episode is a HUE for $minUtil = 9$.

Moreover, to avoid finding some very long minimal occurrences which may not be meaningful, the concept of *maximum time duration* is used in the proposed problem, as in previous work [12].

Definition 13 (Maximum time duration). Let $maxDur$ be a user-specified *maximum time duration*. A minimal occurrence $[t_s, t_e]$ of an episode α is said to satisfy the *maximum time duration* if and only if $t_e - t_e + 1 \leq maxDur$.

The redefined HUEM problem is defined as follows.

Definition 14 (Redefined High Utility Episode Mining). Given $minUtil$ and $maxDur$ thresholds, the redefined problem of HUEM is to find all HUEs when considering only minimal occurrences satisfying the $maxDur$ constraint, and calculating the utility using Definition 12.

4 The HUE-Span algorithm

This subsection introduces the proposed *HUE-Span* algorithm to efficiently discover HUEs in a complex event sequence. HUE-Span adopts the prefix-growth paradigm [17], starting from patterns containing single events and then recursively concatening events to obtain larger patterns.

Definition 15 (Simultaneous and serial concatenations). Let $\alpha = \langle SE_1, SE_2, \cdots, SE_x \rangle$ and $\beta = \langle SE'_1, SE'_2, \cdots, SE'_y \rangle$ be episodes. The *simultaneous concatenation*) of α and β is defined as $simul\text{-}concat(\alpha, \beta) = \langle SE_1, SE_2, \cdots, SE_x \cup SE'_1, SE'_2, \cdots, SE'_y \rangle$. The *serial concatenation* of episodes α and β is defined as $serial\text{-}concat(\alpha, \beta) = \langle SE_1, SE_2, \cdots, SE_x, SE'_1, SE'_2, \cdots, SE'_y \rangle$.

To reduce the search space, the EWU upper bound on the utility was proposed [17]. It is presented, and then a tighter upper bound is proposed.

Definition 16 (Episode-Weighted Utilization of a minimal occurrence of an episode). Let there be a minimal occurrence $[t_s, t_e]$ of an episode $\alpha = \langle SE_1, SE_2, \cdots, SE_k \rangle$ satisfying $maxDur$, where simultaneous event sets are

associated with some time points $t_{g(1)}, t_{g(2)} \cdots t_{g(k)}$, respectively. The *episode-weighted utilization* (EWU) of the minimal occurrence $[t_s, t_e]$ of α is $EWU(\alpha, [t_s, t_e]) = \sum_{i=1}^{k-1} u(SE_i, t_{g(i)}) + \sum_{j=t_e}^{t_s+maxDur-1} u(tSE_j, j)$, where tSE_j is the simultaneous event set at time point j in CES.

Definition 17 (Episode-Weighted Utilization of an episode). The *episode-weighted utilization* of α is the sum of the EWU of its minimal occurrences, that is $EWU(\alpha) = \sum_{mo \in moSet(\alpha)} EWU(\alpha, mo)$.

For example, if $maxDur = 3$, $EWU(\langle (A), (D) \rangle) = \{u(A, t_1) + [u((BD), t_2) + u((BC), t_3)]\} + \{u(A, t_4) + u(D, t_5)\} = \{2 + [4 + 6]\} + \{4 + 2\} = 18$. The EWU is an anti-monotonic upper bound on an episode's utility and can be used to reduce the search space [17]. To be more effective, a tighter upper bound is proposed in this paper, inspired by the concept of *remaining utility* [8, 9, 14].

Definition 18 (Episode-Remaining Utilization of a minimal occurrence of an episode). Let $\succ$ be a total order on events from ε. The *episode-remaining utilization* of a minimal occurrence $[t_s, t_e]$ of an episode α is $ERU(\alpha, [t_s, t_e]) = \sum_{i=1}^{\mathbf{k}} u(SE_i, t_{g(i)}) + u(rSE_{t_e}, t_e) + \sum_{j=t_e+1}^{t_s+maxDur-1} u(tSE_j, j)$, where $u(rSE_{t_e}, t_e) = \sum_{x \in tSE_{t_e} \wedge x \succ SE_k} u(x, t_e)$.

Definition 19 (Episode-Remaining Utilization of an episode). For an episode α, its *episode-remaining utilization* is the sum of the ERU of its minimal occurrences, that is $ERU(\alpha) = \sum_{mo \in moSet(\alpha)} ERU(\alpha, mo)$.

For example, if $maxDur = 3$, $ERU(\langle (A), (D) \rangle) = \{[u(A, t_1) + u(D, t_2)] + 0 + u((BC), t_3)\} + \{u(A, t_4) + u(D, t_5)\} = \{[2 + 2] + 0 + 6\} + \{4 + 2\} = 16$.

Lemma 1 (Anti-monotonicity of the ERU). Let α and β be episodes, and $\gamma = \textit{simult-concat}(\alpha, \beta)$ or $\gamma = \textit{serial-concat}(\alpha, \beta)$. It follows that $u(\gamma) \leq ERU(\gamma) \leq ERU(\alpha) \leq EWU(\alpha)$.

Proof. *Let* $moSet(\alpha) = [mo_1, mo_2, \cdots, mo_x]$, $moSet(\gamma) = [mo'_1, mo'_2, \cdots, mo'_y]$. *Because* $\gamma = \textit{simult-concat}(\alpha, \beta)$ *or* $\gamma = \textit{serial-concat}(\alpha, \beta)$, $|moSet(\alpha)| \geq |moSet(\gamma)|$. *Based on Def. 16 and 18,* $EWU(\alpha) \geq ERU(\alpha) = \sum_{i=1}^{x} ERU(\alpha, mo_i) \geq \sum_{i=1}^{y} ERU(\gamma, mo_i) \geq \sum_{i=1}^{y} ERU(\gamma, mo'_i) = ERU(\gamma) \geq u(\gamma)$.

Theorem 1 (Search space pruning using ERU). For an episode α, if $ERU(\alpha) < minUtil$, then α is not a HUE as well as all its *super-episodes* (obtained by concatenations). Proof. This follows from Lemma 1.

The proposed algorithm uses the EWU and ERU to eliminate candidate episodes during the search for HUEs. However, the ERU and EWU cannot be used to remove events from the complex sequence, and it is costly to calculate these upper bounds. To remove events, we introduce a measure called AWU of events. If the AWU of an event is less than $minUtil$, it can be removed. The rationale for pruning using the AWU will be explained after.

Definition 20 (Action-Window Utilization of an episode). Let $moSet(\alpha) = [t_{s1}, t_{e1}], [t_{s2}, t_{e2}], \cdots, [t_{sk}, t_{ek}]$ be the set of all minimal occurrence of the episode α. Each minimal occurrence $[t_{si}, t_{ei}]$, must be a *sub-time interval* of $[t_{ei} - maxDur + 1, t_{si} + maxDur - 1]$. Hence, the *action-window utilization* of α is defined as $AWU(\alpha) = \sum_{i=1}^{k} \sum_{j=t_{ei}-maxDur+1}^{t_{si}+maxDur-1} tSE_j$.

For example, let $maxDur = 3$. The $AWU(\langle\langle(A)\rangle\rangle) = [0 + 0 + 2 + 4 + 6] + [4 + 6 + 7 + 2 + 0] = 31$. The $AWU(\langle\langle(AC)\rangle\rangle) = [4 + 6 + 7 + 2 + 0] = 19$. The $AWU(\langle\langle(A),(B)\rangle\rangle) = [0 + 2 + 4 + 6] = 12$.

Algorithm 1: The HUE-Span algorithm

input : CES: a complex event sequence,
$minUtil$ and $maxDur$: the user-specified thresholds.
output: The complete set of high utility episodes

1 Scan CES once to calculate the AWU of each event, and remove events such that their $AWU < minUtil$ from the CES, let $\varepsilon*$ be the events that their $AWU \geq minUtil$. Let $\succ$ be the total order of AWU ascending values on $\varepsilon*$;
2 Scan the updated CES to build $EEUCS_{Simul}$ and $EEUCS_{Serial}$;
3 **foreach** *event* $\alpha \in \varepsilon*$ *such that* $ERU(\alpha) \geq minUtil$ **do**
4 | $MiningHUE(\alpha, moSet(\alpha), minUtil, maxDur)$;
5 **end**
6 **Procedure** $MiningHUE(\alpha, moSet(\alpha), minUtil, maxDur)$**:**
7 | **if** $u(\alpha) \geq minUtil$ **then** Output α;
8 | $MiningSimultHUE(\alpha, moSet(\alpha), minUtil, maxDur)$;
9 | $MiningSerialHUE(\alpha, moSet(\alpha), minUtil, maxDur)$;
10 **EndProcedure**

The HUE-Span algorithm. The pseudocode is shown in Algorithm 1. It takes as input a complex event sequence CES with utility values, and the $minUtil$ and $maxDur$ thresholds. The algorithm first scans the CES to calculate the AWU of each event. The algorithm removes events having AWU values that are less than $minUtil$ from CES, and identifies the set $\varepsilon*$ of all events having AWU values that are no less than $minUtil$. The AWU values of events are then used to establish a total order $\succ$ on events, which is the order of ascending AWU values. A second sequence scan is then performed to build a new structure, named $EEUCS$ (Estimated Episode Utility Co-occurrence Structure) used by a novel strategy named $EEUCP$ (Estimated Episode Utility Co-occurrence Pruning) to reduce the number of EWU and ERU calculations. The $EEUCS$ structure is defined as a set of triples of the form (x, y, c), where a triple (x, y, c) indicates that $AWU(\langle\langle(xy)\rangle\rangle) = c$ (where $y \succ x$), or that $AWU(\langle\langle(x),(y)\rangle\rangle) = c$. A $EEUCS$ of the first type is called $EEUCS_{simult}$, while a $EEUCS$ of the second type is called $EEUCS_{serial}$. The $EEUCS_{simult}$ and $EEUCS_{serial}$ structures can be implemented as two matrices as shown in Fig. 2. But in HUE-Span, they are implemented as hashmaps of hashmaps where only tuples of the form (x, y, c)

such that $c \neq 0$ are stored. This representation is more memory efficient because on real data, few events typically co-occur with other events in $EEUCS_{simult}$.

Event	A	D	B	C
A		0	0	19
D			19	0
B				21
C				

(a) $EEUCS_{simult}$

Event	A	D	B	C
A	0	27	12	12
D	17	0	19	19
B	36	15	19	38
C	19	30	0	19

(b) $EEUCS_{serial}$

Fig. 2: $EEUCS_{simult}$ and $EEUCS_{serial}$ for the sequence of Fig. 1 and $maxDur = 3$

Then, the algorithm considers extending each episode $\alpha \in \varepsilon*$ such that $ERU(\alpha) \geq minUtil$ by calling the *MiningHUE* procedure. Other episodes are not extended, based on Theorem 1. The *MiningHUE* procedure spans a prefix α (Line 6-10) by calling two procedures *MiningSimultHUE* and *MiningSerialHUE*. The former first considers concatenating simultaneous events to α according to $\succ$, while the latter first considers concatenating serial events to α.

Algorithm 2: *MiningSimultHUE*

input : α: an episode, $occSet(\alpha)$: possible occurrences of α, $minUtil$, $maxDur$: the user-specified thresholds.
output: The set of high utility simultaneous episodes w.r.t prefix α

1 Let $lastEvent$ be the last event of α, and initialize β-$Set \leftarrow \emptyset$;
2 **foreach** occurrence of α $[t_s, t_e] \in occSet(\alpha)$ **do**
3 **foreach** event e that occurs at t_e and $e \succ lastEvent$ **do**
4 $\beta = simult\text{-}concat(\alpha, e)$;
5 $occSet(\beta) \leftarrow occSet(\beta) \cup [t_s, t_e]$;
6 **if** $[t_s, t_e]$ is a minimal occurrence of β **then** $moSet(\beta) \leftarrow moSet(\beta) \cup [t_s, t_e]$;
7 Add β to β-Set;
8 **end**
9 **end**
10 **foreach** episode $\beta \in \beta$-Set **do**
11 **if** $\forall (x, y) \in \beta, EEUCS_{simult}(x, y) \geq minUtil \wedge EEUCS_{serial}(x, y) \geq minUtil$ **and** $ERU(\beta) \geq minUtil$ **then** $MiningHUE(\beta, occSet(\beta), minUtil, maxDur)$;
12 **end**

The *MiningSimultHUE* procedure (Algorithm 2) is applied as follows. For each occurrence $[t_s, t_e] \in occSet(\alpha)$, the algorithm considers each event e that

occurs at t_e and such that e is greater than all events in the last event of α according to $\succ$. For each such event, the algorithm performs a simultaneous concatenation of α with e to obtain an episode β (Line 4). Then, $[t_s, t_e]$ is added to $occSet(\beta)$ (Line 5). If $[t_s, t_e]$ is a minimal occurrence, $[t_s, t_e]$ is added to $moSet(\beta)$ (Line 6). Then, β is added to a set β-Set. After $occSet(\alpha)$ has been traversed, the algorithm considers each episode $\beta \in \beta$-Set. If the episode does not pass the $EEUCP$ pruning conditions, it is ignored (Line 11). Let y be the last added event to β. The pruning conditions are that there is no tuple (x, y, c) such that $c \geq minUtil$ (1) in $EEUCS_{simult}$ for an event x appearing at the same time point as y, and no such tuple (2) in $EEUCS_{serial}$ for an event x followed by y. If the conditions are verified, then β and all its super-episodes are not HUEs and are ignored. Otherwise, $ERU(\beta)$ is calculated by using $moSet(\beta)$, and if $ERU(\beta) \geq minUtil$, the procedure *MiningHUE* is called to further concatenate events to β to find larger HUEs (Line 11).

Algorithm 3: *MiningSerialHUE*

```
   input  : α: an episode, occSet(α): possible occurrences of α,
            minUtil, maxDur: the user-specified thresholds.
   output: The set of high utility serial episodes w.r.t prefix α
 1 Let β-Set ← ∅;
 2 foreach minimal occurrence [ts1, te1] ∈ occSet(α) do
 3 |   Let [ts2, te2] be the next minimal occurrence;
 4 |   foreach event e occurring at a time point t in
   |     [te1 + 1, min(ts1 + maxDur − 1, te2)] do
 5 |     β = serial-concat(α, e);
 6 |     occSet(β) ← occSet(β) ∪ [ts1, t];
 7 |     if [ts1, t] is a minimal occurrence of β then
   |       moSet(β) ← moSet(β) ∪ [ts1, t];
 8 |     Add β to β-Set;
 9 |   end
10 end
11 foreach episode β ∈ β-Set do
12 |   if ∀(x, y), EEUCS_serial(x, y) ≥ minUtil and ERU(β) ≥ minUtil then
   |     MiningHUE(β, occSet(β), minUtil, maxDur);
13 end
```

The *MiningSerialHUE* procedure (Algorithm 3) is applied as follows. For each minimal occurrence $[t_{s1}, t_{e1}]$ of $occSet(\alpha)$, the algorithm finds the next minimal occurrence $[t_{s2}, t_{e2}]$ (Line 2-3). Then, the algorithm processes each event e occurring at a time point t in interval $[t_{e1} + 1, \min(t_{s1} + maxDur - 1, t_{e2})]$ (Line 4). For each such event e, a serial concatenation of α with e is done to obtain an episode β (Line 7). Here, it is important to note that existing HUEM algorithms [11, 17] ignore the fact that t cannot exceed the end time point of the next minimal occurrence (or the largest time point of CES). If t exceeds

the end time point of the next minimal occurrence ($t > t_{e2}$), then $[t_{s1}, t]$ cannot be a minimal occurrence because $[t_{s2}, t]$ is a *sub-time interval* of $[t_{s1}, t]$. Hence, this technique reduces the search space. Then, the next operations are the same as *MiningSimultHUE* with the only difference being that only $EEUCS_{serial}$ is used in *MiningSerialHUE* (Line 12). This is because no events are simultaneous events with the last added event y in β.

The $EEUCP$ strategy is correct (only prunes low-utility episodes) because the AWU considers a larger window than ERU, and ERU is an upper bound on the utility (Theorem 1). Because HUE-Span starts from single events and considers larger episodes by recursively performing simultaneous and serial concatenations, and only prunes the search space by Theorem 1 and using $EEUCP$, HUE-Span is correct and complete to discover all HUEs.

5 Experimental Evaluation

We performed experiments to assess the performance of the proposed algorithm. Experiments were performed on a computer having a 64 bit Xeon E3-1270 3.6 Ghz CPU, running Windows 10 and having 64 GB of RAM. We compared the performance of HUE-Span with the state-of-the-art UP-Span algorithm for high-utility episode mining. Both real and synthetic datasets were used to compare the algorithms' performance. The real *Retail* and *Kosarak* datasets are commonly used in the pattern mining literature and were obtained from the SPMF library website (`https://www.philippe-fournier-viger.com/spmf/`), while synthetic datasets were generated using the IBM data generator. The IBM generator has four parameters: T is the average size of a simultaneous event set at a time point; I is the average size of event sets in frequent episodes; N is the number of distinct events; D is the total number of time points. Internal utility and external utility values were then generated using the SPMF generator. It has two parameters: Q is the maximum internal utility (quantity) of each event at a time point; F is the average external utility of each event. The obtained *Retail* and *Kosarak* datasets have unit profits (external utility) and purchased quantities (internal utility). Note that these three datasets are sometimes considered as transaction databases but they also can be considered as a single complex event sequence by regarding each item as an event and each transaction as a simultaneous event set. Characteristics of the datasets are presented in Table 1.

Table 1: Characteristics of the datasets

Dataset	#Time Point	#Event	Avg. Length
T25I10N1KD10KQ10F5	9,976	929	24.8
Retail	88,162	16,470	10.3
Kosarak	990,002	41,270	8.1

In the experiments, UP-Span is compared with three versions of the proposed algorithm: HUE-Span(ERU) only uses ERU for search space pruning; HUE-Span(EEUCP) only uses $EEUCP$ for search space pruning; HUE-Span(ERU + EEUCP) uses both ERU and $EEUCP$ for pruning. In the following, an algorithm name followed by a star * means that it applies the proposed redefined utility (Definition 12) to find all HUEs by calculating the maximum utility for each minimal occurrence of an episode. In the following experiments, $minUtil$ is expressed as *a percentage* of the total utility of the complex event sequence.

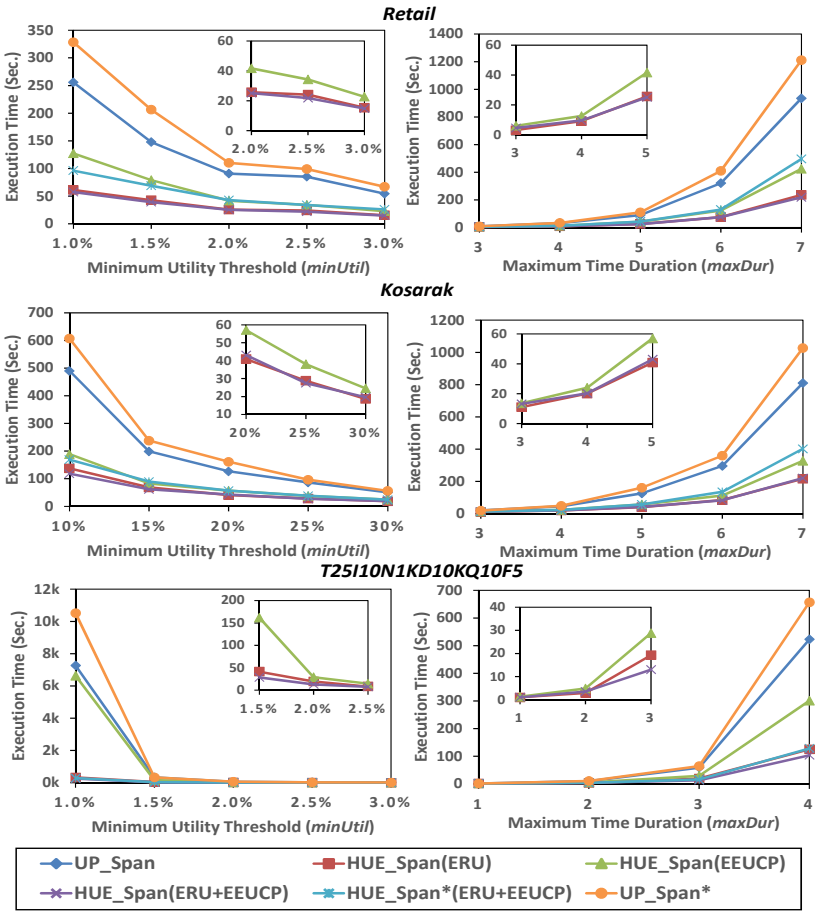


Fig. 3: Comparison of execution times for various $minUtil$ and $minDur$ values

Fig. 3 compares the runtimes of HUE-Span and UP-Span for different $minUtil$ and $maxDur$ values on these three datasets respectively. In the leftmost figures from top to bottom, $maxDur$ was set to 5, 5 and 3 respectively. In the rightmost

figures from top to bottom, $minUtil$ was set to 2%, 20% and 2% respectively. It is observed that HUE-Span(ERU+EEUCP) is up to 5 times faster than UP-Span on all datasets. Moreover, for the dense dataset (T25I10N1KD10KQ10F5), HUE-Span(ERU+EEUCP) is up to 20 times faster than UP-Span. This is because pruning using ERU is more effective for dense datasets. Six sub-figures show enlarged parts of the charts to better show the runtime differences between ERU pruning and $EEUCP$. It can be observed that ERU pruning is better than $EEUCP$. The reason is that $EEUCP$ uses the AWU of an episode, and this latter considers a larger window than the ERU.

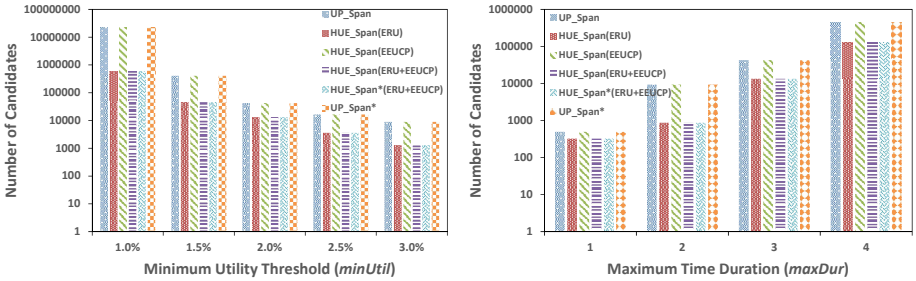


Fig. 4: Number of candidates on T25I10N1KD10KQ10F5 for various $minUtil$ and $minDur$ values

Fig. 4 shows the number of candidates generated by different algorithms on T25I10N1KD10KQ10F5 using a log scale. In the left figure, $maxDur = 3$ and in the right figure, $minUtil = 2\%$. It is observed in Fig. 4 that the number of candidates grows rapidly when $minUtil$ decreases or $maxDur$ increases. It is also seen that using ERU pruning reduces the number of candidates by a large amount compared to if only EWU pruning or $EEUCP$ are used. For example, in Fig. 4 (left), HUE-Span(ERU+EEUCP) generates 10 times less candidates than UP-Span when $minUtil$ is set to 1%.

Memory consumption of the algorithms was also evaluated. Fig. 5 (left) compares the memory consumption of the algorithms on the Retail dataset for different $minUtil$ values. It is observed that HUE-Span with pruning strategies uses less memory than UP-Span since the proposed pruning strategies reduce the number of candidates. Fig. 5 (right) compares memory consumption of the algorithms on the Kosarak dataset for different $minUtil$ values. It is found that HUE-Span(ERU) sometimes uses less memory than HUE-Span(ERU+EEUCP) because the $EEUCS_{simult}$ and $EEUCS_{serial}$ require memory. Overall, results show that the HUE-Span(*) algorithm is better than the UP-Span(*) algorithm.

The number of patterns found was also compared for different $minUtil$ and $maxDur$ values. Results for Retail and Kosarak are shown in Table 2. The column $\#HUE^*$ indicates the number of HUEs found using the proposed redefined utility (calculated as the maximum utility for each minimal occurrence of an episode). The column $\#HUE$ indicates the number of HUEs found by

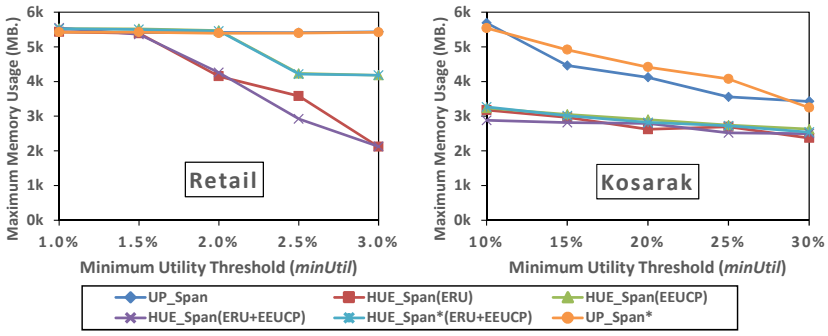


Fig. 5: Memory consumptions of the algorithms

UP-Span. The column $\#HUE^{-}$ indicates the number of patterns for which the utility is understimated by UP-Span. It can be observed that UP-Span finds much less HUEs than the proposed HUE-Span* algorithm, missing up to 65% of the HUEs found by HUE-Span*. Moreover, UP-Span underestimates the utility of up to 79% of the HUEs that it outputs in terms of redefined utility.

Table 2: Number of patterns found by the algorithms

Dataset	$minUtil$	$maxDur$	$\#HUE^{*}$	$\#HUE$	$\#HUE^{-}$
Retail	1%	5	1,556	1,174	745
	1.5%	5	523	422	196
	2%	5	179	170	98
	2%	6	730	439	296
	2%	7	2,084	1,077	858
Kosarak	10%	5	105	73	29
	15%	5	27	22	5
	20%	5	3	2	0
	20%	6	21	8	1
	20%	7	81	28	16

6 Conclusion

This paper demonstrated that the traditional HUEM model can underestimate the utility of episodes and thus miss HUEs. To address this issue, this paper has adapted the HUEM model to consider the highest (maximal) utility for each minimal occurrence. Moreover, to mine HUEs efficiently, an algorithm named HUE-Span was proposed. It relies on a novel ERU upper bound to reduce the search space and a novel pruning strategy based on event co-occurrences. Extensive experiments on both synthetic and real datasets have shown that HUE-Span

not only discovers all HUEs but is up to five times faster than the state-of-the-art UP-Span algorithm. The source code of HUE-Span and datasets can be downloaded from the SPMF website. For future work, we will design other optimizations for high utility episode mining and consider using high utility episodes to derive high utility episode rules [5, 6], peak episodes [10] and significant patterns [7].

References

1. Achar, A., Laxman, S., Sastry, P.S.: A unified view of the apriori-based algorithms for frequent episode discovery. Knowl. Inf. Syst. **31**(2), 223–250 (2012)
2. Ahmed, C.F., Tanbeer, S.K., Jeong, B.S.: A framework for mining high utility web access sequences. IETE Technical Review **28**(1), 3–16 (2011)
3. Amiri, M., Mohammad-Khanli, L., Mirandola, R.: An online learning model based on episode mining for workload prediction in cloud. Future Generation Computer Systems **87**, 83–101 (2018)
4. Ao, X., Luo, P., Li, C., Zhuang, F., He, Q.: Online frequent episode mining. In: Proc. 31st IEEE Int. Conf. on Data Eng. pp. 891–902 (2015)
5. Ao, X., Luo, P., Wang, J., Zhuang, F., He, Q.: Mining precise-positioning episode rules from event sequences. IEEE Trans. Knowl. Data Eng. **30**(3), 530–543 (2018)
6. Fahed, L., Brun, A., Boyer, A.: DEER: distant and essential episode rules for early prediction. Expert Syst. Appl. **93**, 283–298 (2018)
7. Fournier-Viger, P., Li, X., Yao, J., Lin, J.C.W.: Interactive discovery of statistically significant itemsets. In: Proc. 31rd Intern. Conf. on Industrial, Engineering and Other Applications of Applied Intelligent Systems. pp. 101–113 (2018)
8. Fournier-Viger, P., Lin, J.C.W., Truong-Chi, T., Nkambou, R.: A survey of high utility itemset mining. In: High-Utility Pattern Mining, pp. 1–45. Springer (2019)
9. Fournier-Viger, P., Wu, C., Zida, S., Tseng, V.S.: FHM: faster high-utility itemset mining using estimated utility co-occurrence pruning. In: Proc. 21st Int. Syp. on Metho. for Intell. Systems. pp. 83–92 (2014)
10. Fournier-Viger, P., Zhang, Y., wei Lin, J.C., Fujita, H., Koh, Y.S.: Mining local and peak high utility itemsets. Information Sciences **481**, 344–367 (2019)
11. Guo, G., Zhang, L., Liu, Q., Chen, E., Zhu, F., Guan, C.: High utility episode mining made practical and fast. In: Proc. 10th Int. Conf. on Advanced Data Mining and Applications. pp. 71–84 (2014)
12. Huang, K., Chang, C.: Efficient mining of frequent episodes from complex sequences. Inf. Syst. **33**(1), 96–114 (2008)
13. Lin, Y., Huang, C., Tseng, V.S.: A novel methodology for stock investment using high utility episode mining and genetic algorithm. Appl. Soft Comput. **59**, 303–315 (2017)
14. Liu, M., Qu, J.: Mining high utility itemsets without candidate generation. In: Proc. 21st ACM Int. Conf. on Inf. and Knowl. Management. pp. 55–64 (2012)
15. Mannila, H., Toivonen, H., Verkamo, A.I.: Discovery of frequent episodes in event sequences. Data Min. Knowl. Discov. **1**(3), 259–289 (1997)
16. Rathore, S., Dawar, S., Goyal, V., Patel, D.: Top-k high utility episode mining from a complex event sequence. In: Proc. 21st Int. Conf. on Management of Data. pp. 56–63 (2016)
17. Wu, C., Lin, Y., Yu, P.S., Tseng, V.S.: Mining high utility episodes in complex event sequences. In: Proc. 19th ACM SIGKDD Int. Conf. on Knowl. Discovery. pp. 536–544 (2013)