Discovering and Visualizing Efficient Patterns in Cost/Utility Sequences

Philippe Fournier-Viger[1], Jiaxuan Li[2],
Jerry Chun-Wei Lin[3], and Tin Truong-Chi[4]

[1] School of Humanities and Social Sciences, Harbin Institute of Technology (Shenzhen), Shenzhen, China
[2] School of Computer Sciences, Harbin Institute of Technology (Shenzhen), Shenzhen, China
[3] Department of Computing, Mathematics and Physics, Western Norway University of Applied Sciences (HVL), Bergen, Norway
[4] Department of Mathematics and Informatics, University of Dalat, Dalat, Vietnam
philfv@hit.edu.cn, jiaxuanliniki@gmail.com,
jerrylin@ieee.org, tintc@dlu.edu.vn

Abstract. Many algorithms have been proposed to discover interesting patterns in sequences of events or symbols, to support decision-making or understand the data. In sequential pattern mining, patterns are selected based on criteria such as the occurrence frequency, periodicity, or utility (eg. profit). Although this has many applications, it does not consider the effort or resources consumed to apply these patterns. To address this issue, this paper proposes to discover patterns in cost/utility sequences, in which each event/symbol is annotated with a cost, and where a utility value indicates the benefit obtained by performing each sequence. Such sequences are found in many fields such as in e-learning, where learners do various sequences of learning activities having different cost (time), and obtain different utility (grades). To find patterns that provide a good trade-off between cost and benefits, two algorithms are presented named CEPDO and CEPHU. They integrate many optimizations to find patterns efficiently. Moreover, a visualization module is implemented to let users browse patterns by their skyline and visualize their properties. A case study with e-learning data has shown that insightful patterns are found and that the designed algorithms have excellent performance.

Keywords: Pattern mining · Cost/Utility Sequences · Visualization.

1 Introduction

Sequences of symbols or events are an important type of data found in many domains [8]. A popular method for analyzing sequences is sequential pattern mining (SPM). It consists of discovering all subsequences of symbols or events that have an occurrence frequency (support) exceeding some minimum frequency threshold [1, 4, 8]. Albeit discovering sequential patterns is useful to understand data and to support decision making, frequent patterns are not always relevant [15].

To find more interesting patterns, SPM has been recently extended as the problem of High Utility Sequential Pattern Mining (HUSPM) [2, 15]. The aim is to find sequences having a high utility, where the utility is a numerical measure of the profit, importance or benefits provided by a pattern. Though, HUSPM has many applications, it focuses on the utility or benefits provided by patterns but ignores the cost or effort for obtaining these benefits. For instance, a pattern $\langle material1, material20, A+\rangle$ may be found in e-learning data, indicating that reading learning material 1 and then learning material 20 leads to obtaining a high score (A+). But HUSPM ignores the cost of patterns (e.g. the monetary cost of the materials and the effort required to learn using these materials).

Generally, the cost of a pattern could be calculated in terms of different aspects such as the time, money, resources consumed and effort. Because HUSPM does not consider the cost of patterns, it can find numerous patterns that have a high utility but have a high cost. For example, in e-learning, many patterns may lead to a desirable outcome such as a high grade (a high utility) but have a cost that is prohibitive. Moreover, HUSPM may miss numerous patterns that have a lower utility but provides an excellent trade-off between cost and utility. Thus, integrating the concept of cost in HUSPM is desirable but hard since cost and utility may measure different aspects (e.g. profit vs time). Hence, cost cannot be simply subtracted from the utility to apply existing HUSPM algorithms. Moreover, doing so would fail to assess how strong the correlation between cost and utility is for each pattern, while it is desirable to discover patterns that not only have a low cost and high utility but that provide a good trade-off and strong correlation between cost and utility.

This paper addresses this limitation of HUSPM by proposing a novel problem of mining *cost-effective patterns* (CEP) by considering both utility and cost. A CEP is a pattern having a good cost/utility trade-off. Two variations of the problem are defined to handle binary utility values indicating desirable/undesirable outcomes (e.g. *passed* vs $failed$) and numeric values (e.g. exam grades), respectively. Moreover, statistical measures are designed to assess the cost/utility correlation. Two algorithms are proposed to find all CEPs (one for each problem variation). The algorithms rely on optimizations, a lower-bound on the average cost of patterns and an upper-bound on their utility, to find all CEPs efficiently. Moreover, a visualization module is proposed to facilitate pattern analysis. It lets users visualize properties of CEPs, and browse CEPs using their skyline. An experimental study has shown that the algorithms are efficient and that the proposed lower-bound can considerably reduce the search space. In addition, a case study on e-learning data was carried and shows that interesting cost-effective patterns are found using the designed algorithms.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 defines the problem of cost-effective pattern mining. Section 4 presents the proposed algorithms. Section 5 presents the performance evaluation. Section 6 describes a case study with e-learning data, and presents the visualization module. Finally, Section 7 draws a conclusion and discusses future work.

2 Related work

High utility sequential pattern mining (HUSPM) has become an important data mining problem in recent years, and many algorithms have been proposed for this problem [2, 15]. Although HUSPM has many applications, it was first proposed to mine sequences of purchases that yield a high profit from sequences of customer transactions [15]. The input of HUSPM is a minimum utility threshold *minutil* and a database of quantitative sequences, where each sequence is an ordered list of transactions. A transaction is a set of symbols or events, each annotated with some internal utility value (e.g. a purchase quantity). Moreover, each symbol has a weight called external utility, indicating its relative importance (e.g. unit profit). Table 1 shows an example quantitative sequence database containing four sequences. The first sequence indicates that a customer purchased three units of item a, followed by three units of item b, then two units of item b, and finally one unit of item c. The output of HUSPM is the set of all sequences having a utility that is no less than *minutil* (e.g. all sequences that yield a high profit).

Table 1. A quantitative sequence database containing four sequences.

Quantitative sequences with purchase quantities (internal utility)
sequence 1: $\langle(a,3),(b,3),(c,1),(b,4)\rangle$
sequence 2: $\langle(a,1),(e,3)\rangle$
sequence 3: $\langle(a,6),(c,7),(b,8),(d,9)\rangle$
sequence 4: $\langle(b,3),(c,1)\rangle$
Unit profits (external utility)
$a = 5\$$, $b = 1\$$, $c = 2\$$, $d = 1\$$

Although HUSPM is useful, it does not consider the cost of patterns. Until now, few pattern mining studies have considered a concept of cost. The closest work to the one presented in this paper is in the field of process mining, where cost was considered to perform behavior analysis. The main goal of process mining is to extract models from event logs that characterize a business process. An event log is a list of events with timestamps, ordered by time. Various methods have been used to analyze processes [11, 13]. Dalmas et al. integrated the concept of cost in process mining by proposing the TWINCLE algorithm [5]. It was applied to find patterns in event logs of hospitals, where each patient activity has a cost. The algorithm finds sequential rules, where the antecedent and consequent of a rule are events. Rules are selected based on their cost and displayed to the user with the aim of reducing the monetary cost of medical treatments. Although interesting low cost patterns were found using TWINCLE, it ignores the utility of patterns. As a result, TWINCLE can find many patterns representing cheap medical treatments but having a low effectiveness (utility).

To address the limitations of the above studies, the next section proposes to mine patterns in event sequences by considering both a cost and a utility model, and where utility is the sequence outcome and cost is event annotations.

3 Problem definition

The type of data considered in this paper is a **sequential activity database** (SADB), which contains several sequences, where a sequence is a time-ordered list of activities. Sequences of activities can model various types of data, and are similar to event logs in process mining [3]. The main difference is that this paper considers that each sequence of events includes cost and utility information. Formally, a SADB is a set of sequences $SADB = \{S_1, S_2, \ldots S_n\}$, where each sequence S_s has a unique identifier s ($1 \leq s \leq n$). A sequence S_s is a list of activities $\langle\{v_1[c_1], v_2[c_2], \ldots v_m[c_m]\}\ |Utility\rangle$, where the notation $v_i[c_i]$ indicates that the activity v_i had a cost c_i (a positive number representing some effort or resources consumed), and u_s represents the utility of the sequence S_s (the benefits obtained after performing the sequence). To be able to address the needs of different applications, the utility is viewed as either binary or numeric values. A SADB containing numeric or binary utility values is called *binary SADB* or *numeric SADB*, respectively.

For instance, Table 2 shows a binary SADB from the e-learning domain containing five sequences, each indicating the learning materials (denoted as a, b, c, d, e) studied by a learner. Each learning material is annotated with a cost value indicating the amount of time that it was studied (in hours), and each sequence has a binary utility indicating whether the student passed (+) or failed (-) the final exam. The first sequence indicates that a student spent 2 hours to study a, followed by 4 hours to study b, 9 hours for c, 2 hours for d, and then passed the exam. Table 3 shows a numeric SADB where sequences are represented using the same format but utility is numeric values indicating scores obtained at the final exam. That SADB contains five sequences. The first sequence has a positive utility value of 80.

Table 2. SADB with Binary Utility

Sid	Sequence (activity[cost])	Utility
1	<(a[2]),(b[4]),(c[9]),(d[2])>	+
2	<(b[1]),(d[12]),(c[10]),(e[1])>	−
3	<(a[5]),(e[4]),(b[8])>	+
4	<(a[3]),(b[5]),(d[1])>	−
5	<(b[3]),(e[4]),(c[2])>	+

Table 3. SADB with Numeric Utility

Sid	Sequence (activity[cost])	Utility
1	<(a[20]),(b[40]),(c[50])>	80
2	<(b[25]),(d[12]),(c[30])>	60
3	<(a[25]),(e[14]),(b[30])>	50
4	<(a[40]),(b[16]),(d[40])>	40
5	<(b[20]),(e[24]),(c[20])>	70

To find interesting patterns in sequences of activities, this paper consider that a pattern p is an ordered list of activities $\{v_1, v_2, \ldots, v_o\}$. To select interesting patterns, three measures are first considered: support, cost and occupancy. The *support* measure is used to reduce the search space and eliminate rare patterns that may represent noise in the data and may be insignificant. It is used and defined as in SPM [8].

Definition 1. *The **support of a pattern** p is the number of sequences containing p, i.e. $sup(p) = |\{S_s | p \subseteq S_s \wedge S_s \in SADB\}$* [8].

The second measure is the *cost*. It allows evaluating the amount of resources spent to apply a pattern. But because a pattern's cost may not be the same in all sequences where it appears, we propose to calculate the average cost.

Definition 2. *The* ***cost of a pattern*** p ***in a sequence*** S_s *is:* $c(p, S_s) = \sum_{v_i \in first(p,S_s)} c(v_i, S_s)$ *if* $p \subseteq S_s$ *and otherwise* 0*, where* $first(p, S_s)$ *denotes the first occurrence of* p *in* S_s*. The* ***cost of a pattern*** p ***in a SADB*** *is the sum of its cost in all sequences, i.e.* $c(p) = \sum_{p \subseteq S_s \wedge S_s \in SADB} c(p, S_s)$*. The* ***average cost*** *of a pattern* p *is :* $ac(p) = \frac{c(p)}{|sup(p)|}$*, and represents the average effort to apply the pattern* p*.*

The average cost measure is useful as it provides an overview of the resources consumed to apply each pattern. It is to be noted that since a pattern may have multiple occurrences in a sequence, the pattern's cost could be calculated in different ways. A possibility could be to use the sum or the average of the costs of all occurrences. But this is not trivial to calculate and can lead to overestimating the cost because two occurrences may overlap and share events. Thus, it was decided to simply use the first occurrence. The proposed definition can also work for the last occurrence (by processing sequences in backward order).

The third measure is the *occupancy*, which is borrowed from SPM [17]. This measure aims at finding patterns that cover large parts of sequences where they appear. The assumption is that high occupancy patterns represent well these sequences and thus are more likely to have influenced their outcome (utility) than low occupancy patterns.

Definition 3. *The* ***occupancy of a pattern*** p *is calculated as:* $occup(p) = \frac{1}{\text{sup}(p)} \sum_{p \subseteq S_s \wedge S_s \in SADB} \frac{|p|}{|S_s|}$ [17].

For example, consider the database of Table 2 and pattern $p = \{ab\}$. It is found that $c(p, S_1) = 6$, $c(p, S_3) = 13$, $c(p, S_4) = 8$, $sup(p) = 3$, $ac(p) = (6 + 13 + 8)/3 = 9$, and $occup(p) = (2/4 + 2/3 + 2/3)/3 \approx 0.61$.

The above measures are used to ensure that patterns have a low cost, are not infrequent, and are representative of sequences. But those measures cannot evaluate if patterns are cost-efficient (provide a good trade-off between cost and utility) and are correlated to a positive outcome (utility). For this purpose, the following paragraphs propose two measures of cost-efficiency called *cor* and *trade-off*, which are defined to handle binary and numeric SADBs, respectively.

Correlation of a pattern in a binary SADB. In a binary SADB, the utility of a sequence is a binary value indicating a positive or negative outcome (e.g. passed/failed an exam). To find patterns that contribute to a positive outcome, the *cor* measure is defined as follows.

Definition 4. *The* ***correlation of a pattern*** p ***in a binary SADB*** *is:* $cor(p) = \frac{ac(D_p^+) - ac(D_p^-)}{Std} \sqrt{\frac{|D_p^+||D_p^-|}{|D_p^+ \cup D_p^-|}}$ *where* D_p^+ *and* D_p^- *respectively denote the set of positive and negative sequences containing the pattern* p*,* $ac(D_p^+)$ *and* $ac(D_p^-)$ *are the pattern* p*'s average cost in* D_p^+ *and* D_p^-*,* Std *is the standard deviation of* p*'s cost, and* $|D_p^+|$ *and* $|D_p^-|$ *are the support of* p *in* D_p^+ *and* D_p^-*, respectively.*

The *cor* measure is a variation of the biserial correlation measure [12] used in statistics to assess the correlation between a binary and a numeric attribute. The *cor* measure values are in the $[-1, 1]$ interval. The greater positive (smaller negative) the *cor* measure is, the more a pattern is correlated with a positive (negative) utility. In the *cor* measure, the term $ac(D_p^+) - ac(D_p^-)$ is used to find patterns that have a large difference in terms of average cost for positive and negative sequences. That cost difference is divided by the standard deviation of the cost to avoid using absolute values in the equation. The term $\sqrt{\frac{|D_p^+||D_p^-|}{|D_p^+ \cup D_p^-|}}$ is used to find patterns that appears more frequently in sequences having a positive outcome than having a negative outcome.

For example, consider the binary SADB of Table 2 and $p = \{ab\}$. It is found that $c(p, S_1) = 6$, $c(p, S_3) = 13$, $c(p, S_4) = 8$, $sup(p, +) = 2$, $sup(p, -) = 1$, $ac(D_p^+) = (6 + 13)/2 = 9.5$ and $ac(D_p^-) = (8)/1 = 8$. Hence, $cor(ab) = \frac{9.5-8}{1.06}\sqrt{\frac{2\times 1}{|3|}} \approx 0.314 > 0$ is correlated with a positive outcome.

Correlation of a pattern in a numeric SADB. In a numeric SADB, the utility of a sequence is a positive number (e.g. a grade obtained at an exam) where a high (low) value indicate a good (bad) outcome. To measure the correlation of a pattern with a good outcome represented as numeric utility, the *trade-off* measure is proposed. It evaluates the relationship between cost and utility. The utility of a pattern and its *trade-off* are calculated as follows.

Definition 5. *The* ***utility of a pattern*** p ***in a numeric SADB*** *is the average of the utility of sequences in which it appears, that is* $u(p) = \frac{\sum_{p \subseteq S_s \wedge S_s \in SADB} su(S_s)}{|sup(p)|}$, *where* $su(S_s)$ *denotes the utility of a sequence* S_s.

Definition 6. *The* ***trade-off of a pattern*** p *is the ratio of its average cost to its average utility, that is* $tf(p) = ac(p)/u(p)$.

Trade-off values are in the $(0, \infty]$ interval. The trade-off value of a pattern indicates how efficient the pattern is. A pattern having a small trade-off is viewed as being cost-effective as it provides utility at a low cost (it requires a small amount of resources to obtain a desirable outcome). For example, in e-learning, a pattern with a small trade-off may indicate that studying some learning materials (events) typically requires a small amount of time (cost) and is correlated with obtaining high scores (utility). On the other hand, patterns having a larger trade-off may be viewed as being less efficient.

For instance, consider Table 3 and pattern $p = \{ab\}$. It is found that $c(p, S_1) = 60$, $c(p, S_3) = 55$, $c(p, S_4) = 56$, $sup(p) = 3$, $ac(p) = (60 + 55 + 56)/3 = 57$, $u(p) = \frac{su(p,S_1)+su(S_3)+su(p,S_4)}{sup(p)} = \frac{80+50+40}{3} \approx 56.67$. The trade-off of pattern p is $tf(p) = ac(p)/u(p) = 57/56.67 \approx 1.01$.

Based on the measures presented in this section, two problems are defined to find cost-effective patterns in binary and numeric SADBs, respectively.

Problem 1. Mining Cost-Effective Patterns in a binary SADB. Given user-specified *minsup*, *minoccup* and *maxcost* thresholds, a pattern p

is a cost-effective pattern in a binary SADB if $(\text{sup}(p) \geq minsup) \wedge (occup(p) \geq minoccup) \wedge (ac(p) \leq maxcost)$, and $cor(p)$ has a high positive value.

Problem 2. Mining Cost-Effective Patterns in a numeric SADB. Given user-specified *minsup*, *minoccup*, *minutil* and *maxcost* thresholds a pattern p is a cost-effective pattern in a numeric SADB if $(\text{sup}(p) \geq minsup) \wedge (occup(p) \geq minoccup) \wedge (ac(p) \leq maxcost) \wedge (u(p) \geq minutil)$ and $tf(p)$ has a small value.

4 The CEPDO and CEPHU Algorithms

This section presents two algorithms, designed for mining cost-effective patterns leading to a desirable outcome (CEPDO), and cost-effective patterns leading to a high utility (CEPHU), respectively (Problem 1 and Problem 2).

Both algorithms search for patterns using a pattern-growth approach inspired by the PrefixSpan algorithm for SPM [8]. This approach consists of performing a depth-first search starting from patterns of length 1 (containing single activities), and recursively extending each pattern with additional activities to find larger patterns. For each visited pattern p of length k, a projected database is created, and then scanned to find extensions of p of length $(k+1)$. Formally, a pattern $e = \{r_1, r_2, ..., r_q\}$ is said to be an *extension* of a pattern $p = \{v_1, v_2, ..., v_o\}$ if there exists integers $1 \leq y_1 \leq y_2 \leq \ldots \leq y_k \leq o < q$ such that $r_{y1} = v_1$, $r_{y2} = v_2, \ldots, r_{yo} = v_o$. To avoid exploring all possible patterns, it is necessary to use techniques to reduce the search space. Hence, five search space pruning strategies are introduced. **Strategy 1, 2, 3, 4** are used in CEPDO, while **Strategy 1, 2, 3, 5** are used in CEPHU. Strategies are first described, and then the algorithms. The first strategy calculates an upper-bound on the occupancy measure introduced in [17] to reduce the search space.

Definition 7. ***(Upper bound on occupancy)*** *For a pattern p and sequence S_i, let $psl[S_i]$, $ssl[S_i]$ and $sl[S_i]$ be p's length in S_i, the length of the subsequence after p in S_i, and S_i's length, respectively. An* ***upper-bound on the occupancy of a pattern*** *p is:* $ou(p) = \frac{1}{sup(p)} \cdot \max_{S_1,\ldots,S_{sup(p)}} \sum_{i=1}^{\text{sup}(p)} \frac{psl[S_i]+ssl[S_i]}{sl[S_i]}$ [17].

Strategy 1 *For a pattern p, if $ou(p) < minoccup$, then pattern p and all its extensions have an occupancy lower than minoccup and can be eliminated* [17].

The second strategy relies on the support measure to reduce the search space and is commonly used in SPM [8].

Strategy 2 *For a pattern p, if $sup(p) < minsup$, then pattern p and all its extensions have a support lower than minsup and can be eliminated.*

The third strategy is novel, and based on the cost measure. A challenge for reducing the search space using the average cost is that this measure is neither monotonic nor anti-monotonic (proof is ommitted due to space limitation). For this reason, Strategy 3 relies on a new lower-bound on the average cost.

Definition 8. ***(Lower bound on the average cost).*** *Let $C(p)$ be the set of costs of a pattern p in a sequence S_s where $p \subseteq S_s$, and $c_i(p)$ denotes the i-th smallest cost of p in $C(p)$. The* Average Minimum Supported Cost *(AMSC) is a lower-bound on $ac(p)$, defined as: $amsc(p) = \frac{1}{minsup} \sum_{i=1,2,...,minsup} c_i(p)$.*

Theorem 1. ***(Underestimation property of the AMSC).*** *The AMSC of a pattern p is smaller than or equal to its true cost, that is $amsc(p) \leq c(p)$.*

Proof. Let $N = sup(p)$ and $M = minsup$. Without loss of generality, assume that all the cost values in $C(p) = \{c_1, c_2, \dots c_N\}$ are sorted in ascending order, i.e. $c_1 \leq c_2 \leq ... \leq c_M \leq ... \leq c_N$. Then, $\frac{1}{M} \sum_{i=1,2,...,M} c_i(p) \leq \frac{1}{N} \sum_{i=1,2,...,N} c_i(p)$, because cost values are in ascending order, and thus $(N - M) \sum_{i=1,2,...,M} c_i(p) \leq (N - M) \cdot M \cdot c_M(p) \leq M \cdot (N - M) \cdot c_{M+1}(p) \leq M \sum_{i=M+1,M+2,...,N} c_i(p)$.

Theorem 2. ***(Monotonicity property of the AMSC).*** *The AMSC measure is monotonic. Let $p_x \subseteq p_y$ be two frequent patterns. Then, $amsc(p_x) \leq amsc(p_y)$.*

Proof. Assume that $n = sup(p_x), m = sup(p_y)$ and $M = minsup$, then $n \geq m \geq M$. Because $\forall p_x \subseteq p_y$, $c(p_x, S_i) \leq c(p_y, S_i)$ where $p_y \subseteq S_i$. And all the cost values $c(p_x, S_i)$ and $c(p_y, S_i)$ are sorted in ascending order, $c(p_y, S_{i_k}) \geq c(p_x, S_i)$ where $\{i_1, i_2, ..., i_m\} \subseteq \{1, 2, ..., n\}$, therefore $amsc(p_y) = \frac{1}{M} \sum_{i=1,2,...,M} c(p_y, S_i) \geq amsc(p_x) = \frac{1}{M} \sum_{i=1,2,...,M} c(p_x, S_i)$.

Strategy 3 *For a pattern p, if $AMSC(p) > maxcost$, then p and its extensions have an average cost greater than maxcost, and thus can be eliminated.*

The fourth search space pruning strategy is novel and used by CEPDO. It consists of eliminating all patterns that do not exist in at least one sequence having a positive utility when mining patterns in a binary SADB (Problem 1).

Strategy 4 *For a pattern p, if $D_p^+ = \emptyset$, then pattern p and all its extensions can be eliminated.*

The rationale of this strategy is that if a pattern only appears in negative sequences, then this pattern is not interesting as one cannot evaluate if it contributes to the positive outcome. To prune the search space in Problem 2, where utility is a numeric value, a fifth strategy is presented. Because numeric utility is neither monotonic nor anti-monotonic, an upper-bound on the utility is proposed to reduce the search space. Note that this upper-bound is different from those used in HUSPM since in this paper each sequence is annoted with the utility rather than each activity.

Definition 9. ***(Upper bound on numeric utility).*** *For a pattern p, let $n = sup(p)$ and $M = minsup$. An upper-bound on the numeric utility of p is $upperu = \frac{1}{M} \sum_{i=1,2,...,n} u(p, S_i)$.*

Theorem 3. ***(Overestimation and antimonotonicity of the numeric utility).*** *Let p_x and p_y be two frequent patterns. Then, $u(p_x) \leq upperu(p_x)$. Moreover, if $p_x \subseteq p_y$, then $upperu(p_x) \geq upperu(p_y)$.*

Proof. Assume that $n = sup(p_x)$, $m = sup(p_y)$, *and* $M = minsup$. *Because* $n \geq m \geq M$, $upperu = \frac{1}{M}\sum_{i=1,2,\ldots,n} u(p, S_i) \geq \frac{1}{n}\sum_{i=1,2,\ldots,n} u(p, S_i)$. *Besides,* $\frac{1}{M}\sum_{i=1,2,..m,..,n} u(p, S_i) \geq \frac{1}{M}\sum_{i=1,2,\ldots,m} u(p, S_i)$.

Strategy 5 *For a pattern p, if* $upper(p) < minutil$, *p and its extensions have a utility lower than minutil and can be eliminated.*

The five strategies are designed to reduce the search space, and thus the time and memory required for finding cost-effective patterns in binary and numeric SADBs. The next paragraphs presents the two proposed algorithms.

The CEPDO algorithm (Algorithm 1) first scans the input SADB to calculate measures of length-1 patterns (Line 1). For a pattern p, if the support is not zero in positive sequences (**Strategy 4**) (Line 4), then the support, occupancy and average cost of p are compared with *minsup* (**Strategy 2**), *minoccup* and *maxcost* (Line 5). If the threshold requirements are passed, the correlation of p is calculated, and p is saved as a cost-effective pattern with its correlation (Line 6-7). Then, if p's $AMSC$ is no larger than *maxcost* (**Strategy 3**) and the occupancy upper-bound is no less than *minoccup* (**Strategy 1**), the algorithm creates p's projected database, and scans it to find extensions of the form $p \cup \{a\}$ where a is an activity (Line 8-9). The CEPDO function is then recursively called to enumerate all cost-effective patterns having p as prefix (line 9) following a depth-first search. Since the algorithm only eliminates non cost-efficient patterns using the designed strategies, all cost-efficient patterns are output.

```
   input  : a binary SADB D, minsup, maxcost, minoccup
   output: the cost-efficient patterns
 1 Scan D once to calculate the support, occupancy, average cost and AMSC of each
     length-1 pattern;
 2 def CEPDO():
 3 foreach activity p do
 4 |   if |D_p^+| ≠ 0 then
 5 |   |   if (sup(p) ≥ minsup and o(p) ≥ minoccup and (ac(p) ≤ maxcost) then
 6 |   |   |   if |D_p^+| = sup(p) then cor(p):=1;
 7 |   |   |   else Calculate(cor(p)) and Save(p);
 8 |   |   |   if (amsc(p) ≤ maxcost and ou(p) ≥ minoccup then
 9 |   |   |   |   foreach extension q = (p ∪ {a}) do call CEPDO(...);
10 |   |   |   end
11 |   |   end
12 |   end
13 end
```

Algorithm 1: The CEPDO algorithm

The CEPHU algorithm (Algorithm 2) scans the database to calculate the support of each length-1 pattern p (Line 1). For a pattern p, if the support is no less than *minsup* (**Strategy 2**) (Line 4), then the average cost, utility and occupancy of p are compared with *maxcost*, *minutil* and *minoccup* (Line 5). If the threshold requirements are met, the trade-off of p is calculated, and p is saved as a cost-effective pattern with its trade-off. Then, if p's $AMSC$ is no

larger than *maxcost* (**Strategy 3**), the occupancy upper-bound is no less than *minoccup* (**Strategy 1**), and the utility upper-bound is no less than *minutil* (**Strategy 5**), the algorithm creates p's projected database, and scans it to find extensions of the form $p \cup \{a\}$ where a is an activity (Line 6-7). The CEPHU function is then recursively called to find all cost-effective patterns having p as prefix following a depth-first search (line 7). Since CEPHU only eliminates non cost-efficient patterns using its strategies, it finds all cost-efficient patterns.

```
   input  : a numeric SADB D, minsup, maxcost, minoccup, minutil
   output: the cost-efficient patterns
 1 Scan D to calculate the support, occupancy, average cost, utility, upperu and ASMC of
     each length-1 pattern;
 2 def CEPHU():
 3 foreach activity p do
 4 |   if (sup(p) ≥ minsup) then
 5 |   |   if (ac(p) ≤ maxcost and u(p) ≥ maxutility and o(p) ≥ minoccup) then
   |   |     Calculate tf(p) and Save(p);
 6 |   |   if (amsc(p) ≤ maxcost and upperu(p) ≥ minutil) and ou(p) ≥ minoccup then
 7 |   |   |   foreach extension q = (p ∪ {a}) do call CEPHU(...);
 8 |   |   end
 9 |   end
10 end
```

Algorithm 2: The CEPHU algorithm

The proposed CEPDO and CEPHU algorithms can find cost-effective patterns in binary and numeric SADBs, respectively. To let users conveniently analyze patterns found, a user interface providing various visualizations were designed. This module will be described in the case study (Section 6).

5 Performance Evaluation

To evaluate the performance of the proposed CEPDO and CEPHU algorithms, experiments were carried out on a computer having a 64 bit Xeon E3-1270 3.6 Ghz CPU and 64 GB of RAM, running the Windows 10 operating system. Algorithms were implemented in Java, and source code is offered in the SPMF software [6]. Algorithms were evaluated in terms of performance on three standard benchmark datasets used in SPM, namely Bible, BMS and SIGN, obtained from the SPMF website. Those datasets have different characteristics such as dense, sparse, long and short sequences. In these datasets, the cost and utility values were randomly generated using a simulation model as in previous work in HUSPM [15]. The Bible dataset contains 36,369 sequences with 13,905 distinct items and an average sequence length of 44.3. The BMS dataset contains 59,601 sequences with 497 distinct items and an average sequence length of 6.02. The SIGN dataset contains 730 sequences with 267 distinct items and an average sequence length of 104.1. Because CEPDO and CEPHU are the first algorithms for mining cost-effective patterns, they cannot be compared with previous SPM and HUSPM techniques. For this reason, two versions of each algorithm were

compared using *i*.Strategy 3 based on the proposed AMSC lower-bound on the average cost and *ii*. without using Strategy 3. The execution times of the two versions of CEPDO and CEPHU are shown in Fig. 1 A and Fig. 1 B for each dataset, while the *maxcost* threshold is varied. As shown in Fig. 1, using Strategy 3 with the AMSC decreases runtime by up to 4 times. Besides, as the *maxcost* threshold is increased, the version using AMSC become more efficient. Because of space limitation and that occupancy and support were used in previous papers, results obtained when varying the *minsup* and *minoccup* threshold are not presented. It is found that the AMSC pruning strategy greatly improves efficiency.

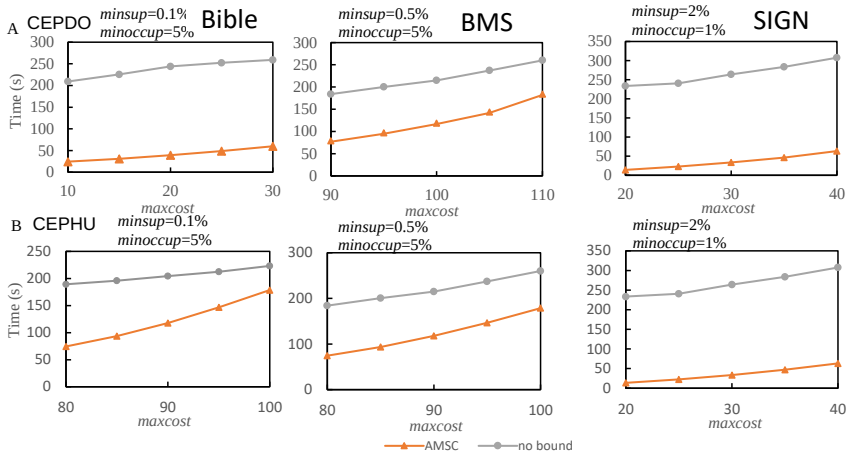

Fig. 1. Runtimes of CEPHU/CEPDO with/without Strategy 3 when *maxcost* is varied

6 Case Study in E-Learning

To evaluate how useful the proposed patterns are, the developed algorithms were applied on e-learning data of Vahdat et. al [16] to find cost-effective patterns that can lead to obtaining high scores. This data consists of logs of the Deeds e-learning environment, for learning digital electronics. The environment provides learning materials. Furthermore, learners are asked to solve problems of different levels of difficulty to assess their knowledge. The dataset contains sequence of activities performed by each student in each learning sessions. The time spent for each activity by a student is indicated. Moreover a score is given at the end of each session, and a score is provided for the fnal exam.

Finding patterns using CEPDO. CEPDO was first applied to find cost-effective sequences of sessions to pass the final exam. Each sequence consists of several sessions, each annotated with a time cost. The utility of each sequence is the exam score, transformed into binary classes ($passed/failed$). CEPDO was applied with $minsup = 0.5$, $minoccup = 0.5$, and $maxcost = 600$. Some

patterns providing interesting insights were discovered. For example, the patterns $\{1, 2, 5, 6\}$, $\{1, 2, 6\}$ and $\{1, 5, 6\}$ have a large positive correlation (0.21, 0.20, 0.19), respectively, where numbers represents session IDs. It was also found that some patterns such as $\{4, 5\}$ and $\{5\}$ have a negative correlation (-0.11 and -0.15, respectively). Moreover, it was found that some patterns such as $\{2, 3\}$, $\{3, 4, 5, 6\}$ are barely correlated with the final exam result as their correlation are both 0.001. Overall, based on these patterns, it is found that students who learnt Session 1, Session 2, Session 5 and Session 6 are more likely to *pass* the final exam. On the other hand, if a student only studies Session 5, or Session 4 and Session 5, he is more likely to $fail$ the exam. Besides, if a student spends time on other unrelated sessions, it may increase the study time but not increase much his chances of passing the exam.

Visualizing patterns found by CEPDO. We designed a visualization system to further investigate the relationship between cost and utility for each pattern found by CEPDO. Charts in the top row of Fig. 2 shows the cost-utility distribution graph of three patterns ($\{1, 2, 5\}$, $\{4, 5\}$ and $\{2, 3\}$). Since a pattern may have the same cost but different outcomes in multiple sequences, the average outcome was calculated for each cost value (a value in [-1,1]). It can be seen that the pattern $\{1, 2, 5\}$, which has the largest positive correlation, has a dense distribution (Figure 2) and for this pattern a larger cost is spent in sequences leading to the desirable outcome (1) than the negative outcome (-1). Pattern $\{4, 5\}$, has the most dense distribution and most cost is spent in sequences leading to the undesirable outcome (-1), which is reasonable since $\{4, 5\}$ has the largest negative correlation. Pattern $\{2, 3\}$'s distribution does not show a considerable difference for all sequences in terms of cost, which is reasonable since its correlation is 0.001. Another observation is that because few students passed the exam for the 60 point threshold, the correlation is on overrall small. If that threshold is reduced, correlation values increase.

Finding patterns using CEPHU. In another experiment, the designed CEPHU algorithm was applied to the same e-learning database to analyse activities within sessions rather than analyzing the whole learning process. The database contains multiple sequences for the same session (one for each student) in which each activity (learning material) has a time cost. The score of each session exam is the utility of the sequence. The goal is to obtain insights about how to efficiently use learning materials to obtain high scores in a session test. The algorithm was applied to data from six sessions to find patterns that have a small trade-off and lead to a high score. It was found that it's unreasonable to recommend the patterns that only have a small trade-off, because even though some patterns have a small trade-off, they sometimes also have a low utility. Thus, we defined several utility ranges (e.g. $[14, 15)$, $[15, 16)$), and sorted patterns by trade-off in each utility range. By considering ranges, interesting patterns having a relatively small trade-off but a high utility are observed. For example, consider session 6, $minsup = 0.1$, $occup = 0.28$, $minutility = 10$ and $maxcost = 100$. To obtain the average score of 15, the most cost-effective pattern is (Study_Es_6_2)(Deeds_Es_6_2)(Study_Es_6_3), having a trade-off of

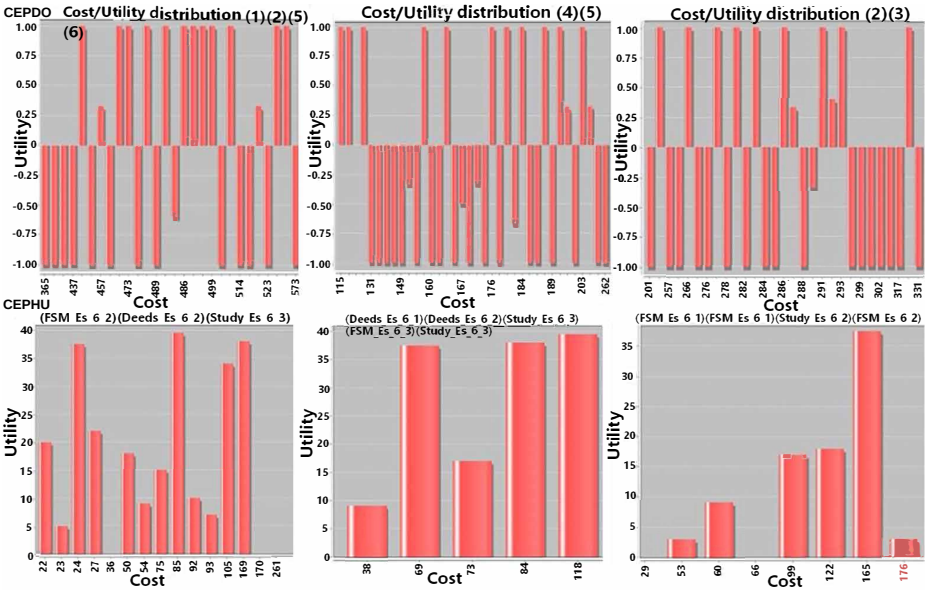

Fig. 2. Distribution graph in terms of cost-utility

1.74. But, the most cost-effective pattern to obtain a high score of 28/40 is (Deeds_Es_6_1)(Deeds_Es_6_2)(Study_Es_6_3)(FSM_Es_6_3) (Study_Es_6_3) with a trade-off of 2.71. Pictures of the bottom row of Fig. 2 show the visualization of the cost/utility distribution of three patterns found by CEPHU, which haves utility of 15, 28, and 10, respectively. Two observations are made from these charts. First, although most effective patterns have a relatively small cost, they can still lead to a relatively high score. Second, we can find what is the proper cost range that should be spent on a pattern to obtain a target score. For example, Fig. 2 (bottom row) shows that (Deeds_Es_6_1)(Deeds_Es_6_2)(Study_Es_6_3)(FSM_Es_6_3)(Study_Es_6_3) generally provides a high score at a low cost compared to the other patterns. And the cost range where this pattern is most effective is approximately [69, 118].

After discovering the patterns, we have also carefully looked at the questions in each session's final exam and compared them with the materials of the mined patterns. This has confirmed that there is indeed a strong correlation between patterns and exam questions, which indicates that the patterns are reasonable. Then, we also applied the CEPDO algorithm for individual sessions by transforming the sequence utility to a binary class based on the average score. It was found that several patterns having a large positive correlation also have a high utility and a low trade-off. This again show that the proposed measures to find cost-effective patterns are reasonable.

Visualizing the skyline of patterns found by CEPDO and CEPHU. To give users a more clear view of patterns found by the algorithms, a second visualization was implemented based on the concept of skyline [14]. Fig. 3 displays

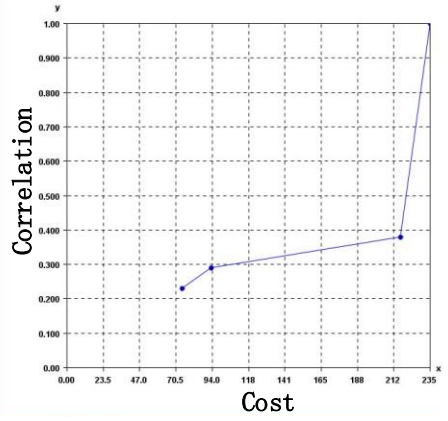

Fig. 3. Skyline patterns of CEPDO

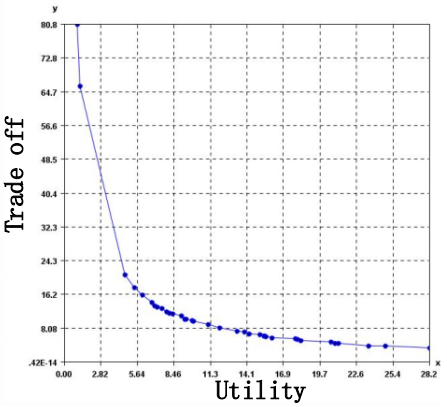

Fig. 4. Skyline patterns of CEPHU

the skyline of patterns that have been extracted by CEPDO. Each data point represents the pattern p having the highest correlation for its average cost value $ac(p)$. The skyline is useful as it lets users quickly find the patterns having the highest correlation for each cost value, and thus to more easily browse patterns. Similarly, Fig. 4 displays the skyline of patterns that have been extracted by CEPHU. Each data point represents the pattern p having the lowest trade-off for its utility value. It is interesting to see that patterns scattered in the utility range [16.9, 28.2] are very efficient, while those scattered in [0 to 2.82] have a high cost but lead to a low utility. Note that the designed skyline visualization system for patterns can be used using other pairs of measures such as $\{utility, occupancy\}$ or $\{support, utility\}$ but this feature is not illustrated due to space limitation.

On overall, Section 6 has shown that insightful cost-efficient patterns have been found in e-learning data using the proposed algorithms.

7 Conclusion

This paper proposed two algorithms named CEPDO and CEPHU for mining cost-effective patterns in sequences of activites with cost/utility sequences, where utility is either binary or numeric values. A tight lower bound on the cost and an upper bound on the utility was proposed to efficiently find cost-effective patterns. Furthermore, occupancy and support measures were integrated into the algorithms to eliminate non representative and infrequent patterns. Moreover, a visualization system was developed to show the cost-utility distribution of patterns and browse patterns using their skyline. Experimental results have shown that the algorithms have good performance and a case study has shown that interesting patterns are discovered in e-learning data.

In future work, we will consider optimizing the proposed algorithms, and extending the model to other types of patterns such as periodic patterns [7],

peak patterns [10], and itemsets [9]. Moreover, applications of the algorithms in other domains such as for analyzing business procsses will be investigated.

Acknowledgement. This project was funded by the National Science Fundation of China and the Harbin Institute of Technology (Shenzhen).

References

1. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proc. of the 11th Intern. Conf. on Data Engineering. pp. 3–14. IEEE (1995)
2. Alkan, O.K., Karagoz, P.: Crom and huspext: Improving efficiency of high utility sequential pattern extraction. IEEE Transactions on Knowledge and Data Engineering **27**(10), 2645–2657 (2015)
3. Bogarín, A., Cerezo, R., Romero, C.: A survey on educational process mining. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **8**(1) (2018)
4. Choi, D.W., Pei, J., Heinis, T.: Efficient mining of regional movement patterns in semantic trajectories. Proc. of VLDB Endowment **10**(13), 2073–2084 (2017)
5. Dalmas, B., Fournier-Viger, P., Norre, S.: Twincle: A constrained sequential rule mining algorithm for event logs. Procedia Computer Science **112**, 205–214 (2017)
6. Fournier-Viger, P., Gomariz, A., Gueniche, T., Soltani, A., Wu, C.W., Tseng, V.S.: Spmf: a java open-source pattern mining library. The Journal of Machine Learning Research **15**(1), 3389–3393 (2014)
7. Fournier-Viger, P., Li, Z., Lin, J.C.W., Kiran, R.U., Fujita, H.: Efficient algorithms to identify periodic patterns in multiple sequences. Information Sciences (2019)
8. Fournier-Viger, P., Lin, J.C.W., Kiran, R.U., Koh, Y.S., Thomas, R.: A survey of sequential pattern mining. Data Science and Pattern Recognition **1**(1), 54–77 (2017)
9. 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)
10. Fournier-Viger, P., Zhang, Y., Lin, J.C.W., Fujita, H., Koh, Y.S.: Mining local and peak high utility itemsets. Information Sciences **481**, 344–367 (2019)
11. Garcıa-Algarra, J.: Subgroup discovery in process mining. In: Proc. of the 20th Intern. Conf. on Business Information Systems. vol. 288, p. 237. Springer (2017)
12. Glass, G.V., Hopkins, K.D.: Statistical methods in education and psychology. Pearson (1996)
13. Ranjan, J., Malik, K.: Effective educational process: a data-mining approach. Vine **37**(4), 502–515 (2007)
14. Soulet, A., Raïssi, C., Plantevit, M., Cremilleux, B.: Mining dominant patterns in the sky. In: IEEE 11th Intern. Conf. on Data Mining. pp. 655–664. IEEE (2011)
15. Truong-Chi, T., Fournier-Viger, P.: A survey of high utility sequential pattern mining. In: High-Utility Pattern Mining, pp. 97–129. Springer (2019)
16. Vahdat, M., Oneto, L., Anguita, D., Funk, M., Rauterberg, M.: A learning analytics approach to correlate the academic achievements of students with interaction data from an educational simulator. In: Proc. 10th European Conference on Technology Enhanced Learning, pp. 352–366. Springer (2015)
17. Zhang, L., Luo, P., Tang, L., Chen, E., Liu, Q., Wang, M., Xiong, H.: Occupancy-based frequent pattern mining. ACM Transactions on Knowledge Discovery from Data **10**(2), 14 (2015)