# Discovering Low-Cost High Utility Patterns


Philippe Fournier-Viger*
Harbin Institute of Technology (Shenzhen)
Shenzhen, China
philfv8@yahoo.com

Jiaxuan Li
Harbin Institute of Technology (Shenzhen)
Shenzhen, China
jiaxuanliniki@gmail.com

Jerry Chun-Wei Lin
Harbin Institute of Technology (Shenzhen)
Shenzhen, China
jerrylin@ieee.org

Tin Truong-Chi
University of Dalat
Dalat, Vietnam
tintc@dlu.edu.vn



## ABSTRACT

High Utility Pattern Mining (HUPM) is an emerging research topic in data mining, which consists of discovering patterns having a high utility (importance) in databases. However, a fundamental limitation of HUPM is that it is focused on the utility or benefits provided by patterns but completely ignores the cost or effort required to obtain these benefits. Generally, the cost of a pattern can be expressed in terms of various aspects such as time, money, resources consumed and effort. Because HUPM does not consider the cost of patterns, it can find numerous patterns that have a high utility but a very high cost and miss numerous patterns that have very low cost but a relatively high utility. For example, in the context of hospital data, many patterns may be found that lead to a desirable outcome such as being cured but these patterns may have a cost that is prohibitive. This paper addresses this important limitation of HUPM by defining the problem of discovering Low-cost High Utility Patterns in sequences. Three sub-problems are defined corresponding to three cases where (1) the utility is represenbted as binary classes representing a desirable and undesirable outcome (e.g. *cured* or *died* after some medical treatments), (2) the utility is represented as a numeric value (e.g. score obtained at an exam), and (3) the utility is represented using binary classes but where only records representing the positive class are available. For each of these cases, utility and cost are represented separately, and appropriate measures are designed to assess the trade-off and correlation between cost and utility. To efficiently find low-cost high utility patterns, three algorithms are designed, corresponding to the three cases. They rely on a novel lower-bound on the average cost of patterns to reduce the search space. An experimental study shows that the proposed algorithms are efficient and can discover interesting patterns in e-learning data.



*Corresponding author.





*UDM 2018 Workshop, August 2018, London, UK*





ACM ISBN 123-4567-24-567/08/06...$15.00
https://doi.org/10.475/123_4



## CCS CONCEPTS

• **Information systems** → *Information extraction*;

## KEYWORDS

Pattern mining, sequential patterns, high utility patterns, low-cost patterns, trade-off, e-learning

**ACM Reference Format:**

Philippe Fournier-Viger, Jiaxuan Li, Jerry Chun-Wei Lin, and Tin Truong-Chi. 2018. Discovering Low-Cost High Utility Patterns. In *Proceedings of ACM conference (UDM 2018 Workshop),* Jennifer B. Sartor (Ed.). ACM, New York, NY, USA, Article 4, 9 pages. https://doi.org/10.475/123_4


## 1 INTRODUCTION

Frequent Pattern Mining (FPM) [6] is a popular data mining task, which discovers frequently occurring sets of items (frequent itemsets) in customer transactions. Although frequent itemsets provide information that can be used for decision making, it has some important limitations such that it does not consider the time or sequential ordering of transactions. To overcome this issue, FPM was generalized as Sequential Pattern Mining (SPM) [1, 13]. The goal of SPM is to find frequently occuring subsequences in a set of sequences. SPM has broad applications in real life, such as analyzing customers' purchase habits, DNA sequences and business processes. Some early sequential pattern mining algorithms are AprioriAll and GSP [1, 13]. Given a minimum support (frequency) threshold *minsup*, they enumerate all sequences appearing in at least *minsup* sequences using a breadth-first search. Then, several algorithms were proposed to improve the performance of SPM such as SPADE, which relies on a vertical database representation [19], and FreeSpan [10], which adopts a pattern-growth approach to avoid generating candidates. SPM is a very active research area and novel algorithms and extensions are frequently proposed.

One of the most important application of SPM is to analyse customers' behavior. However, a major limitation of SPM is that it does not consider the purchase quantities and unit profits of items. To find patterns that yield a high profit (or more generally, have a high importance) rather than frequent patterns, SPM has been generalized as High Utility Pattern Mining (HUPM). Given a sequence database with profit (utility) information and a user defined minimum utility threshold, HUPM discovers all profitable (high utility) subsequences. However, HUPM is a much more difficult problem than FPM because the downward closure property of the support typically used in SPM to reduce the search space does not hold for the utility measure used in HUPM, which means that the utility of a pattern can be either higher, equal or lower than that of its subsets [7, 14, 18].

HUPM is useful for many applications as it can discover high utility patterns in data, where the utility is a numerical measure of the importance or benefit provided by a pattern (e.g. the profit). For example, a pattern that may be discovered in customer data is $\langle smartphone, headphone, battery\rangle$ indicating that these items are typically purchased in that order and yield a high profit. However, a fundamental limitation of HUPM is that it is focused on the utility or benefits provided by patterns but completely ignores the cost or effort required to obtain the benefits of these patterns. For example, in the context of hospital data, a pattern $\langle medicine, treatment, cured\rangle$ may be found indicating that taking a given medicine and treatment may result in getting cured, where being cured may be considered as the utility of the pattern. However, if HUPM is applied on such data, it will ignore the cost of the medicine, treatment and effort to obtain the desired result of being cured. Generally, the cost of a pattern can be expressed in terms of aspects such as time, money, resources consumed and effort.

Because HUPM does not consider the cost of high utility patterns, it can find numerous patterns that have a high utility but have a high cost. For example, in the context of hospital data, many patterns may lead to a desirable outcome (have a high utility) but have a cost that is prohibitive. Moreover, HUPM may fail to discover patterns that have a slightly lower utility but a much lower cost. Thus, high utility patterns may be highly misleading for decision makers, as they do not mention the cost required by the patterns to achieve the utility. Integrating the concept of cost in HUPM is thus desirable but it is not a trivial task since cost and utility may be measured in terms of different aspects (e.g. profit vs time spent). Moreover, cost and utility should not be combined in a naive way by subtracting the cost from the utility because it would not allow assessing how strong is the correlation between the cost and the utility for each pattern. In fact, it is desirable to discover patterns that not only have a low cost and high utility but that also provide a good trade-off between the cost and utility, and have a strong correlation between the cost and utility. Thus, a new model must be proposed and efficient techniques to find such patterns.

In this paper, we address this challenge by defining a novel problem of discovering patterns providing a good trade-off between the cost and utility in sequences containing utility and cost information. These patterns are called Low-Cost High Utility Patterns (LCHUP). The main contributions of this study are as follows.

- The problem of discovering Low-cost High Utility Patterns in sequences is proposed. Three sub-problems are defined corresponding to three cases occuring in real-life where (1) the utility is represented as binary classes representing a desirable and undesirable outcome (e.g. *cured* vs *died*), (2) the utility is represented as a numeric value (e.g. the score obtained at an exam), and (3) the utility is represented using binary classes but where only records representing the positive class are available. For each of these cases, appropriate statistical measures are designed to assess the correlation between cost and utility. Moreover, the properties of the proposed problems are studied.
- A novel lower-bound on the average cost of patterns is designed to be able to reduce the search space and discover patterns efficiently. Based on these theoretical results, three algorithms are designed to find patterns for the three considered cases. The algorithms adopt a pattern-growth approach to efficiently discover patterns.
- An experimental study has been carried to evaluate the performance of the proposed algorithm on several datasets for several parameters values. Results show that the algorithms are efficient and that the proposed lower-bound can considerably reduce the search space. A case study on e-learning data also shows that interesting patterns are found.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 defines the problem. Section 4 presents the proposed algorithms. Section 5 describes the experimental evaluation. Finally, Section 6 draws the conclusion.

## 2 RELATED WORK

Some prelimiary work toward the integration of cost to analyse people's behavior have been done in the field of Process Mining (PM). PM is a subfield of data mining that aims at extracting useful information from event logs such as patterns and models, which characterise a business process [17]. PM has been used to analyze various types of event logs from domains such as business, education and healthcare [2, 9, 16], using both unsupervised and supervised methods [5, 12]. An event log is a sequence of events, each representing an activity, and annotated with a timestamp. Mannhardt et al. [8] applied PM to analyze pathways of inpatients with the sepsis condition. The result is a graph where nodes denotes activities and edges represents the temporal relationships between these activities. For this application, PM allowed to extract guidelines to improve sepsis treatment. However, this studies also highlighted limitations of PM such as the extraction of a graph structure representing a process, which is difficulty understandable by patients and doctors. This is an important problem since for more complex business processes, the produced model (graph) can be much larger and complex. Some other limitations of that study are that the model only shows relationships between consecutive activities (it cannot skip some activities) and do not consider the cost. Recently, Dalmas et al. [3] proposed a pattern mining algorithm named TWINCLE for analyzing event-logs of hospitals, where each patient activities has a cost. The TWINCLE algorithm finds sequential rules, where the antecedent and consequent of a rule are events. The rules can be used to predict what will happen to a patient if some events occur. Rules are selected based on their cost and displayed to the user with the aim of reducing the monetary cost of medical treatments. The algorithm first generates rules having two events and then recursively produces larger rules from these rules using a depth-first search. The user must specify several parameters in terms of pattern length, cost, confidence and time. Although interesting low cost patterns were extracted by TWINCLE, a major limitation is that it only focuses on the cost of patterns and ignore the utility. In this paper, we argue that both factors must be considered as well as their relationship, to find patterns offering a good trade-off between utility and cost.

Another related work is emerging pattern mining, which aims at discovering patterns that appear significantly more often in a class than in another class. For example, Poon et al. [11] proposed to discover emerging sequential patterns in educational data. The data is an event log from a statistic course, which includes series of activities with timestamps such as access to simulations, videos and quizzes. To analyze the behavior of learners, they were categorized in two groups (class labels) based on quiz scores (above and below average scores). Then, patterns were extracted that discriminate between the two groups to recommend better ways of using e-learning resources. However, a major limitation of this work is that it does not consider the cost such as the time spent by students for each activity to obtain high scores. Moreover, score is defined as a binary variable rather than a numeric variable, which results in informatin loss. It is thus desirable to find methods that can identify patterns having a low cost and high utility (e.g. high scores).

To address the above limitations of previous work, the next section proposes the novel problem of discovering low-cost high utility patterns by combining both the concept of utility and cost for pattern extraction. In the proposed model, the cost is viewed as a numeric value associated to each event, while the utility is either defined as a binary class label (e.g. *cured* or *died*) or a numeric value (e.g. an exam score) that is associated to each sequence. Moreover, a concept of trade-off is also introduced to assess the correlation between utility and cost.

## 3 PROBLEM DEFINITION

This section presents the proposed task of mining LCHUP. Two types of sequence databases are considered, for the needs of different real-life applications. The first one considers that the utility is represented as a binary class, while the second one considers that it is a numerical value. From these definitions, three versions of the problem of LCHUP mining are defined. The next paragraphs introduce important preliminary definitions.

**Table 1: SADB with Binary Classes**

| Sid | Sequence (activity[cost]) | Class |
|---|---|---|
| 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])> | + |

*Definition 3.1.* (Sequential Activity Database) A sequential activity database (SADB) is a set of sequences $SADB = \{S_1, S_2, \dots S_n\}$, where each sequence $S_s$ has a sequence identifier $s$ $(1 \leq s \leq n)$. A sequence $S_s$ is a list of activities $\langle \{v_1[c_1], v_2[c_2], \dots v_m[c_m]\}\ |Utility\rangle$, where the notation $v_i[c_i]$ indicates that an activity $v_i$ was performed with a cost $c_i$ (a positive number). In a *binary SADB*, the value $Utility$ of a sequence $S_s$ is a binary class label $+$ or $-$, respectively representing a positive or negative outcome. In a *numeric SADB*, the value $Utility$ is a positive number. Activities in sequences are ordered by ascending order of timestamps.

For example, Table 1 illustrates a binary SADB containing five sequences. The first sequence indicates that the activity $a$ was performed with a cost of 2, followed by $b$ with a cost of 4, $c$ with a cost of 9, and $d$ with a cost of 2. The utility of this sequence is the class label $+$ (a positive outcome).

*Definition 3.2.* (Pattern) A pattern $p$ is an ordered set of activities $\{v_1, v_2, ..., v_o\}$. The pattern $p$ is a sub-pattern of another pattern $q = \{w_1, w_2, ..., w_p\}$ if there exists integers $1 \leq x_1 \leq x_2 \leq \ldots \leq x_k \leq o$ such that $w_{x1} = v_1$, $w_{x2} = v_2, \ldots, w_{xo} = v_o$. A pattern $e = \{r_1, r_2, ..., r_q\}$ is an extension of pattern $p$ if there exists integers $1 \leq y_1 \leq y_2 \leq \ldots \leq y_k \leq o < q$ such that $r_{y1} = v_1$, $r_{y2} = v_2, \ldots, r_{yo} = v_o$. In other words, an extension of a pattern is a pattern that has the same prefix but some additional activities. The pattern $p$ is said to appear in a sequence $S_s = \langle \{v'_1[c_1], v'_2[c_2], \ldots v'_m[c_m]\}\ |Utility\rangle$ (denoted as $p \subseteq S_s$) if there exists integers $1 \leq a_1 \leq a_2 \leq \ldots \leq a_k \leq o$ such that $v'_{a1} = v_1$, $v'_{a2} = v_2, \ldots, v'_{ao} = v_o$. For the smallest such set of integers $a_1, a_2, \ldots a_k$ such that $p \subseteq S_s$, the set of activities $\{v'_{a1}, v'_{a2} \ldots v'_{ao}\}$ is said to be the *first occurrence* of $p$ in $S_s$ and is denoted as $first(p, S_s)$.

*Definition 3.3.* (Cost of a pattern) 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. The cost of a pattern $p$ in a SADB is the sum of its cost in all sequences, i.e. $c(p) = \sum_{p \subseteq S_s \wedge S_s \in SADB} c(p, S_s)$.

*Definition 3.4.* (Database projection) The projection of a sequence $S_s$ by a pattern $p$ is the part of sequence $p$ that appears after the first occurrence of $p$ in $S_S$ or the empty sequence if $p$ does not appear in $S_s$. The projection of a database SADB by a pattern is the set of all projected sequences by $p$.

For example, the projected database of pattern $\{a, b\}$ in the database of Table 1 is $\langle\{c[9], d[2]\}|+\rangle$, $\langle\{d[1]\}|-\rangle$.

*Definition 3.5.* (Support of Pattern) The support of a pattern $p$ is denoted as $sup(p)$ and is defined as the number of sequences that contains $p$. In other words, $sup(p) = |\{S_s | p \subseteq S_s \Lambda S_s \in SADB\}|$.

*Definition 3.6.* (Average Cost of Pattern) The average cost of a pattern $p$ is defined as: $ac(p) = \frac{\sum_{p \subseteq S_s \Lambda S_s \in SADB} c(p, S_s)}{|sup(p)|}$.

For example, consider the pattern $\{a, b\}$ and the SADB of Table 1. The cost of $\{a, b\}$ in $S_1$ is $c(\{a, b\}, S_1) = 2 + 4 = 6$, and the cost of $\{a, b\}$ in the whole database is $c(\{a, b\}) = c(\{a, b\}, S_1) + c(\{a, b\}, S_3) + c(\{a, b\}, S_4) = (2 + 4) + (5 + 8) + (3 + 5) = 27$. Since, sup({a,b}) = 3, the average cost of $\{a, b\}$ is $\frac{27}{3} = 9$.
While the support is an antimonotonic measure that can be used to reduce the search space, the average cost is neither monotonic nor anti-monotonic.

Property 1. *The support is antimonotonic, that is the support of a pattern must be greater or equal to the support of its extensions [1].*

Property 2. *The average cost is not monotonic nor anti-monotonic, that is a pattern's average cost may be smaller, equal or greater than that of its extensions.*

The above property is proved using an example. In Table 1, $ac(a) = c(\{a\}, S_1) + c(\{a\}, S_3) + c(\{a\}, S_4))/sup(\{a\}) = (2 + 5 + 3)/3 = 3.3$, while $ac(d) = c(\{d\}, S_1) + c(\{d\}, S_2) + c(\{d\}, S_4))/\ sup(\{d\}) = (2 + 12 + 1)/3 = 5$ and $ac(\{a, d\}) = c(\{a, d\}, S_1) + c(\{a, d\}, S_4))/\ sup(\{a, d\}) = \frac{(4+4)}{2} = 4$. Thus the average cost cannot be used to prune the search space. To solve this problem, a lower-bound on the average-cost will be introduced in Section 4.

## 3.1 Positive Patterns in a Binary SADB

The first version of the proposed problem aims at discovering low cost patterns in a SADB where the utility is a binary value (positive or negative). Many real-life databases of this type exists such as medical pathways where a patient may be cured or may die as a result of medical treaments, and e-learning data where a student may pass or fail an exam as a result of learning events. The simplest way of analyzing such databases is by considering only the positive class to mine low-cost patterns associated to the positive class. This problem is defined as follows:

*Definition 3.7.* (Problem Definition) Given a minimum support threshold *minsup* and a maximum cost *maxcost*, the problem of mining positive patterns in a binary SADB is to find each pattern $p$ such that $(\text{sup}(p) \geq minsup) \wedge (ac(p) \leq maxcost)$.

**Table 2: Positive Low-Cost Patterns**

| Pattern | Average cost | Pattern | Average cost |
|---|---|---|---|
| $\{a\}$ | 3.5 | $\{c\}$ | 5.5 |
| $\{e\}$ | 4.0 | $\{b, c\}$ | 9.0 |
| $\{b\}$ | 5.0 | $\{a, b\}$ | 9.5 |

Thus, low cost patterns that can provide a high utility may be found. Patterns can then be sorted by decreasing average cost before being presented to the user. For example, sequences corresponding to the positive classes in the SADB of Table 1 are $S_1$, $S_3$ and $S_5$. For $minsup = 2$ and $maxcost = 20$, six patterns are found, as shown in Table 2.

## 3.2 Correlated Patterns in a Binary SADB

Although mining positive patterns in a binary SADB is interesting, it has an important limitation, which is that only positive sequences are considered and that the correlation between a pattern and the utility is not measured. Thus, some patterns may be misleading to users as they may also appear in negative sequences. To address this issue, the problem of mining correlated low-cost patterns in a binary SADB is defined based on the following definitions.

Consider a binary SADB containing both sequences with positive and negative class labels such as the database of Table 1. Let $D_p^+$ and $D_p^-$ respectively denote the set of positive and negative sequences containing the pattern $p$ in SADB. We define the correlation between a pattern and the utility as follows:

*Definition 3.8.* (Correlation with binary utility) The correlation of a pattern $p$ to the utility in a binary SADB is defined as: $cor(p) = \frac{ac(D_p^+) - ac(D_p^-)}{Std} \sqrt{\frac{\text{sup}(D_p^+)}{|D_p|} \frac{\text{sup}(D_p^-)}{|D_p|}}$ where $ac(D_p^+)$ and $ac(D_p^-)$ respectively denote the pattern $p$'s average cost in $D_p^+$ and $D_p^-$, $Std$ is the standard deviation of $p$'s cost and $sup(D_p^+)$, $sup(D_p^-)$ are respectively the support of $p$ in $D_p^+$ and $D_p^-$.

For example, in Table 1, $cor(\{c\}) = \frac{(\frac{9+2}{2} - \frac{10}{1})}{Std(9,2,10)} \sqrt{\frac{2}{3} \times \frac{1}{3}} \approx -0.60$. The proposed correlation measure adapts the concept of biserial correlation used in statistics to assess the correlation between a binary attribute and a numeric attribute. The range of the *cor* measure is $[-1, 1]$. If the value is positive, it means that there is a positive correlation between the pattern and a positive utility, while a negative value indicates a negative correlation. A larger (smaller) positive (negative value) indicates a greater positive (negative) correlation. For example, $cor(\{a, b\}) = 0.8$ indicates that {a,b} is strongly correlated with the positive class, while $cor(\{a, c\}) = -0.5$ indicates that this pattern is correlated with the negative class.

**Table 3: Correlated Low-Cost Patterns (Binary classes)**

| Pattern | Correlation | Average cost |
|---|---|---|
| $\{e\}$ | 1.0 | 3.0 |
| $\{b\}$ | 0.42 | 4.2 |
| $\{a, b\}$ | 0.24 | 9.0 |
| $\{a\}$ | 0.19 | 3.3 |
| $\{b, c\}$ | $-0.28$ | 9.7 |
| $\{d\}$ | $-0.43$ | 5.0 |
| $\{b, d\}$ | $-0.50$ | 8.3 |
| $\{c\}$ | $-0.60$ | 7.0 |

*Definition 3.9.* (Problem Definition) Given a minimum support *minsup* and a maximum cost *maxcost*, the problem of mining correlated low-cost patterns in a binary SADB is to find each pattern $p$ such that $(\text{sup}(p) \geq minsup) \wedge (ac(p) \leq maxcost)$ and calculate its correlation $cor(p)$.

The goal is thus to find low-cost patterns where cost is correlated with a high utility. For example, for the database of Table 1, $minsup = 3$ and $maxcost = 10$, eight patterns are found, as shown in Table 3. It is observed that several patterns have a negative correlation such as $\{c\}$, $\{b, d\}$, $\{d\}$, and $\{b, c\}$. Thus, considering both positive and negative classes with the *cor* measure is useful to identify the truly positive patterns.

## 3.3 Correlated Patterns in a Numeric SADB

In the problems of Section 3.1 and Section 3.2, utility is binary. However, in many real-life activity logs, the utility is instead represented as numeric values. For example, in e-learning activity logs, exam scores may be values in the [0,100] interval. For instance, a numeric SADB is shown in Table 4. The previous problem definitions are unsuitable for mining low-cost patterns in such data. Thus, this subsection presents an adaptation of the problem of low-cost high utility pattern mining for numeric data. We redefine the concept of a pattern's utility and introduce a novel measure called trade-off to assess the relationship between cost and numeric utility.

*Definition 3.10.* (Utility of a Pattern) Let $su(S_s)$ denotes the utility of a sequence $S_s$. The utility of a pattern $p$ in a numeric SADB is the average of the utility of sequences in which it appears, that is $u(p) = \frac{\sum_{p \subseteq S_s \wedge S_s \in SADB} su(S_s)}{|sup(p)|}$.

*Definition 3.11.* (Trade-off) The trade-off of a pattern $p$ aims at measuring the efficiency of a pattern. It is the ratio of the average cost to the average utility: $tf(p) = ac(p)/u(p)$.

**Table 4: A numeric SADB**

| Sid | Sequence (activity[cost]) | Utility |
|---|---|---|
| 1 | <(a[20]),(b[40]),(c[50]),(d[20])> | 80 |
| 2 | <(b[25]),(d[12]),(c[30]),(e[25])> | 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 |

**Table 5: Low-Cost Patterns with Trade-off**

| Utility | Pattern | Trade-off |
|---|---|---|
| 70 | {b, c} | 0.88 |
| 65 | {b, e} | 0.72 |
| 60 | {b, d} | 0.85 |
| 56 | {a, b} | 1.01 |

For instance, the trade-off of $\{a, d\}$ in Table 4 is $tf(\{a, d\}) = ac(\{a, d\})/u(\{a, d\}) = \frac{[c(\{a,d\},S_1)+c(\{a,d\},S_4)]/sup(\{a,d\})}{[su(S_1)+su(S_4)]/2}$ $= \frac{[(20+20)+(40+40)]/2}{[(80+40)]/2} = 1$.

The trade-off of a pattern is a value in the $(0, \infty)$ interval. If a trade-off is small, it indicates that this pattern is efficient, that is that utility is obtained at a low cost by using this pattern.

*Definition 3.12.* (Problem Definition) Given a minimum support *minsup* and a maximum cost *maxcost*, the problem of mining correlated patterns low-cost patterns in a numeric SADB is to find each pattern $p$ such that $(\text{sup}(p) \geq minsup) \wedge (ac(p) \leq maxcost)$ and calculate its trade-off $tf(p)$.

The above problem allows to find patterns with a small $tradeoff$ under the conditions of minimum support and maximum cost. For each utility values, the most efficient patterns can then be presented to the user. For example, for $minsup = 2$ and $maxcost = 100$, four patterns are found in Table 4, shown in Table 5. The pattern $\{b, e\}$ is considered the most efficient.

# 4 PROPOSED ALGORITHM

This section presents algorithms to efficiently discover LCHUP for the three problems introduced in the previous section. The basic search procedure for exploring the search space is based on the PrefixSpan algorithm [10] for SPM. The procedure is adapted to consider the cost and utility, and a novel lower-bound on the cost is introduced to reduce the search space.

## 4.1 Lower Bound of Average Cost

In HUPM, several upper-bounds on the utility of itemsets have been defined to reduce the search space, and improve the performance of HUPM. In this study, the cost is neither monotonic nor anti-monotonic and thus cannot be directly used for search space pruning. Moreover, upper-bounds on the utility from previous work cannot be used in this work as the utility and cost are defined differently. Hence, a lower-bound on the cost is designed, named *Average Supported Cost* (ASC), and a corresponding pruning property.

*Definition 4.1.* (Average Supported Cost) For a pattern $p$, let $seq(p)$ be the set of sequences where $p$ appears. Furthermore, let $sc(p)$ be the costs of $p$ in these sequences. Assume that $sc(p)$ is sorted in ascending order. The *Average Supported Cost* (ASC) of $p$ is the first *minsup* cost values in $sc(p)$, divided by $sup(p)$, and is denoted as $asc(p)$.

Property 3. *(Underestimation) The ASC of a pattern $p$ is smaller than or equal to its true cost, that is $asc(p) \leq ac(p)$.*

Property 4. *(Monotonicity) The ASC measure is monotonic. Let $p_x$ and $p_y$ be two frequent patterns. If $p_x \subset p_y$, then $asc(p_x) \leq asc(p_y)$.*

Property 5. *(Pruning) For a pattern $p$, if $asc(p) > maxcost$, then the pattern $p$ is a high-cost pattern ($ac(p) > maxcost$) as well as all its extensions.*

The proofs are ommitted due to space limitation. For example, in Table 4, assume that the minimum support is 2, $asc(\{a\}) = (20+25)/3 = 15 < ac(\{a\}) \approx 28.3$. The $ASC$ of pattern $b$ that after $a$ is $asc(\{b\}) = (16+30)/3 \approx 15.3 < ac(\{b\}) \approx 28.7$. Furthermore, $asc(\{a,b\}) = \frac{(25+30,\{a,b\})+(40+16,\{a,b\})}{3} = 37 < ac(\{a,b\}) = 57$. In this paper, a key consideration is to find patterns having a low average cost. Using the proposed Property 5, if the $ASC$ of a pattern is larger than *maxcost*, this pattern and all its extensions can be eliminated.

## 4.2 The LCHUB Algorithm

The first proposed algorithm is named LCHUB (Low-Cost High Utility pattern mining in Binary SADB). It is designed to mine positive patterns in a binary SADB (see Section 3.1). The main procedure (Algorithm 1) takes as input a binary SADB, and the *minsup* and *maxcost* thresholds. The algorithm first scans the database to calculate the support, average cost and $ASC$ (Average Supported Cost) of each activity. For each activity $a_x$ such that $sup(a_x) \geq minsup$, if the average cost of $a_x$ is no greater than *maxcost*, the pattern $\{a_x\}$ is output. Then the pruning Property 5 is applied. If $asc(a_x) \leq maxcost$ the extensions of the pattern $a_x$ are explored recursively using a depth-first search by calling the Search procedure (Algorithm 2). This latter takes as input a database $D$, the thresholds and a pattern $p$. The procedure constructs the projected database of $D$ by the pattern $p$ as in the PrefixSpan algorithm [10]. Then, it scans this projected database to calculate the support, average cost and $ASC$ of each activity in $PD$. For each activity $a_y$ such that $sup(a_y) \geq minsup$, if the average cost of $a_y$ in $PD$ is no greater than *maxcost*, the pattern $p \cup a_y$ is output. Then, if the pruning Property 5 is passed, the Search procedure is called to explore extensions of $p \cup a_y$. When the algorithms terminates, all patterns have been found.

The LCHUB algorithm applies the novel pruning property based on the cost, and also prune patterns using the *minsup* threshold as done in traditional SPM. As it will be shown in the experiments, the novel pruning property based on the cost, can greatly reduce the runtime of the algorithm.

**input** : a SADB $D$, *minsup*, *maxcost*
**output** : the positive low-cost patterns

Scan $SADB$ to calculate the support, average cost and $ASC$ of each activity;
**foreach** *each activity* $a_x$ **do**
  **if** ($sup(a_x) \geq$ minsup) **then**
    **if** ($ac(a_x) \leq$ maxcost) **then**
      Calculate $tf(a_x)$ // Added for LCHUN
      Output($a_x$);
    **end**
    **if** ($asc(a_x) \leq$ maxcost **then**
      Search($D, minsup, maxcost, a_x$)
  **end**
**end**

**Algorithm 1:** The LCHUB/LCHUN algorithm

**input** : a SADB $D$, *minsup*, *maxcost*, a pattern $p$
**output** : the positive patterns having $p$ as prefix

Project the database $D$ using pattern $p$ to obtain a projected database $PD$. Scan $PD$ to calculate the support, average cost and ASC of each activity;
**foreach** *each activity* $a_y \in PD$ **do**
  **if** ($sup(a_y) \geq$ minsup) **then**
    **if** ($ac(p \cup a_y) \leq$ maxcost) **then**
      Calculate $tf(p \cup a_y)$ // Added for LCHUN
      Output($p \cup a_y$);
    **end**
    **if** $asc(p \cup a_y) \leq$ maxcost **then**
      Search( $PD, minsup, maxcost, p \cup a_y$);
    **end**
  **end**
**end**

**Algorithm 2:** Search procedure of LCHUB/LCHUN

## 4.3 The Correlated LCHUB Algorithm

The second proposed algorithm is named *corLCHUB* (correlated LCHUB), and is designed to solve Problem 3.2. The main procedure (Algorithm 3) takes as input a $SADB$ such as Table 1, and the *minsup* and *maxcost* thresholds. The algorithm first scans the database to calculate the support in positive sequences, average cost, and the ASC lower-bound of each activity $a$. If the $sup(a_x, D^+_{a_x})$ is equal to 0, then the patern $a_x$ and its extensions are ignored (the reason is that if $a_x$ only appears in the negative class, then all its extensions will also only appear in the negative class). Otherwise, if $sup(a_x)$ is larger than $minsup$ and $ac(a_x) \leq maxcost$, then $cor(a_x)$ is calculated and $a_x$ is output. If $asc(a_x) \leq$, then the Search procedure (Algorithm 4) is called to consider larger patterns having $a_x$ as prefix. The Search procedure takes as input a database $D$, the thresholds and a pattern $p$ to be extended. The Search procedure constructs the projection $PD$ of database ($D$) with $p$ and then scans $PD$. For each activity $a_y$ if $sup(p \cup a_y) \geq minsup$ and $ac(p \cup a_y) \leq maxcost$, then $cor(a_y)$ is calculated and saved as a pattern. And if the cost pruning condition is satisfied $asc(p \cup a_y) \leq maxcost$, the $Search$ procedure is recursively called to extend $p \cup a_y$.

The difference between the $corLCHUB$ algorithm and $LCHUB$ is that the *correlation* is calculated for each pattern to assess the effect of the pattern on the utility. This information is useful for the user to select the most efficient patterns.

**input** : a SADB $D$, $minsup$, $maxcost$
**output** : the correlated low-cost patterns (for binary SADB)

Scan SADB to calculate the support and average cost in positive and negative sequences, and the $ASC$ of each activity $a$;
**foreach** *activity* $a_x$ **do**
  **if** $sup(a_x, D^+_{a_x}) \neq 0$ **then**
    **if** ($sup(a_x) \geq$ minsup) **then**
      **if** ($ac(a_x) \leq$ maxcost) **then**
        **if** ($sup(a_x, D^+_{a_x}) = sup(a_x)$ **then** $cor(a_x)$:=1 **else** calculate $cor(a_x)$
        Output $a_x$
        **if** $asc(a_x) \leq$ maxcost **then**
        Search($D, minsup, maxcost, a_x$)
    **end**
  **end**
**end**

**Algorithm 3:** The corLCHUB Algorithm

## 4.4 The LCHUN Algorithm

Another algorithm named LCHUN (Low-Cost High Utility pattern mining in Numeric SADB) is designed for solving the problem of Section 3.3. Its main procedure (Algorithm 1) takes as input a $SADB$ such as Table 4, $minsup$ and $maxcost$. The algorithm first scans the database to calculate the support, average cost and $ASC$ of each activity $a_x$. If $sup(a_x)$ is no less than $minsup$ and $ac(a_x)$ is no greater than $maxcost$, the trade-off of $a_x$ is calculated and $a_x$ is output. If the ASC of $a_x$ is no greater than $maxcost$, the $Search$ procedure (Algorithm 4) is called with $p = a_x$ to

**input** : a SADB $D$, $minsup$, $maxcost$, a pattern $p$
**output** : the patterns having $p$ as prefix

Construct the projected database of $p$ as $PD$. Scan $PD$ to calculate the support $sup(a)$, $asc(a)$, $ac(a)$ and the support of $a$ in positive sequences;
**foreach** *activity* $a_y \in PD$ **do**
  **if** $sup(a_y, PD^+_{a_y}) \neq 0$ **then**
    **if** ($sup(a_y) \geq$ minsup) **then**
      **if** $ac(a_y) \leq$ maxcost) **then**
        **if** ($sup(a_y, PD^+_{a_y}) = sup(a_y)$) **then** $cor(p \cup a_y) := 1$ **else** calculate $cor(p \cup a_y)$ Output($p \cup a_y$)
        **if** $asc(p \cup a_y) \leq$ maxcost **then**
        Search($PD, minsup, maxcost, p \cup a_y$)
    **end**
  **end**
**end**

**Algorithm 4:** Search procedure of the corLCHUB

consider patterns that are extensions of $p$. This procedure constructs the projected database ($PD$) of $p$ and scans $PD$. For each activity $a_y$ , if $sup(p \cup a_y)$ is no less than $minsup$ and $ac(p \cup a_y)$ is no greater than $maxcost$, the trade-off of $a_y$ is calculated and $(p \cup a_y)$ is output. Then, if $asc(p \cup a_y)$ is no greater than $maxcost$, the $Search$ procedure is recursively called to extend $p \cup a_y$. When the algorithms terminates all patterns have been found. The patterns can be sorted by average utility and/or trade-off before being presented to the user. The main originality of using LCHUN is to use the trade-off measure to quantify the efficiency of patterns for numeric utility values.

# 5 EXPERIMENTS

In this section, we evaluate the performance of the three proposed algorithms: LCHUB, corLCHUB and LCHUN. Algorithms are implemented in Java and experiments were performed on a computer having a 64 bit Xeon E3-1270 3.6 Ghz CPU, running the Windows 10 operating system and 64 GB of RAM. A performance evaluation is first described and then a case study in e-learning.

## 5.1 Performance Evaluation

The algorithms were evaluated in terms of performance on three standard benchmark datasets used in sequential pattern mining, namely Bible, BMS and SIGN, which were obtained from the SPMF software website [4]. 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 on high utility sequential pattern mining [20]. The Bible dataset contains $36,369$ transactions with $13,905$ distinct items and an average transaction length of 44.3. The BMS dataset contains 59,601 transactions with 497 distinct items and an average transaction length of 6.02. The SIGN dataset

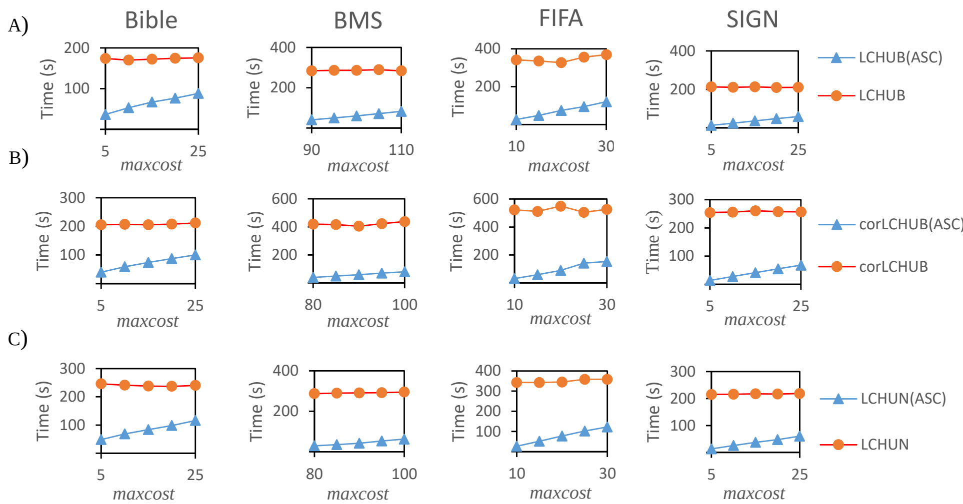


**Figure 1: Runtimes of the proposed algorithms when $maxcost$ is varied**

contains 730 transactions with 267 distinct items and an average transaction length of 104.1. Since this is the first paper on discovering low-cost high utility patterns, the performance of the proposed algorithms cannot be directly compared to other algorithms. Thus we compared each algorithm with a version where search space pruning using the cost was deactivated. Each algorithm was run on the three datasets while increasing the maxcost threshold. For each dataset, we recorded the execution time, number of candidate patterns and number of patterns found. The comparison of execution times for various $maxcost$ threshold values are shown in Fig. 1 for all datasets. It can be seen in Fig. 1 that algorithms are at least four times faster using the cost pruning strategy. Moreover, as the $maxcost$ threshold is increased, the cost pruning strategy becomes less effective (lines are closer on each chart) and the number of found patterns increases. This is reasonable since when $maxcost$ is increased, more and more patterns satisfy the contraints on the cost measure. In addition, as $minsup$ increases, the number of patterns and runtime decrease (as found in traditional SPM, and not shown due to space limitation).

## 5.2 Case study in E-learning

To evaluate whether interesting patterns can be found in real data, we applied the developed algorithms on e-learning data collected by Vahdat et. al [15]. This data consists of logs of the Deeds e-learning environment, used for learning digital electronics. The environment provides learning materials. Furthermore, learners are asked to solve problems with different levels of difficulty to assess their knowledge. The dataset indicates the 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 where a student took and exam, and a score is provided for the final exam.

*5.2.1 Application of Correlated LCHUB.* We first applied corLCHUB to find useful patterns that have a low cost in terms of time and that resulted in passing the final exam. More precisely, we wanted to find which sessions are the most important for passing the exam. The data was first preprocessed to obtain a sequence of sessions for each student, annotated with the time cost of activities. The utility of a sequence was defined as the final exam score, transformed as binary classes PASS/FAIL based on a 60% minimum passing threshold.

After data preparation, the correlated LCHUB algorithm was applied with $minsup = 0.5$ and $maxcost = 600$. Results provide some interesting insights such that pattern$\{1, 6\}$, $\{1, 2, 5, 6\}$, $\{2, 6\}$ and $\{1, 2, 6\}$ have a positive correlation with success (0.21, 0.209, 0.208, 0.204, respectively), where numbers represents session IDs. It was also found that some patterns such as $\{4, 5\}$ and $\{5\}$ have a negative correlation ($-0.109$ and $-0.147$, respectively). Moreover, it was found that some patterns such as $\{2, 3\}$, $\{3, 4, 5, 6\}$ are barely correlated with the final exam result, because their correlation are both 0.001. Overall, based on these patterns, it is found that students who learnt Session 1 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 likely to fail the exam. Besides, if a student spend time on the other unrelated sessions, it may increase time consumption but not increase much the chances of passing the exam. Another observation is that positive correlation is rather small in this dataset.The reasons are that the dataset is small (only 62 students took the exam), and that few students passed the exam given the 60 point threshold. If the threshold is reduced, the correlation values increase.

*5.2.2 Application of LCHUN.* In a second experiment, we applied the designed LCHUN algorithm to the same e-learning database but to analyse activities within sessions rather than analyzing the whole learning process. For a student, a session is a sequence of activities where each activity has a time duration (cost), and the score at the session exam is the utility of the sequence. The database thus contains multiple sequences for the same session (one for each student). Unlike the previous experiment, the score is here considered as a numeric utility rather than a binary utility. The goal of this experiment is to obtain insights about how to efficiently use learning materials to obtain high scores in each session.

For this experiment, parameters were set as $minsup = 0.1$ and $maxcost = 100$. For Session 6, the average score is 14. The most efficient pattern to obtain this score is (DeedsEs_6_2), which has a trade-off of 0.63. For Session 5, to obtain the average score of 6 the most efficient pattern is (Study_Es_5_2), having a trade-off of 1.35. For Session 4, the average score of 14 is obtained with the pattern (Study_Es_4_2) having a trade-off of 0.71. From these patterns, it is seen that to obtain an average score, students don't need to do all learning activities, and one activity is often enough. Although the above patterns have a small trade-off, they often also have a low utility. Thus, on overall we suggest considering not just the trade-off but also the utility to select efficient patterns. For example, for Session 6, to obtain a high score of 28(/40), the pattern with the smallest trade-off (1.35) is (Deeds_Es_6_1 )(Deeds_Es_6_2 )(Study_Es_6_3 )(Study_Es_6_3).

On overall, it is found in that experiment that efficient patterns can be found for various range of utility values, which may thus let a user select different patterns based on his goal in terms of utility. 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 the patterns and the exam questions, which indicates that the patterns found are reasonable.

## 6 CONCLUSION

In this paper, we proposed a novel problem of mining low cost-high utility patterns. We defined three versions of this problem, which correspond to three different real-life scenarios. Algorithms were proposed for each problem, which rely on a novel ASC lower-bound on the average cost of patterns to reduce the search space and discover patterns efficiently. We have performed an experimental study on three real-life datasets to evaluate the performance of the algorithm. Results show that the pruning strategy is effective. Moreover, a case study with e-learning data has shown that useful patterns can be found having a low cost and a high utility. Those patterns can provide insights to students and teachers about how to use learning material more efficiently.

For future work, we are interested in exploring other optimizations to improve the performance, including developping tighter lower-bounds on the cost of patterns. Moreover, we plan to integrate the concept of cost in other types of patterns such as itemsets and sequential rules, and add other constraints such as the length of patterns to the proposed model.

## REFERENCES


[1] Rakesh Agrawal and Ramakrishnan Srikant. 1995. Mining sequential patterns. In *Proc. of the 11th Intern. Conf. on Data Engineering*. IEEE, 3–14.

[2] Alejandro Bogarín, Rebeca Cerezo, and Cristóbal Romero. 2018. A survey on educational process mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8, 1 (2018).

[3] Benjamin Dalmas, Philippe Fournier-Viger, and Sylvie Norre. 2017. TWINCLE: A Constrained Sequential Rule Mining Algorithm for Event Logs. *Procedia Computer Science* 112 (2017), 205–214.

[4] Philippe Fournier-Viger, Antonio Gomariz, Ted Gueniche, Azadeh Soltani, Cheng-Wei Wu, and Vincent S Tseng. 2014. SPMF: a Java open-source pattern mining library. *The Journal of Machine Learning Research* 15, 1 (2014), 3389–3393.

[5] Javier Garcıa-Algarra. 2017. Subgroup Discovery in Process Mining. In *Proc. of the 20th Intern. Conf. on Business Information Systems*, Vol. 288. Springer, 237.

[6] Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. 2004. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data mining and knowledge discovery* 8, 1 (2004), 53–87.

[7] Ying Liu, Wei-keng Liao, and Alok Choudhary. 2005. A two-phase algorithm for fast discovery of high utility itemsets. In *Proc. of the 9th Pacific-Asia Conf. on Knowledge Discovery and Data Mining*. Springer, 689–695.

[8] Felix Mannhardt and Daan Blinde. 2017. Analyzing the trajectories of patients with sepsis using process mining. *RADAR+ EMISA* 1859, 72–80.

[9] Ronny S Mans, Wil MP van der Aalst, Rob JB Vanwersch, and Arnold J Moleman. 2013. Process mining in healthcare: Data challenges when answering frequently posed questions. In *Process Support and Knowledge Representation in Health Care*. Springer, 140–153.

[10] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. 2004. Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Transactions on knowledge and data engineering* 16, 11 (2004), 1424–1440.

[11] Leonard KM Poon, Siu-Cheung Kong, Michael YW Wong, and Thomas SH Yau. 2017. Mining sequential patterns of students' access on learning management system. In *Intern. Conf. on Data Mining and Big Data*. Springer, 191–198.

[12] Jayanthi Ranjan and Kamna Malik. 2007. Effective educational process: a data-mining approach. *Vine* 37, 4 (2007), 502–515.

[13] Ramakrishnan Srikant and Rakesh Agrawal. 1996. Mining sequential patterns: Generalizations and performance improvements. In *Proc. of the 5th Intern. Conf. on Extending Database Technology*. Springer, 1–17.

[14] Vincent S Tseng, Cheng-Wei Wu, Bai-En Shie, and Philip S Yu. 2010. UP-Growth: an efficient algorithm for high utility itemset mining. In *Proc. of the 16th ACM SIGKDD Intern. Conf. on Knowledge discovery and data mining*. ACM, 253–262.

[15] Mehrnoosh Vahdat, Luca Oneto, Davide Anguita, Mathias Funk, and Matthias Rauterberg. 2015. A learning analytics approach to correlate the academic achievements of students with interaction data from an educational simulator. In *Design for Teaching and Learning in a Networked World*. Springer, 352–366.

[16] Wil MP Van Der Aalst, Marcello La Rosa, and Flávia Maria Santoro. 2016. Business process management. (2016).

[17] Wil MP Van der Aalst and AJMM Weijters. 2004. Process mining: a research agenda. *Computers in Industry* 55, 3 (2004), 231–244.

[18] Junfu Yin, Zhigang Zheng, and Longbing Cao. 2012. USpan: an efficient algorithm for mining high utility sequential patterns. In *Proc. of the 18th ACM SIGKDD Intern. Conf. on Knowledge discovery and data mining*. ACM, 660–668.

[19] Mohammed J Zaki. 2001. SPADE: An efficient algorithm for mining frequent sequences. *Machine learning* 42, 1-2 (2001), 31–60.

[20] Souleymane Zida, Philippe Fournier-Viger, Cheng-Wei Wu, Jerry Chun-Wei Lin, and Vincent S Tseng. 2015. Efficient mining of high-utility sequential rules. In *Proc. of the 11th Intern. Conf. on Machine Learning and Data Mining in Pattern Recognition*. Springer, 157–171.