High Average-Utility Itemset Sampling Under Length Constraints

Lamine Diop(✉)

University of Tours, 3 Place Jean Jaurès, 41029 Blois, France
lamine.diop@univ-tours.fr

Abstract. High Utility Itemset extraction algorithms are methods for discovering knowledge in a database where the items are weighted. Their usefulness has been widely demonstrated in many real world applications. The traditional algorithms return the set of all patterns with a utility above a minimum utility threshold which is difficult to fix, while top-k algorithms tend to lack of diversity in the produced patterns. We propose an algorithm named HAISampler to sample itemsets where each itemset is drawn with a probability proportional to its average-utility in the database and under length constraints to avoid the long and rare itemsets with low weighted items. The originality of our method stems from the fact that it combines length constraints with qualitative and quantitative utilities. Experiments show that HAISampler extracts thousands of high average-utility patterns in a few seconds from different databases.

1 Introduction

High Utility Itemset mining (HUIM) [21] is an extension of the frequent pattern mining [1] which takes into account the quality and the quantity of an item in a transaction (the price for instance). Its usefulness has been widely demonstrated in many real life applications like user behavior analysis [17], marketing analysis [14], mobile commerce [16], stream web clicks [6] and interactive pattern mining [2]. Interactive pattern mining is a process that requires a short loop with rapid interaction between the system and the user [13]. Indeed, the instant discovery imposes a constraint on the response time of a few seconds to extract a representative set of patterns. Complete methods do not provide the relevant patterns in such a short time. Methods based on a condensed representation [20] or on top-k patterns [18] are also used to find the best patterns. Therefore, they often focus on the same part which contains slightly different patterns and then leads to a lack of diversity. The latter is crucial to present to the user a set of varied patterns at each iteration in order to improve his/her view and help the system to know his/her interest.

To solve this problem, we propose to benefit from the pattern sampling techniques [3,4]. It consists in providing a representative sample of patterns according to a distribution proportional to an interestingness measure chosen by the user while ensuring good diversity between the sampled patterns. Weighted utilities

J. Gama et al. (Eds.): PAKDD 2022, LNAI 13281, pp. 134–148, 2022.
https://doi.org/10.1007/978-3-031-05936-0_11

are proposed in [5,9], but independently of any transaction. In other words, these methods do not work in our case where the utility of an occurrence of a pattern depends on the transaction in which it appears and its length. To the best of our knowledge, this paper is the first to address the problem of high average-utility itemset sampling while integrating length constraints on the sampled patterns.

The rest of this paper is organized as follows: Sect. 2 presents a state-of-the-art on HUIM and pattern sampling methods. Section 3 gives basic notions and formalizes the problem. Section 4 presents HAISampler (High Average-utility Itemset Sampler) and Sect. 5 evaluates its accuracy and complexity. Finally, we present some experiments in Sect. 6 and conclude in Sect. 7.

2 Related Works

HUIM is one of the most difficult tasks to extract useful patterns in pattern mining area. Two of its main challenges are the control of the returned candidate patterns and the time cost of computing the utility of each pattern of the database. To solve these problems, many efficient methods are proposed [16,18,21]. Unfortunately, the efficiency of the exhaustive HUIM often depends on the size of the database on which they are applied. Nowadays, the used databases are very large and contain information as rich as their variety. The diversity of the information that a database contains is proportional to the cardinality of its pattern language. But, the more the number of pattern the more it is difficult to explore the corresponding database. Another problem encountered by HUIM methods is the long tail where patterns containing low weighted items have high utilities thanks to their length. In [15], the authors propose an average-based utility measure to avoid the long tail problem. However, this one favors itemsets of length 1, since they are not affected by the division, and therefore the returned patterns become obvious to the user. An alternative consists in setting a minimum frequency threshold in order to avoid the long patterns. However, it is very difficult for the user to set a minimum frequency threshold.

Pattern sampling [3] is a non-exhaustive method for discovering relevant patterns while offering strong statistical guarantees thanks to its randomness. Its usefulness has been widely demonstrated in many applications such as classification [4,7], anomaly detection [9,11] and instant discovery [10,13]. It has also been applied to many types of structured data like graphs [3], itemsets [4], numerical data [12] and sequences [8]. In [7] the authors weight each pattern with a norm-based utility (regardless of any sequence) to avoid the long tail problem. However, the output sampling is much more difficult in the case where the draw of a pattern is not proportional to its frequency in the database, and even more when length constraints are integrated.

In this paper, we propose an original method of output pattern sampling to address the high average-utility itemsets under length constraints. Contrary to the methods which are based on heuristic algorithms [19] to find the top-k high utility itemsets, the sampling method that we propose is exact. It draws an itemset proportionally to its average-utilities from the set of all patterns of the database that respect the length constraints.

3 Preliminaries and Problem Formulation

This section begins by presenting some basic notions and notations as well as the necessary definitions for the understanding of the subject. It ends with a formalization of the problem that we want to solve in this paper.

Let $\mathcal{I} = \{e_1, \cdots, e_N\}$ be a finite set of literals called items with an arbitrary total order $>_\mathcal{I}$ between items : $e_1 >_\mathcal{I} \cdots >_\mathcal{I} e_N$. An itemset or pattern, denoted by $\varphi = e_{i_1} \cdots e_{i_n}$, with $n \leq N$, is a none empty subset of $\mathcal{I}$, $\varphi \subseteq \mathcal{I}$. The pattern language corresponds to $\mathcal{L} = 2^{|\mathcal{I}|} \setminus \emptyset$ and the length of a pattern $\varphi \in \mathcal{L}$ denoted by $|\varphi|$, is the number of items it contains (its cardinality). A transactional database $\mathcal{D}$ corresponds to a set of couple (j, t) where $j \in \mathbb{N}$ is the unique identifier of a transaction and $t = e_1 \cdots e_n$ is an itemset of length $|t| = n$ defined in $\mathcal{I}$. We denote by $\mathcal{L}(\mathcal{D})$ the set of all patterns that appear in $\mathcal{D}$. In the rest of this paper, a transaction identified by j is denoted by t_j. In addition, for a transaction $t_j = e_{j_1} \cdots e_{j_n}$, we denote by $t_j^i = e_{j_{i+1}} \cdots e_{j_n}$ an itemset formed by discarding the i first items of t. So we have $|t_j^i| = |t_j| - i$. The i^{th} item of the transaction t_j is $t_j[i] = e_{j_i}$. In this paper, each item e_{j_i} of a transaction t_j has a weight, a strict positive real, which depends on this transaction, called its utility. For instance, in the case of a transaction which represents the set of all items purchased by a customer, the utility of an item e_i in the transaction t can be the product of its quantity $q(e_i, t)$ and its unit price $p(e_i)$. To be simpler on the rest of this paper, we associate each item e_i with its quantity in the transaction t that it appears, $e_i : q(e_i, t)$. Table 1 shows a database $\mathcal{D}$ with 5 transactions t_1, t_2, t_3, t_4 and t_5 defined on the set of items $\mathcal{I} = \{A, B, C, D, E, F\}$. We suppose that $A >_\mathcal{I} B >_\mathcal{I} C >_\mathcal{I} D >_\mathcal{I} E >_\mathcal{I} F$. In the database $\mathcal{D}$, the unit prices are: $p(A) = 25$, $p(B) = 30$, $p(C) = 10$, $p(D) = 5$, $p(E) = 15$ and $p(F) = 10$. With the transaction t_1, we have the following quantities $q(A, t_1) = 2$, $q(B, t_1) = 3$ and $q(C, t_1) = 2$.

Table 1. Example of database $\mathcal{D}$ with utilities on items

$\mathcal{D}$						
t_1	A:2	B:3	C:2			
t_2		B:2		D:4		
t_3	A:1		C:1	D:1		
t_4	A:3	B:1				
t_5	A:1	B:2		D:1	E:1	F:1

Items	Price
A	25
B	30
C	10
D	5
E	15
F	10

This toy database will be used in the rest of this paper to give illustrations. Since items in the transaction t_1 have not the same weight, then the occurrences of patterns in t_1 may not have the same utility in t_1.

Definition 1 (Occurrence of a pattern). *Let φ be a pattern defined on a language $\mathcal{L}$ of a database $\mathcal{D}$. If it exists a transaction t_j of $\mathcal{D}$ such that $\varphi \subseteq t_j$, then φ_j is an occurrence of the pattern φ in the transaction t_j. The utility of the pattern φ in the transaction t_j, denoted by $\mathtt{uOcc}(\varphi, t_j)$, is equal to 0 if $\varphi \not\subseteq t_j$ or $\varphi = \emptyset$, else $\mathtt{uOcc}(\varphi, t_j) = \sum_{e \in \varphi} (q(e, t_j) \times p(e))$.*

There are also utilities that are independent of any database such as length-based utilities [9]. In the following, we consider the length-based utility defined by $\mathtt{uLen}_{[m..M]}(\varphi) = 1/|\varphi|$ if $|\varphi| \in [m..M]$ and 0 otherwise, with m and M two positive integers. Thus, a pattern whose length is larger than M or smaller than m will be deemed useless.

Definition 2 (Average-Utility of a pattern under length constraints). *Let $\mathcal{D}$ be a database, $\mathcal{L}$ its language, m and M two integers such that $m \leq M$. The average-utility of the pattern $\varphi \in \mathcal{L}$ in $\mathcal{D}$ under minimum m and maximum M length constraints, denoted by $u^{avg}_{[m..M]}(\varphi, \mathcal{D})$, is the product of the sum of utilities of its occurrences and its length-based utility. Formally, $u^{avg}_{[m..M]}(\varphi, \mathcal{D}) = (\sum_{(j,t) \in \mathcal{D} \wedge \varphi \subseteq t} \mathtt{uOcc}(\varphi, t)) \times \mathtt{uLen}_{[m..M]}(\varphi)$.*

It is important to note that $u^{avg}_{[m..M]}$ is not a length-based utility.

Example 1. Let's consider the database $\mathcal{D}$ in Table 1 and the length constraints $m = 1$ and $M = 2$. We note that the pattern AC belongs in t_1 and t_3 only. So, AC has only two occurrences in $\mathcal{D}$: AC_1 et AC_3. We also have $\mathtt{uLen}_{[m..M]}(AC) = 1$ because $|AC| = 2 \in [1..2]$. So, $u^{avg}_{[1..2]}(AC, \mathcal{D}) = (\mathtt{uOcc}(AC, t_1) + \mathtt{uOcc}(AC, t_3))/2 = ((2 \times 25 + 2 \times 10) + (1 \times 25 + 1 \times 10)) \times 1/2 = (70 + 35)/2 = 52.5$. By the same way, we have $u^{avg}_{[1..2]}(ABEF, \mathcal{D}) = 110 \times 0 = 0$. Indeed, $|ABEF| = 4 \notin [1..2]$.

In this paper, we want to solve the problem formulated as follows:
Given $\mathcal{D}$ a transactional database with weighted items (quantity and/or quality), two positive integers m and M such that $m \leq M$, our main goal is to draw a pattern φ from the language $\mathcal{L}$ with a probability exactly equal to:

$$\mathbb{P}(\varphi, \mathcal{D}) = \frac{u^{avg}_{[m..M]}(\varphi, \mathcal{D})}{\sum_{\varphi' \in \mathcal{L}(\mathcal{D})} u^{avg}_{[m..M]}(\varphi', \mathcal{D})}.$$

The notations of this paper are summarized in Table 2.

4 Two-Phase Sampling of High Average-Utility Itemsets

In this section, we will first present the basics of our method (detailing the weighting phase and the drawing phase of a pattern) before presenting the HAISampler algorithm that we propose to sample high average-utility itemsets under length constraints.

4.1 Basics of Our Sampling Method

The high average-utility itemset sampling method that we propose in this paper uses a position-based length weighting system to weight each transaction. It is a system which consists in weighting each item of a given transaction according to the position it occupies there. The item weights are then used to draw a pattern using a conditional probability.[1]

Table 2. Notations

Symbol	Definition
$\mathtt{uOcc}(\varphi, t)$	Utility of the pattern φ in the transaction t
$\mathtt{uLen}_{[m..M]}(\varphi)$	Length-based utility of φ equal to $1/\lvert\varphi\rvert$ if $\lvert\varphi\rvert \in [m..M]$ and 0 otherwise
$u^{avg}_{[m..M]}(\varphi, \mathcal{D})$	Average-utility of the pattern φ in $\mathcal{D}$. It is equal to 0 if $\lvert\varphi\rvert \notin [m..M]$
t^i	Itemset formed by discarding the i first items of t, $t^i = t[i+1]\cdots t[n]$
$\omega^+_\ell(t[i], t)$	Sum of occurrences' utilities of length ℓ in t^{i-1} with item $t[i]$
$\omega^-_\ell(t[i], t)$	Sum of occurrences' utilities of length ℓ in t^i (without item $t[i]$)
$\omega^{avgU}_{[m..M]}(t)$	Sum of average-utilities of all occurrences in t
$\mathbb{P}^t_\ell(t[i]\lvert\varphi, \ell')$	Probability to draw item $t[i]$ in the transaction t after drawing $\ell - \ell'$ items and storing them in φ

Transaction Weighting: Let t be a transaction of length n defined on a set of items $\mathcal{I}$ endowed with a total order relation $>_\mathcal{I}$, m and M maximum and minimum length constraints respectively. The i^{th} item of the transaction t, $t[i]$, is associated with two lists of values $\omega^+_\ell(t[i], t)$ and $\omega^-_\ell(t[i], t)$, for $\ell \in [m..M]$.

Definition 3. *The weight $\omega^+_\ell(t[i], t)$ is the sum of utilities of the occurrences of length $\ell - 1$ in the transaction $t^i = t[i+1]\cdots t[n]$ to which we add the item $t[i]$, and the weight $\omega^-_\ell(t[i], t)$ is that of occurrences of length ℓ in t^i.*

$$\omega^+_\ell(t[i], t) = \sum_{\varphi \subseteq t^i \wedge |\varphi| = \ell - 1} \mathtt{uOcc}(\{t[i]\} \cup \varphi, t) \quad \text{and} \quad \omega^-_\ell(t[i], t) = \sum_{\varphi \subseteq t^i \wedge |\varphi| = \ell} \mathtt{uOcc}(\varphi, t).$$

Property 1 gives a formalization of the weights based on Definition 3.

Property 1 (Item weights $\omega^\bullet_\ell(t[i], t)$). The weights $\omega^+_\ell(t[i], t)$ and $\omega^-_\ell(t[i], t)$ of the item $t[i]$, for all $\ell \in [m..M]$, may be formally written as follows:[2] $\omega^+_\ell(t[i], t) = \omega^+_1(t[i], t) \times \binom{|t^i|}{\ell-1} + \sum_{\star \in \{+,-\}} \omega^\star_{\ell-1}(t[i+1], t)$ and $\omega^-_\ell(t[i], t) = \sum_{\star \in \{+,-\}} \omega^\star_\ell(t[i+1], t)$, with $\omega^+_1(t[i], t) = \mathtt{uOcc}(t[i], t)$ for all $i \in [1..|t|]$ and $\omega^\star_\ell(t[i], t) = 0$ for all $i > |t|$.

Indeed, the weights of an item $t[i]$ are deduced from those of $t[i+1]$. Using Property 1, we can easily compute the weight of a transaction under length constraints. By definition, the average-utility of an occurrence $\varphi \subseteq t$ is $\mathtt{uOcc}(\varphi, t)/|\varphi|$.

[1] Proof of theoretical results are available in Sect. A.

[2] By convention $\binom{n}{k} = 0$ if k>n and 1 if k = 0.

Property 2 (Transaction weight). The weight of a transaction t under minimum m and maximum M length constraints, denoted by $\omega^{avgU}_{[m..M]}(t)$, is the sum of average-utilities of the occurrences that it contains. Formally,

$$\omega^{avgU}_{[m..M]}(t) = \sum_{\ell=m}^{M} \left(\frac{1}{\ell} \sum_{i=1}^{|t|} \omega^{+}_{\ell}(t[i], t) \right) = \sum_{\ell=m}^{M} \frac{1}{\ell} \left(\omega^{+}_{\ell}(t[1], t) + \omega^{-}_{\ell}(t[1], t) \right).$$

Example 2. Let's consider transaction $t_1 = \{A{:}2, B{:}3, C{:}2\}$. The prices of its items are $p(A) = 25$, $p(B) = 30$ and $p(C) = 10$. Following the total order relation $>_{\mathcal{I}}$, the occurrences that start with A are : $\{A, AB, AC, ABC\}$, those who start with B are : $\{B, BC\}$, and finally only a pattern begins with $C : \{C\}$. The sum of the utilities of the occurrences of length $\ell \in [1..3]$ that start with $t_1[i]$, $i \in [1..3]$, in the transaction t_1 is: $\omega^{+}_{1}(A, t_1) = \texttt{uOcc}(A, t_1) = 2 \times 25 = 50$. Using Property 2 we have: $\omega^{+}_{2}(A, t_1) = \texttt{uOcc}(AB, t_1) + \texttt{uOcc}(AC, t_1) = (50 + 90) + (50 + 20) = 210$. In an identical way, we have the following weights for the transaction t_1:

$$t_1 : \{ \begin{array}{c|ccc|ccc|ccc} & \multicolumn{3}{c|}{\text{A}} & \multicolumn{3}{c|}{\text{B}} & \multicolumn{3}{c}{\text{C}} \\ \omega^{+}_{\ell} & 50 & 210 & 160 & 90 & 110 & 0 & 20 & 0 & 0 \\ \omega^{-}_{\ell} & 110 & 110 & 0 & 20 & 0 & 0 & 0 & 0 & 0 \end{array} \}$$

From this weighting, we deduce the weight of the transaction t_1 under the minimum $m = 1$ and maximum $M = 3$ length constraints which is equal to: $\omega^{avgU}_{[1..3]}(t_1) = (50 + 110)/1 + (210 + 110)/2 + (160 + 0)/3 = 373.33$.

We are going to show how to draw an occurrence from our weighting system.

Drawing an Itemset from a Transaction: Drawing a pattern from a position-based weighted transaction can be done using conditional probability. Lemma 1 gives an idea on the computation of the probability of drawing a given item knowing that we have already drawn (or not) higher items according to the order relation $>_{\mathcal{I}}$ introduced in Sect. 3.

Lemma 1. *Let ℓ be the length of the itemset to output and $\mathbb{P}^{t}_{\ell}(t[i]|\varphi, \ell')$ the probability to draw the item $t[i]$ in the transaction t after drawing $\ell - \ell'$ items and storing them in φ, with $e >_{\mathcal{I}} t[i]$ for all $e \in \varphi$. The probability to draw $t[i]$ knowing φ and ℓ' can be formulated as follows:*

$$\mathbb{P}^{t}_{\ell}(t[i]|\varphi, \ell') = \frac{\sum_{\varphi' \subseteq t^{i} \wedge |\varphi'| = \ell' - 1} \texttt{uOcc}(\varphi \cup \{t[i]\} \cup \varphi', t)}{\sum_{\varphi' \subseteq t^{i-1} \wedge |\varphi'| = \ell'} \texttt{uOcc}(\varphi \cup \varphi', t)}.$$

Property 3. The probability to draw the item $t[i]$ in the transaction t knowing the itemset φ and the length ℓ', with $|\varphi| = \ell - \ell'$, denoted by $\mathbb{P}^{t}_{\ell}(t[i]|\varphi, \ell')$, is given by the following formula:

$$\mathbb{P}^{t}_{\ell}(t[i]|\varphi, \ell') = \frac{\left(\sum_{k<i \wedge t[k] \in \varphi} \omega_1(t[k], t) \right) \times \binom{|t^{i}|}{\ell' - 1} + \omega^{+}_{\ell'}(t[i], t)}{\left(\sum_{k<i \wedge t[k] \in \varphi} \omega_1(t[k], t) \right) \times \binom{|t^{i-1}|}{\ell'} + \left(\sum_{\star \in \{+,-\}} \omega^{\star}_{\ell'}(t[i], t) \right)}.$$

The probability that the item $t[i]$ is not drawn knowing φ and ℓ' is $1 - \mathbb{P}^{t}_{\ell}(t[i]|\varphi, \ell')$.

The proofs of these two formulas follow from the fact that the probability of drawing $t[i]$ depends on the utilities of the items already drawn and those of the items which follow it to form a pattern of length ℓ.

Example 3. Suppose we need to draw a pattern of length $\ell = 2$ from the transaction t_1. The probability to draw $\varphi = AC$ is computed as follow: $\mathbb{P}^{t_1}(AC|\ell) = \mathbb{P}^{t_1}_{\ell}(A|\varphi = \emptyset, \ell' = 2) \times (1 - \mathbb{P}^{t_1}_{\ell}(B|\varphi = A, \ell' = 1)) \times \mathbb{P}^{t_1}_{\ell}(C|\varphi = A, \ell' = 1)$. Which gives us $\mathbb{P}^{t_1}(AC|\ell) = \frac{0+210}{0+320} \times (1 - \frac{50\times\binom{1}{1-1}+90}{50\times\binom{2}{1}+90+20}) \times \frac{50\times\binom{0}{1-1}+20}{50\times\binom{1}{1}+20} = \frac{210}{320} \times (1 - \frac{140}{210}) \times \frac{70}{70} = \frac{210}{320} \times \frac{70}{210} \times \frac{70}{70} = \frac{70}{320}$.

We are now going to formalize and present our two-phase approach for drawing patterns from a transactional database whose items are weighted.

4.2 HAISampler: High Average-utility Itemset Sampler Algorithm

As we described it in Sect. 4.1, our approach is done in two phases: preprocessing and drawing. The phase of drawing an itemset is divided into several steps: drawing a transaction t, drawing a length ℓ and finally, drawing an itemset of length ℓ based on the conditional probability. This phase is repeated K times to draw K patterns.

Algorithm 1. HAISampler (High Average-utility Itemset Sampler)

Input: A transactional database $\mathcal{D}$ having weighted items with a total order relation $>_{\mathcal{I}}$ and minimum m and maximum M length constraints
Output: φ a pattern drawn proportionally to its average-utility: $\varphi \sim u^{avg}_{[m..M]}(\mathcal{L}, \mathcal{D})$
//Phase 1: Preprocessing
Compute the weight of each transaction t in $\mathcal{D}$: $\omega^{avgU}_{[m..M]}(t)$
//Phase 2: Drawing
Draw a transaction proportionally to its weight: $t \sim \omega^{avgU}_{[m..M]}(\mathcal{D})$
Draw a length ℓ according to its weight $\sum_{\star\in\{+,-\}} \omega^{\star}_{[\ell..\ell]}(t[1], t)$: $\ell \sim \omega^{avgU}_{[m..M]}(t)$
$\varphi \leftarrow \emptyset$ ▷ Empty initialization of the pattern to return
$y \leftarrow 0$
$i \leftarrow 1$
while $\ell > 0$ **do**
  $z \leftarrow y \times \binom{|t^i|}{\ell} + \omega^{+}_{\ell}(t[i], t) + \omega^{-}_{\ell}(t[i], t)$
  $x \leftarrow random() \times z$ ▷ Randomly draw a real number between 0 and z
  if $x \leq y \times \binom{|t^i|}{\ell-1} + \omega^{+}_{\ell}(t[i], t)$ **then**
    $\varphi \leftarrow \varphi \cup \{t[i]\}$
    $y \leftarrow y + \omega^{+}_{1}(t[i], t)$
    $\ell \leftarrow \ell - 1$
  $i \leftarrow i + 1$
return φ ▷ A pattern drawn proportionally to its average-utility in $\mathcal{D}$

Algorithm 1 takes as input a transactional database defined over a set of items with a total order relation $>_{\mathcal{I}}$ and minimum m and maximum M length

constraints. We start with a preprocessing phase which computes the weight of each transaction (line 1) using Property 1 and Property 2. To draw a pattern, we first draw a transaction t proportionally to its weight $\omega^{avgU}_{[m..M]}(t)$ (line 2). Second, we draw a length ℓ proportionally to the sum of average-utilities of the occurrences of length ℓ that appear in the transaction t previously drawn (line 3). Lines 4 to 14 allow us to randomly draw an occurrence of length ℓ with a probability proportional to its utility in t. In line 8, we compute the total sum, z, of the utilities of the itemsets that start with $\varphi \cup \{t[i]\}$ following the order relation $>_{\mathcal{I}}$ in the transaction t. Then, we randomly draw a real number between 0 and z (line 9). If the drawn number is smaller than the sum of utilities of the ordered items, which starting with φ also contain the item $t[i]$ of transaction t, then we add $t[i]$ in the set of items to output (lines 10 and 11). In that case, the sum of utilities of the drawn items is updated in the variable y (line 12) and the number of remaining items decrements (line 13). When $\ell = 0$, we return on line 15 an itemset φ drawn proportionally to its average-utility in the database $\mathcal{D}$.

5 Theoretical Analysis of the Method

This section shows in Property 4 that HAISampler performs an exact draw of a pattern and gives finally its time complexity.

Property 4 (Soundness). Let $\mathcal{D}$ be a transactional database having utilities on items with a total order relation $>_{\mathcal{I}}$, and m and M two integers such that $m \leq M$. HAISampler randomly draws a pattern φ from the language $\mathcal{L}(\mathcal{D})$ with a probability equal to $u^{avg}_{[m..M]}(\varphi, \mathcal{D})/Z$ where $Z = \sum_{\varphi' \in \mathcal{L}(\mathcal{D})} u^{avg}_{[m..M]}(\varphi', \mathcal{D})$.

The complexity of our method, can be split into two parts: the complexity of preprocessing and that of drawing a pattern. It is important to note that the combination values are computed incrementally and stored in memory.

Preprocessing: To weight a transaction our method, HAISampler, spends a time of $O(|\mathcal{I}| \times (M - m) \times 2)$. Consequently, it weights all the transactions of the database in a complexity of $O(|\mathcal{D}| \times |\mathcal{I}| \times (M - m))$.

Drawing a Pattern: HAISampler starts by drawing a transaction with a complexity in $O(\log(|\mathcal{D}|))$. After, it draws a pattern proportionally to its utility in $O(|\mathcal{I}|)$. So, the complexity of drawing a pattern is $O(\log(|\mathcal{D}|) + |\mathcal{I}|)$. The draw of K patterns is then done in $O(K \times (\log(|\mathcal{D}|) + |\mathcal{I}|))$, hence equal to that of [4].

6 Experiments

In this experimental section, we study the efficiency of our method and present the dispersion of the average-utilities of the sampled patterns according to their length. Finally, we give some memory storage cost of HAISampler. The experiments[3] were carried out on 6 datasets including 3 from the UCI: `Adult`, `Chess`

[3] HAISampler (Python 3.8) https://github.com/HAISampler/haisampler-src.

and Mushroom with preprocessed versions[4], and 3 real datasets from SPMF[5]: BMS, Foodmart and Retail. Table 3 details the characteristics of the benchmarks. The value of the minimum length constraint is fixed at $m = 1$ throughout the experiments. All the experiments were performed on a 2.11 GHz 2 Core CPU PC with 32 GB memory.

Table 3. Characteristics of the benchmarks: the number of transactions, the number of distinct items, the minimum, the maximum and the average length of the transactions, the minimum, maximum and average weight of the items

$\mathcal{D}$	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{I}\rvert$	$\lvert t\rvert_{\min}$	$\lvert t\rvert_{\max}$	$\lvert t\rvert_{avg}$	$p(e,t)_{\min}$	$p(e,t)_{\max}$	$p(e,t)_{avg}$
Adult	48,842	97	12	15	14.87	1.0	99.0	50.04
BMS	59,602	497	1	267	2.51	7.0	9,000.0	724.79
Chess	28,056	58	7	7	7.00	1.0	99.0	50.05
Foodmart	4,141	1,559	1	14	4.42	50.0	2,166.0	655.66
Mushroom	8,124	90	22	23	22.69	1.0	99.0	50.02
Retail	88,162	16470	1	76	10.31	1.0	140.0	16.41

Speed of the Method. The average execution times that we are going to present were obtained by repeating the program 100 times for each case. The standard deviations obtained are very low (mostly equal to 0 in the drawing phase), that is why we have omitted them.

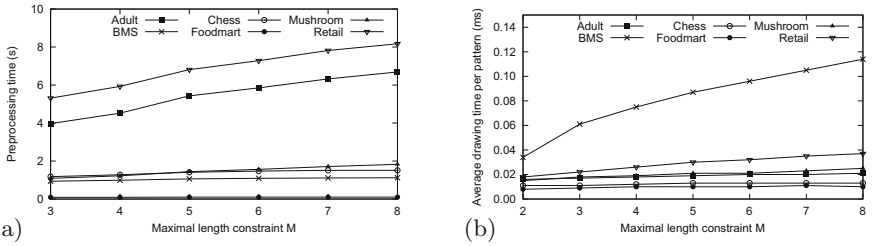

Fig. 1. Preprocessing time (a) and the drawing time of a pattern (b) according to M

Preprocessing Time: Figure 1-(a) shows the preprocessing time according to the maximal length constraint $M \in [3..8]$ of the 6 datasets in Table 3. First, it shows that the preprocessing time varies according to the maximum length constraint. However, it remains less than 9 s in all our datasets with a maximum length constraint $M = 8$, which is already too high if we want to avoid the long tail phenomenon. It is also important to note that the time to preprocess the datasets by HAISAMPLER increases with the size of the database and the average length

[4] Each item was associated with a utility taken randomly between 1 and 100.
[5] http://www.philippe-fournier-viger.com/spmf.

of transactions. Finally, we can say that the method HAISampler consumes low time for preprocessing datasets.

Drawing Time per Pattern: Figure 1-(b) shows the evolution of the drawing time per pattern according to the maximum length constraint $M \in [2..8]$. First, we note that the drawing times change slightly depending on the maximum length constraint. Then, the curves show that the drawing time depends on the size of the database and the maximum length of transactions. Indeed, longest transactions consume a lot of time especially when the maximum length constraint is high. Finally, the time to draw a pattern remains less than 0.15 ms on all datasets we use here. It is less than 0.04 ms when $M \leq 8$ except in `BMS`. This means that HAISampler manages to draw thousands of patterns in a few seconds.

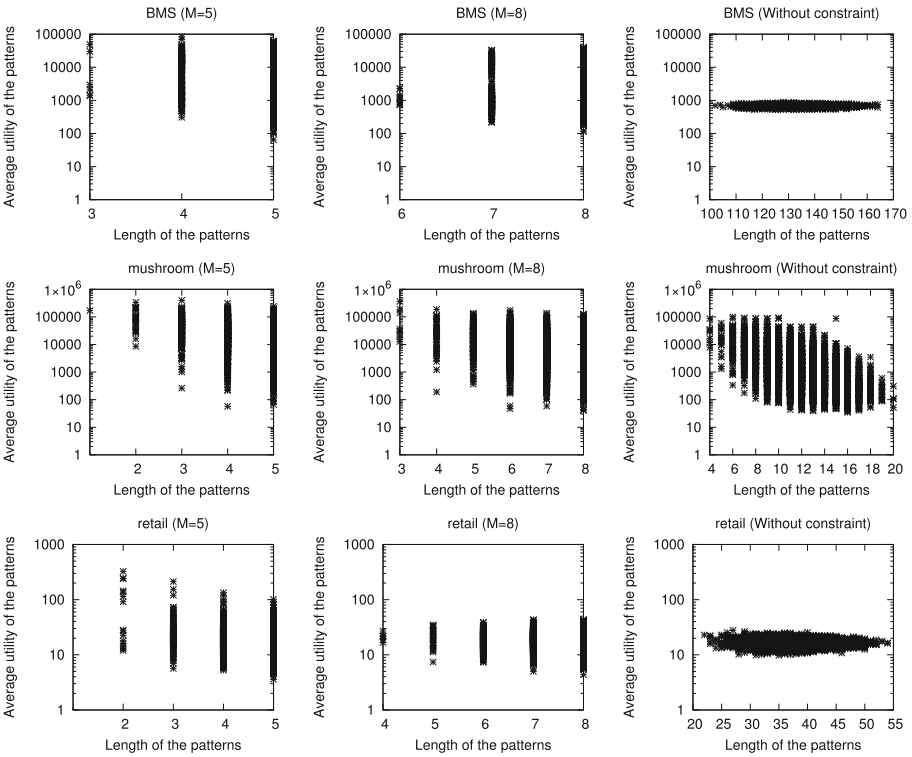

Fig. 2. Utility distribution of 10,000 sampled patterns

Impact of Length Constraints on the Sampled Patterns. We will show how the maximum length constraint can impact the utility of the sampled patterns from databases reaching the long tail curse. To do this, we have chosen two real datasets `BMS` and `Retail`, and one synthetic dataset `Mushroom`. The maximum length constraint M is tested with 5, 8 and ∞ (without constrained).

Figure 2 shows the distribution of the average-utilities of the patterns according to their length for the three chosen datasets (each dot represents a sampled pattern). `BMS` and `Retail` clearly show that if the maximum length constraint is very high or even unused, the drawn patterns are very long and formed by low weighted items (this is the long tail phenomenon). Besides, none of the returned patterns by the unconstrained method has a length smaller than 100 for `BMS`, 20 for `Retail`, and 4 for `Mushroom`. The latter is less impacted by the fact that all of its transactions have almost the same length. So we can say that it is very interesting to use length constraints to sample high utility patterns from a database suffering from the long tail curse, which is often the case with real data.

Memory Storage Cost. Someone may wonder about the memory storage cost of our method since it adds information on the items of each transaction and keeps the combination values $\binom{n}{k}$ in memory. Table 4 shows some statistics computed with the "asizeof"[6] package for the different datasets used in this paper.

Table 4. Memory storage cost in Mega Byte (MB) of HAISampler with $M \in \{5, 7, 8\}$

Maximal length M	$\mathcal{D}$					
	`Adult`	`BMS`	`Chess`	`Foodmart`	`Mushroom`	`Retail`
5	434.629	82.681	109.411	10.051	113.593	542.965
7	483.499	85.658	113.906	10.256	128.513	595.098
8	504.021	86.822	113.906	10.271	135.323	616.793

As expected, the memory storage cost increases slightly depending on the maximum length constraint M, and it remains less than 1 GB with $M = 8$ (maximum 616.793 MB with `Retail`). This means that the weighting approach of HAISampler is not expensive in storage. So it can be used with larger datasets to sample high average-utility itemsets.

7 Conclusion

This paper presents the first method for sampling high average-utility itemsets under length constraints. We have shown that HAISampler is exact and efficient at drawing thousands of patterns in a few seconds on real and synthetic datasets with reasonable preprocessing time. The experiments carried out show the value of length constraints for sampling patterns that have good utilities.

[6] https://code.activestate.com/recipes/546530-size-of-python-objects-revised/.

Moreover, we can easily adapt our approach to situations where the utility of an occurrence of a pattern in a transaction is the product of the utilities of its items. In perspective, we would like to extend our approach to other complex data structures such as sequences [7] and graphs [3]. In the short term, we intend to show how our position-based length weighting system can be extended for sampling high average-utility itemsets on data streams [6].

A Appendix (Proof of Theoretical Results)

Proof **(Property 1).** Let's start by showing that $\omega_\ell^-(t[i],t) = \sum_{\star\in\{+,-\}} \omega_\ell^\star(t[i+1],t)$. By definition, $\omega_\ell^-(t[i],t)$ is the sum of the utilities of the set of patterns of length ℓ in t^i, $\omega_\ell^-(t[i],t) = \sum_{\varphi\subseteq t^i\wedge|\varphi|=\ell} \mathtt{uOcc}(\varphi,t)$. This set can be split into two parts: the one that contains the patterns starting with the item $t[i+1]$ whose sum of their utilities is equal to $\omega_\ell^+(t[i+1],t)$ by definition, and the one that contains the patterns not starting with $t[i+1]$ and whose sum of their utilities is equal to $\omega_\ell^-(t[i+1],t)$. It implies that

$$\sum_{\varphi\subseteq t^i\wedge|\varphi|=\ell} \mathtt{uOcc}(\varphi,t) = \omega_\ell^+(t[i+1],t) + \omega_\ell^-(t[i+1],t) = \sum_{\star\in\{+,-\}} \omega_\ell^\star(t[i+1],t). \tag{1}$$

Let's show that $\omega_\ell^+(t[i],t) = \omega_1^+(t[i],t) \times \binom{|t^i|}{\ell-1} + \sum_{\star\in\{+,-\}} \omega_{\ell-1}^\star(t[i+1],t)$. We know by definition that $\omega_\ell^+(t[i],t)$ is the sum of utilities of itemsets of length ℓ in t^{i-1} which start with $t[i]$ following the total order relation $>_\mathcal{I}$. Formally, we have: $\omega_\ell^+(t[i],t) = \sum_{\varphi\subseteq t^i\wedge|\varphi|=\ell-1} \mathtt{uOcc}(\{t[i]\}\cup\varphi,t)$. But $\mathtt{uOcc}(\{t[i]\}\cup\varphi,t) = \mathtt{uOcc}(\{t[i]\},t) + \mathtt{uOcc}(\varphi,t)$ by definition. Then, $\omega_\ell^+(t[i],t) = \sum_{\varphi\subseteq t^i\wedge|\varphi|=\ell-1} (\mathtt{uOcc}(\{t[i]\},t) + \mathtt{uOcc}(\varphi,t))$. This implies: $\omega_\ell^+(t[i],t) = \sum_{\varphi\subseteq t^i\wedge|\varphi|=\ell-1} \mathtt{uOcc}(\{t[i]\},t) + \sum_{\varphi\subseteq t^i\wedge|\varphi|=\ell-1} \mathtt{uOcc}(\varphi,t)$. However, we know on the one hand that $\sum_{\varphi\subseteq t^i\wedge|\varphi|=\ell-1} \mathtt{uOcc}(\{t[i]\},t) = \mathtt{uOcc}(\{t[i]\},t) \times \binom{|t^i|}{\ell-1}$ and $\mathtt{uOcc}(\{t[i]\},t) = \omega_1^+(t[i],t)$ by definition, so $\sum_{\varphi\subseteq t^i\wedge|\varphi|=\ell-1} \mathtt{uOcc}(\{t[i]\},t) = \omega_1^+(t[i],t) \times \binom{|t^i|}{\ell-1}$. On the other hand, $\sum_{\varphi\subseteq t^i\wedge|\varphi|=\ell-1} \mathtt{uOcc}(\varphi,t)$ is the sum of utilities of the set of patterns of length $\ell-1$ in the transaction t^i. From (1), we can also say that $\sum_{\varphi\subseteq t^i\wedge|\varphi|=\ell-1} \mathtt{uOcc}(\varphi,t) = \sum_{\star\in\{+,-\}} \omega_{\ell-1}^\star(t[i+1],t)$. Then we have: $\omega_\ell^+(t[i],t) = \omega_1(t[i],t) \times \binom{|t^i|}{\ell-1} + \sum_{\star\in\{+,-\}} \omega_{\ell-1}^\star(t[i+1],t)$. Hence the result. □

Proof **(Property 2).** By definition, the weight of the transaction t is the sum of the average-utilities of the pattern occurrences it contains. According to Property 1, the weight of the transaction t under the minimum m and maximum M length constraints is nothing more than the sum of the average-utilities of pattern occurrences that start with the item $t[1]$ and respect the imposed length constraints, $\sum_{\ell=m}^{M}(\frac{1}{\ell} \times \omega_\ell^+(t[1],t))$, and that of the patterns that do not start with the item $t[1]$ but respect the length constraints, $\sum_{\ell=m}^{M}(\frac{1}{\ell} \times \omega_\ell^-(t[1],t))$. However, we know that $\sum_{\ell=m}^{M}(\frac{1}{\ell} \times \omega_\ell^+(t[1],t)) + \sum_{\ell=m}^{M}(\frac{1}{\ell} \times \omega_\ell^-(t[1],t)) = \sum_{\ell=m}^{M} \frac{1}{\ell} \times \left(\omega_\ell^+(t[1],t) + \omega_\ell^-(t[1],t)\right)$. Hence the result. □

Proof **(Lemma 1).** By definition, the probability to draw the item $t[i]$ of the transaction t after having drawing from it $\ell - \ell'$ items and store them in φ is nothing more than the probability of drawing a pattern that begins with $\varphi \cup \{t[i]\}$, according to the order relation $>_{\mathcal{I}}$, among the set of patterns that start with φ. On the one hand, we know that the set of patterns of length ℓ that start with $\varphi \cup t[i]$ is defined by $\{\varphi'' \subseteq t : (\varphi'' = \varphi \cup \{t[i]\} \cup \varphi')(\varphi' \subseteq t^i)(|\varphi'| = \ell' - 1)\}$. The sum of utilities of the patterns of this set is equal to $\sum_{\varphi' \subseteq t^i \wedge |\varphi'| = \ell' - 1} \mathtt{uOcc}(\varphi \cup \{t[i]\} \cup \varphi', t)$. On the other hand, we know that the set of patterns of length ℓ that start with φ is defined by $\{\varphi'' \subseteq t : (\varphi'' = \varphi \cup \varphi')(\varphi' \subseteq t^{i-1})(|\varphi'| = \ell')\}$. The sum of utilities of the patterns of this set is equal to $\sum_{\varphi' \subseteq t^{i-1} \wedge |\varphi'| = \ell'} \mathtt{uOcc}(\varphi \cup \varphi', t)$. So $\mathbb{P}^t_\ell(t[i]|\varphi, \ell') = \frac{\sum_{\varphi' \subseteq t^i \wedge |\varphi'| = \ell' - 1} \mathtt{uOcc}(\varphi \cup \{t[i]\} \cup \varphi', t)}{\sum_{\varphi' \subseteq t^{i-1} \wedge |\varphi'| = \ell'} \mathtt{uOcc}(\varphi \cup \varphi', t)}$. Hence the result. □

Proof **(Property 3).** From Lemma 1, we have: $\mathbb{P}^t_\ell(t[i]|\varphi, \ell') = \frac{\sum_{\varphi' \subseteq t^i \wedge |\varphi'| = \ell' - 1} \mathtt{uOcc}(\varphi \cup \{t[i]\} \cup \varphi', t)}{\sum_{\varphi' \subseteq t^{i-1} \wedge |\varphi'| = \ell'} \mathtt{uOcc}(\varphi \cup \varphi', t)}$. First, by definition we have $\mathtt{uOcc}(\varphi \cup \{t[i]\} \cup \varphi', t) = \mathtt{uOcc}(\varphi, t) + \mathtt{uOcc}(\{t[i]\} \cup \varphi', t)$. Let $z_i = \sum_{\varphi' \subseteq t^i \wedge |\varphi'| = \ell' - 1} \mathtt{uOcc}(\varphi \cup \{t[i]\} \cup \varphi', t)$. It implies that $z_i = \sum_{\varphi' \subseteq t^i \wedge |\varphi'| = \ell' - 1} (\mathtt{uOcc}(\varphi, t) + \mathtt{uOcc}(\{t[i]\} \cup \varphi', t))$. Then we have: $z_i = \sum_{\varphi' \subseteq t^i \wedge |\varphi'| = \ell' - 1} \mathtt{uOcc}(\varphi, t) + \sum_{\varphi' \subseteq t^i \wedge |\varphi'| = \ell' - 1} \mathtt{uOcc}(\{t[i]\} \cup \varphi', t)$. But $\sum_{\varphi' \subseteq t^i \wedge |\varphi'| = \ell' - 1} \mathtt{uOcc}(\varphi, t) = \mathtt{uOcc}(\varphi, t) \times \binom{|t^i|}{\ell' - 1}$ and $\sum_{\varphi' \subseteq t^i \wedge |\varphi'| = \ell' - 1} \mathtt{uOcc}(\{t[i]\} \cup \varphi', t) = \omega^+_{\ell'}(t[i], t)$ by definition. Then $z_i = \mathtt{uOcc}(\varphi, t) \times \binom{|t^i|}{\ell' - 1} + \omega^+_{\ell'}(t[i], t)$. We also know that $\mathtt{uOcc}(\varphi, t) = \sum_{k < i \wedge t[k] \in \varphi} \omega^+_1(t[k], t)$. So, $z_i = \left(\sum_{k < i \wedge t[k] \in \varphi} \omega^+_1(t[k], t)\right) \times \binom{|t^i|}{\ell' - 1} + \omega^+_{\ell'}(t[i], t)$. Second, we have $\mathtt{uOcc}(\varphi \cup \varphi', t) = \mathtt{uOcc}(\varphi, t) + \mathtt{uOcc}(\varphi', t)$. By setting $Z_i = \sum_{\varphi' \subseteq t^{i-1} \wedge |\varphi'| = \ell'} \mathtt{uOcc}(\varphi \cup \varphi', t)$, we get then $Z_i = \sum_{\varphi' \subseteq t^{i-1} \wedge |\varphi'| = \ell'} \mathtt{uOcc}(\varphi, t) + \sum_{\varphi' \subseteq t^{i-1} \wedge |\varphi'| = \ell'} \mathtt{uOcc}(\varphi', t)$. But $\sum_{\varphi' \subseteq t^{i-1} \wedge |\varphi'| = \ell'} \mathtt{uOcc}(\varphi, t) = \mathtt{uOcc}(\varphi, t) \times \binom{|t^{i-1}|}{\ell'} = \left(\sum_{k < i \wedge t[k] \in \varphi} \omega^+_1(t[k], t)\right) \times \binom{|t^{i-1}|}{\ell'}$ et $\sum_{\varphi' \subseteq t^{i-1} \wedge |\varphi'| = \ell'} \mathtt{uOcc}(\varphi', t) = \sum_{\star \in \{+,-\}} \omega^\star_{\ell'}(t[i], t)$, so $Z_i = \left(\sum_{k < i \wedge t[k] \in \varphi} \omega^+_1(t[k], t)\right) \times \binom{|t^{i-1}|}{\ell'} + \sum_{\star \in \{+,-\}} \omega^\star_{\ell'}(t[i], t)$. Finally, $\mathbb{P}^t_\ell(t[i]|\varphi, \ell') = \frac{z_i}{Z_i} = \frac{\left(\sum_{k < i \wedge t[k] \in \varphi} \omega^+_1(t[k], t)\right) \times \binom{|t^i|}{\ell' - 1} + \omega^+_{\ell'}(t[i], t)}{\left(\sum_{k < i \wedge t[k] \in \varphi} \omega^+_1(t[k], t)\right) \times \binom{|t^{i-1}|}{\ell'} + \sum_{\star \in \{+,-\}} \omega^\star_{\ell'}(t[i], t)}$. □

Proof **(Property 4).** Let m be the minimum and M the maximum length constraints, the probability of drawing the pattern φ of length ℓ in the database $\mathcal{D}$ denoted by $\mathbb{P}_{[m..M]}(\varphi, \mathcal{D})$, and Z a normalization constant defined by $Z = \sum_{\varphi' \in \mathcal{L}(\mathcal{D})} u^{avg}_{[m..M]}(\varphi', \mathcal{D})$. We know that : $\mathbb{P}_{[m..M]}(\varphi, \mathcal{D}) = \sum_{(j,t) \in \mathcal{D}} \left(\mathbb{P}_{[m..M]}(t_j, \mathcal{D}) \times \mathbb{P}_{[m..M]}(\varphi, t_j)\right)$. But $\mathbb{P}_{[m..M]}(t_j, \mathcal{D}) = \frac{\omega^{avgU}_{[m..M]}(t_j)}{Z}$, then

$$\mathbb{P}_{[m..M]}(\varphi, \mathcal{D}) = \sum\nolimits_{(j,t) \in \mathcal{D}} \left(\frac{\omega^{avgU}_{[m..M]}(t_j)}{Z} \times \mathbb{P}_{[m..M]}(\varphi, t_j) \right). \tag{2}$$

We also know that:

$$\mathbb{P}_{[m..M]}(\varphi, t_j) = \mathbb{P}_{[m..M]}(\ell | t_j) \times \mathbb{P}^{t_j}_{[m..M]}(\varphi | \ell). \tag{3}$$

$\mathbb{P}_{[m..M]}(\ell|t_j) = \frac{\omega^{avgU}_{[\ell..\ell]}(t_j)}{\omega^{avgU}_{[m..M]}(t_j)}$ and $\mathbb{P}^{t_j}_{[m..M]}(\varphi|\ell) = \frac{\mathtt{uOcc}(\varphi,t_j)}{\omega^{avgU}_{[\ell..\ell]}(t_j)\times\ell}$ then by substituting the two terms in (3), $\mathbb{P}_{[m..M]}(\varphi,t_j) = \frac{\omega^{avgU}_{[\ell..\ell]}(t_j)}{\omega^{avgU}_{[m..M]}(t_j)} \times \frac{\mathtt{uOcc}(\varphi,t_j)}{\omega^{avgU}_{[\ell..\ell]}(t_j)\times\ell} = \frac{\mathtt{uOcc}(\varphi,t_j)}{\omega^{avgU}_{[m..M]}(t_j)\times\ell}$. Now, if we replace $\mathbb{P}_{[m..M]}(\varphi,t_j)$ in (2) by its last expression, we get: $\mathbb{P}_{[m..M]}(\varphi,\mathcal{D}) = \sum_{(j,t)\in\mathcal{D}}\left(\frac{\omega^{avgU}_{[m..M]}(t_j)}{Z} \times \frac{\mathtt{uOcc}(\varphi,t_j)}{\omega^{avgU}_{[m..M]}(t_j)\times\ell}\right) = \frac{1}{Z} \times \frac{\sum_{(j,t)\in\mathcal{D}}\mathtt{uOcc}(\varphi,t_j)}{\ell}$. But by definition, we have $\frac{\sum_{(j,t)\in\mathcal{D}}\mathtt{uOcc}(\varphi,t_j)}{\ell} = u^{avg}_{[m..M]}(\varphi,\mathcal{D})$, so $\mathbb{P}_{[m..M]}(\varphi,\mathcal{D}) = \frac{u^{avg}_{[m..M]}(\varphi,\mathcal{D})}{Z}$. Hence the result. □

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: VLDB'94, pp. 487–499. Morgan Kaufmann Publishers Inc. (1994)
2. Ahmed, C.F., Tanbeer, S.K., Jeong, B., Lee, Y.: Efficient tree structures for high utility pattern mining in incremental databases. IEEE Trans. Knowl. Data Eng. **21**(12), 1708–1721 (2009)
3. Al Hasan, M., Zaki, M.J.: Output space sampling for graph patterns. Proc. VLDB Endow. **2**(1), 730–741 (2009)
4. Boley, M., Lucchese, C., Paurat, D., Gärtner, T.: Direct local pattern sampling by efficient two-step random procedures. In: Proceedings of the 17th ACM SIGKDD, pp. 582–590 (2011)
5. Boley, M., Moens, S., Gärtner, T.: Linear space direct pattern sampling using coupling from the past. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 69–77. ACM (2012)
6. Chu, C.J., Tseng, V.S., Liang, T.: An efficient algorithm for mining temporal high utility itemsets from data streams. J. Syst. Softw. **81**(7), 1105–1117 (2008)
7. Diop, L., Diop, C.T., Giacometti, A., Li, D., Soulet, A.: Sequential pattern sampling with norm-based utility. Knowl. Inf. Syst. **62**(5), 2029–2065 (2019). https://doi.org/10.1007/s10115-019-01417-3
8. Diop, L., Diop, C.T., Giacometti, A., Li Haoyuan, D., Soulet, A.: Sequential pattern sampling with norm constraints. In: IEEE International Conference on Data Mining (ICDM), Singapore, November 2018
9. Diop, L., Diop, C.T., Giacometti, A., Soulet, A.: Pattern on demand in transactional distributed databases. Inf. Syst. **104**, 101908 (2022)
10. Dzyuba, V., Leeuwen, M.V., Nijssen, S., De Raedt, L.: Interactive learning of pattern rankings. Int. J. Artif. Intell. Tools **23**(06), 1460026 (2014)
11. Giacometti, A., Soulet, A.: Anytime algorithm for frequent pattern outlier detection. Int. J. Data Sci. Anal. **5**, 119–130 (2016). https://doi.org/10.1007/s41060-016-0019-9
12. Giacometti, A., Soulet, A.: Dense neighborhood pattern sampling in numerical data. In: Proceedings of SDM 2018, pp. 756–764 (2018)
13. Leeuwen, M.: Interactive data exploration using pattern mining. In: Holzinger, A., Jurisica, I. (eds.) Interactive Knowledge Discovery and Data Mining in Biomedical Informatics. LNCS, vol. 8401, pp. 169–182. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43968-5_9
14. Li, H., Huang, H., Chen, Y., Liu, Y., Lee, S.: Fast and memory efficient mining of high utility itemsets in data streams. In: 2008 Eighth IEEE International Conference on Data Mining, pp. 881–886, December 2008

15. Lin, J.C.W., Li, T., Fournier-Viger, P., Hong, T.P., Zhan, J., Voznak, M.: An efficient algorithm to mine high average-utility itemsets. Adv. Eng. Inform. **30**(2), 233–243 (2016)
16. Shie, B.-E., Hsiao, H.-F., Tseng, V.S., Yu, P.S.: Mining high utility mobile sequential patterns in mobile commerce environments. In: Yu, J.X., Kim, M.H., Unland, R. (eds.) DASFAA 2011. LNCS, vol. 6587, pp. 224–238. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20149-3_18
17. Shie, B.E., Yu, P.S., Tseng, V.S.: Mining interesting user behavior patterns in mobile commerce environments. Appl. Intell. **38**(3), 418–435 (2013)
18. Singh, K., Singh, S.S., Kumar, A., Biswas, B.: TKEH: an efficient algorithm for mining top-k high utility itemsets. Appl. Intell. **49**(3), 1078–1097 (2019). https://doi.org/10.1007/s10489-018-1316-x
19. Song, W., Zheng, C., Huang, C., Liu, L.: Heuristically mining the top-k high-utility itemsets with cross-entropy optimization. Appl. Intell., 1–16 (2021). https://doi.org/10.1007/s10489-021-02576-z
20. Tseng, V.S., Wu, C., Fournier-Viger, P., Yu, P.S.: Efficient algorithms for mining the concise and lossless representation of high utility itemsets. IEEE Trans. Knowl. Data Eng. **27**(3), 726–739 (2015)
21. Yao, H., Hamilton, H.J., Butz, C.J.: A foundational approach to mining itemset utilities from databases. In: Proceedings of the Third SIAM International Conference on Data Mining, pp. 482–486 (2004)