PHM: Mining Periodic High-Utility Itemsets

Philippe Fournier-Viger[1](✉), Jerry Chun-Wei Lin[2], Quang-Huy Duong[3], and Thu-Lan Dam[3,4]

[1] School of Natural Sciences and Humanities,
Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China
philfv@hitsz.edu.cn

[2] School of Computer Science and Technology,
Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China
jerrylin@ieee.org

[3] College of Computer Science and Electronic Engineering,
Hunan University, Changsha, China
huydqyb@gmail.com

[4] Faculty of Information Technology, Hanoi University of Industry,
Hanoi, Vietnam
lanfict@gmail.com

Abstract. High-utility itemset mining is the task of discovering high-utility itemsets, i.e. sets of items that yield a high profit in a customer transaction database. High-utility itemsets are useful, as they provide information about profitable sets of items bought by customers to retail store managers, which can then use this information to take strategic marketing decisions. An inherent limitation of traditional high-utility itemset mining algorithms is that they are inappropriate to discover recurring customer purchase behavior, although such behavior is common in real-life situations (for example, a customer may buy some products every day, week or month). In this paper, we address this limitation by proposing the task of periodic high-utility itemset mining. The goal is to discover groups of items that are periodically bought by customers and generate a high profit. An efficient algorithm named PHM (Periodic High-utility itemset Miner) is proposed to efficiently enumerate all periodic high-utility itemsets. Experimental results show that the PHM algorithm is efficient, and can filter a huge number of non periodic patterns to reveal only the desired periodic high-utility itemsets.

Keywords: High-utility itemset · Periodic itemset · Average periodicity

1 Introduction

High-utility itemset mining (HUIM) [4,7,10–12,15] is a popular data mining task. It has attracted a lot of attention in recent years. It extends the traditional problem of *Frequent Itemset Mining* (FIM) [1]. This latter consists of discovering frequent itemsets, i.e. groups of items (itemsets) appearing frequently in a transaction database [1]. FIM has many applications. However, an important

P. Perner (Ed.): ICDM 2016, LNAI 9728, pp. 64–79, 2016.
DOI: 10.1007/978-3-319-41561-1_6

limitation of FIM is that it assumes that each item cannot appear more than once in each transaction and that all items have the same importance (e.g. weight, unit profit or value). *High-Utility Itemset Mining* (HUIM) addresses this issue by considering that each item may have non binary purchase quantities in transactions and that each item has a weight (e.g. unit profit). The goal of HUIM is to discover itemsets having a high utility (e.g. yielding a high profit) in a transaction database. Besides, market basket analysis, HUIM has several other applications such as website click stream analysis, and biomedical applications [11,15]. Mining high-utility itemsets is widely recognized as more challenging than FIM because the utility measure used in HUIM is not anti-monotonic, i.e. a high utility itemset may have supersets or subsets having lower, equal or higher utilities [4]. Thus, techniques for reducing the search space in FIM cannot be directly reused in HUIM. Though several algorithms have been proposed for HUIM [4,7,10–12,15], an inherent limitation of these algorithms is that they are inappropriate to discover recurring customer purchase behavior, although such behavior is common in real-life situations. For example, in a retail store, some customers may buy some set of products on approximately a daily or weekly basis. Detecting these purchase patterns is useful to better understand the behavior of customers and thus adapt marketing strategies, for example by offering specific promotions to cross-promote products such as reward or points to customers who are buying a set of products periodically. In the field of FIM, algorithms have been proposed to discover periodic frequent patterns (PFP) [2,3,8,9,13,14] in a transaction database. However, these algorithms are inadequate to find periodic patterns that yield a high profit, as they only select patterns based on their frequency. Hence, these algorithms may find a huge amount of periodic patterns that generate a low profit and miss many rare periodic patterns that yield a high profit.

To address this limitation of previous work, this paper proposes the task of periodic high-utility itemset mining. The goal is to efficiently discover all groups of items that are bought together periodically and generate a high profit, in a customer transaction database. The contributions of this paper are fourfold. First, the concept of periodic patterns used in FIM is combined with the concept of HUIs to define a new type of patterns named *periodic high-utility itemsets* (PHIs), and its properties are studied. Second, novel measures of a pattern's periodicity named *average periodicity* and *minimum periodicity* are introduced to provide a flexible way of assessing the periodicity of patterns. Third, an efficient algorithm named PHM (Periodic High-utility itemset Miner) is proposed to efficiently discover the periodic high-utility itemsets. Fourth, an extensive experimental evaluation is carried to compare the efficiency of PHM with the state-of-the-art FHM algorithm for HUIM. Experimental results show that the PHM algorithm is efficient, and can filter a huge number of non periodic patterns to reveal only the desired itemsets. The rest of this paper is organized as follows. Sections 2, 3, 4 and 5 respectively present preliminaries related to HUIM, related work, the PHM algorithm, the experimental evaluation and the conclusion.

2 Related Work

This section reviews related work in high-utility itemset mining and periodic frequent pattern mining.

2.1 High-Utility Itemset Mining

Definition 1 (Transaction database). Let I be a set of items (symbols). A *transaction database* is a set of transactions $D = \{T_1, T_2, ..., T_n\}$ such that for each transaction T_c, $T_c \in I$ and T_c has a unique identifier c called its Tid. Each item $i \in I$ is associated with a positive number $p(i)$, called its external utility (e.g. unit profit). For each transaction T_c such that $i \in T_c$, a positive number $q(i, T_c)$ is called the internal utility of i (e.g. purchase quantity).

Example 1. Consider the database of Table 1, which will be used as running example. This database contains seven transactions ($T_1, T_2 ... T_7$). Transaction T_3 indicates that items a, b, c, d, and e appear in this transaction with an internal utility of respectively 1, 5, 1, 3 and 1. Table 2 indicates that the external utility of these items are respectively 5, 2, 1, 2 and 3.

Table 1. A transaction database

TID	Transaction
T_1	$(a, 1), (c, 1),$
T_2	$(e, 1)$
T_3	$(a, 1), (b, 5), (c, 1), (d, 3), (e, 1)$
T_4	$(b, 4), (c, 3), (d, 3), (e, 1)$
T_5	$(a, 1), (c, 1), (d, 1)$
T_6	$(a, 2), (c, 6), (e, 2)$
T_7	$(b, 2), (c, 2), (e, 1)$

Table 2. External utility values

Item	a	b	c	d	e
Unit profit	5	2	1	2	3

Definition 2 (Utility of an item/itemset). The utility of an item i in a transaction T_c is denoted as $u(i, T_c)$ and defined as $p(i) \times q(i, T_c)$. The utility of an itemset X (a group of items $X \subseteq I$) in a transaction T_c is denoted as $u(X, T_c)$ and defined as $u(X, T_c) = \sum_{i \in X} u(i, T_c)$. The utility of an itemset X (in a database) is denoted as $u(X)$ and defined as $u(X) = \sum_{T_c \in g(X)} u(X, T_c)$, where $g(X)$ is the set of transactions containing X.

Example 2. The utility of item a in T_6 is $u(a, T_6) = 5 \times 2 = 10$. The utility of the itemset $\{a, c\}$ in T_6 is $u(\{a, c\}, T_6) = u(a, T_6) + u(c, T_6) = 5 \times 2 + 1 \times 6 = 16$. The utility of the itemset $\{a, c\}$ (in the database) is $u(\{a, c\}) = u(a) + u(c) = u(a, T_1) + u(a, T_3) + u(a, T_5) + u(a, T_6) + u(c, T_1) + u(c, T_3) + u(c, T_5) + u(c, T_6) = 5 + 5 + 5 + 10 + 1 + 1 + 1 + 6 = 34$.

Definition 3 (High-utility itemset mining). The *problem of high-utility itemset mining* is to discover all high-utility itemsets [4,7,10–12,15]. An itemset X *is a* high-utility itemset if its utility $u(X)$ is no less than a user-specified minimum utility threshold *minutil* given by the user.

Example 3. If $minutil = 30$, the complete set of HUIs is $\{a, c\} : 34$, $\{a, c, e\} : 31$, $\{b, c, d\} : 34$, $\{b, c, d, e\} : 40$, $\{b, c, e\} : 37$, $\{b, d\} : 30$, $\{b, d, e\} : 36$, and $\{b, e\} : 31$, where each HUI is annotated with its utility.

In HUIM, the utility measure is not monotonic or anti-monotonic [12,15], i.e., an itemset may have a utility lower, equal or higher than the utility of its subsets. Several HUIM algorithms circumvent this problem by overestimating the utility of itemsets using the *Transaction-Weighted Utilization (TWU)* measure [12,15], which is anti-monotonic, and defined as follows.

Definition 4 (Transaction weighted utilization). The *transaction utility (TU)* of a transaction T_c is the sum of the utility of all the items in T_c.i.e. $TU(T_c) = \sum_{x \in T_c} u(x, T_c)$. The *transaction-weighted utilization (TWU)* of an itemset X is defined as the sum of the transaction utility of transactions containing X, *i.e.* $TWU(X) = \sum_{T_c \in g(X)} TU(T_c)$.

Example 4. The TUs of T_1, T_2, T_3, T_4, T_5, T_6 and T_7 are respectively 6,3, 25, 20, 8, 22 and 9. The TWU of single items a, b, c, d, e are respectively 61, 54, 90, 53 and 79. $TWU(\{c, d\}) = TU(T_3) + TU(T_4) + TU(T_5) = 25 + 20 + 8 = 53$.

Theorem 1 (Pruning search space using the TWU). *Let X be an itemset, if* $TWU(X) < minutil$*, then* X *and its supersets are low utility.* [12]

Algorithms such as Two-Phase [12], BAHUI [10], PB [7], and UPGrowth+ [15] utilizes the above property to prune the search space. They operate in two phases. In the first phase, they identify candidate high utility itemsets by calculating their TWUs. In the second phase, they scan the database to calculate the exact utility of all candidates found in the first phase to eliminate low utility itemsets. Recently, an alternative algorithm called *HUI-Miner*[11] was proposed to mine HUIs directly using a single phase. Then, a faster depth-first search algorithm FHM [4] was proposed, which extends HUI-Miner. In FHM, each itemset is associated with a structure named *utility-list* [4,11]. Utility-lists allow calculating the utility of an itemset quickly by making join operations with utility-lists of shorter patterns. Utility-lists are defined as follows.

Definition 5 (Utility-list). Let $\succ$ be any total order on items from I. The *utility-list*of an itemset X in a database D is a set of tuples such that there is a tuple $(tid, iutil, rutil)$ for each transaction T_{tid} containing X. The *iutil* element of a tuple is the utility of X in T_{tid}. i.e., $u(X, T_{tid})$. The *rutil* element of a tuple is defined as $\sum_{i \in T_{tid} \wedge i \succ x \forall x \in X} u(i, T_{tid})$.

Example 5. Assume that $\succ$ is the alphabetical order. The utility-list of $\{a\}$ is $\{(T_1, 5, 1)$, $T_3, 5, 20)$, $(T_5, 5, 3)$, $(T_6, 10, 12)\}$. The utility-list of $\{d\}$ is $\{(T_3, 6, 3)$, $(T_4, 6, 3)$, $(T_5, 2, 0)\}$. The utility-list of $\{a, d\}$ is $\{(T_3, 11, 3)$, $(T_5, 7, 0)\}$.

To discover HUIs, FHM performs a single database scan to create utility-lists of patterns containing single items. Then, longer patterns are obtained by performing the join operation of utility-lists of shorter patterns. The join operation for single items is performed as follows. Consider two items x, y such that $x \succ y$, and their utility-lists $ul(\{x\})$ and $ul(\{y\})$. The utility-list of $\{x, y\}$ is obtained by creating a tuple $(ex.tid, ex.iutil + ey.iutil, ey.rutil)$ for each pair of tuples $ex \in ul(\{x\})$ and $ey \in ul(\{y\})$ such that $ex.tid = ey.tid$. The join operation for two itemsets $P \cup \{x\}$ and $P \cup \{y\}$ such that $x \succ y$ is performed as follows. Let $ul(P)$, $ul(\{x\})$ and $ul(\{y\})$ be the utility-lists of P, $\{x\}$ and $\{y\}$. The utility-list of $P \cup \{x, y\}$ is obtained by creating a tuple $(ex.tid, ex.iutil + ey.iutil - ep.iutil, ey.rutil)$ for each set of tuples $ex \in ul(\{x\})$, $ey \in ul(\{y\})$, $ep \in ul(P)$ such that $ex.tid = ey.tid = ep.tid$. Calculating the utility of an itemset using its utility-list and pruning the search space is done as follows.

Property 1 (Calculating utility of an itemset using its utility-list). The utility of an itemset is the sum of *iutil* values in its utility-list [11].

Theorem 2 (Pruning search space using utility-lists). *Let X be an itemset. Let the* extensions *of X be the itemsets that can be obtained by appending an item y to X such that $y \succ i$, $\forall i \in X$. If the sum of iutil and rutil values in $ul(X)$ is less than minutil, X and its extensions are low utility* [11].

FHM is very efficient. However, an important limitation of current HUIM algorithms is that they are not designed for discovering periodic patterns.

2.2 Periodic Frequent Pattern Mining

In the field of FIM, algorithms have been proposed to discover periodical frequent patterns (PFP) [2,3,8,9,13,14] in a transaction database. Discovering PFP has applications in many domains such as web mining, bioinformatics, and market basket analysis [14]. The concept of PFP is defined as follows [14].

Definition 6 (Periods of an itemset). Let there be a database $D = \{T_1, T_2, ..., T_n\}$ containing n transactions, and an itemset X. The set of transactions containing X is denoted as $g(X) = \{T_{g_1}, T_{g_2} ..., T_{g_k}\}$, where $1 \leq g_1 < g_2 < ... < g_k \leq n$. Two transactions $T_x \supset X$ and $T_y \supset X$ are said to be *consecutive with respect to* X if there does not exist a transaction $T_w \in g(X)$ such that $x < w < y$. The *period* of two consecutive transactions T_x and T_y in $g(X)$ is defined as $pe(T_x, T_y) = (y - x)$, that is the number of transactions between T_x *and* T_y. The *periods of an itemset* X is a list of periods defined as $ps(X) = \{g_1 - g_0, g_2 - g_1, g_3 - g_2, ...g_k - g_{k-1}, g_{k+1} - g_k\}$, where g_0 and $g_k + 1$ are constants defined as $g_0 = 0$ and $g_k + 1 = n$. Thus, $ps(X) = \bigcup_{1 \leq z \leq k+1} (g_z - g_{z-1})$.

Example 6. For the itemset $\{a, c\}$, The list of transactions containing $\{a, c\}$ is $g(\{a, c\}) = \{T_1, T_3, T_5, T_6\}$. Thus, the periods of this itemset are $ps(\{a, c\}) = \{1, 2, 2, 1, 1\}$.

Definition 7 (Periodic frequent pattern). The maximum periodicity of an itemset X is defined as $maxper(X) = max(ps(X))$ [14]. An itemset X is a periodic frequent pattern (PFP) if $|g(X)| \geq minsup$ and $maxper(X) < maxPer$, where $minsup$ and $maxPer$ are user-defined thresholds [14].

The first algorithm for mining PFPs is PFP-Tree [14]. It utilizes a tree-based and pattern-growth approach for discovering PFPs. Then, the MTKPP algorithm [2] was proposed for discovering the k most frequent PFPs in a database, where k is a user-specified parameter. MTKPP utilizes a vertical structure to maintain information about itemsets in the database A variation of the PF-Tree algorithm named the ITL-Tree was also introduced [3] to reduce the time for mining PFPs by approximating the periodicity of itemsets. Another approximate algorithm for PFP mining was recently proposed [9]. Other extensions of the PF-Tree algorithm named MIS-PF-tree [8] and MaxCPF [13] were respectively proposed to mine PFPs using multiple *minsup* thresholds, and multiple *minsup* and *minper* thresholds. An important limitation of traditional algorithms for PFP mining is that they are inadequate to find periodic patterns that yield a high profit, since they only consider the support (frequency) of patterns. Hence, they may find a huge amount of periodic patterns that yield a low profit and miss many rare periodical patterns that yield a high profit.

3 The PHM Algorithm

To address the aforementioned limitation of HUI and PFP mining algorithms, this section introduces the concept of periodic high-utility itemsets (PHUIs). The first subsection present novel measures to assess the periodicity of HUIs, while the second subsection presents and efficient algorithm named PHM (Periodic High-Utility Itemset Miner) to discover PHUIs efficiently.

3.1 Measuring the Periodicity of High-Utility Patterns

A drawback of the maximum periodicity measure used by most PFP algorithms is that an itemset is automatically discarded if it has a single period of length greater than the *maxPer* threshold. Thus, this measure may be viewed as too strict. To provide a more flexible way of evaluating the periodicity of patterns, the concept of *average periodicity* is introduced in the proposed algorithm.

Definition 8 (Average periodicity of an itemset). The average periodicity of an itemset X is defined as $avgper(X) = \sum_{g \in ps(X)} / |ps(X)|$.

Example 7. The periods of itemsets $\{a, c\}$ and $\{e\}$ are respectively $ps(\{a, c\}) = \{1, 2, 2, 1, 1\}$ and $ps(\{e\}) = \{2, 1, 1, 2, 1, 0\}$. The average periodicities of these itemsets are respectively $avgper(\{a, c\}) = 1.4$ and $avgper(\{e\}) = 1.16$.

Lemma 1 (Relationship between average periodicity and support). *Let X be an itemset appearing in a database D. An alternative and equivalent way of calculating the average periodicity of X is $avgper(X) = |D|/(|g(X)|+1)$.*

Proof. Let $g(X) = \{T_{g_1}, T_{g_2}, \ldots, T_{g_k}\}$ be the set of transactions containing X, such that $g_1 < g_2 < \ldots < g_k$. By definition, $avgper(X) = \sum_{g \in ps(X)} / |ps(X)|$. To prove that the lemma holds, we need to show that $\sum_{g \in ps(X)} / |ps(X)| = |D|/(|g(X)| + 1)$.
(1) We first show that $\sum_{g \in ps(X)} = |D|$, as follows:
$\sum_{g \in ps(X)} = (g_1 - g_0) + (g_2 - g_1) + \ldots (g_k - g_{k-1}) + (g_{k+1} - g_k)\}$
$= \sum_{g \in ps(X)} = g_0 + (g_1 - g_1) + (g_2 - g_2) + \ldots (g_k - g_k) + (g_{k+1})$
$= g_{k+1} - g_0 = |D|$.
(2) We then show that $|ps(X)| = |g(X)| + 1$, as follows:
By definition, $ps(X) = \bigcup_{1 \leq z \leq k+1} (g_z - g_{z-1})$. Thus, the set $ps(X)$ contains $k+1$ elements. Since X appears in k transactions, $sup(X) = k$, and thus $|ps(X)| = |g(X)| + 1$.
Since (1) and (2) holds, the lemma holds. □

The above lemma is important as it provides an efficient way of calculating the average periodicity of itemsets in a database D. The term $|D|$ can be calculated once, and thereafter the average periodicity of any itemset X can be obtained by only calculating $|g(X)| + 1$, and then dividing $|D|$ by the result. This is more efficient than calculating the average periodicity using Definition 8. Besides, this lemma is important as it shows that there is a relationship between the support used in FIM and the average periodicity of a pattern. Although the average periodicity is useful as it measures what is the typical period length of an itemset, it should not be used as the sole measure for evaluating the periodicity of a pattern because it does not consider whether an itemset has periods that vary widely or not. For example, the itemset $\{b, d\}$ has an average periodicity of 2.33. However, this is misleading since this itemset only appears in transaction T_3 and T_4, and its periods $ps(\{T_3, T_4\}) = \{3, 1, 4\}$ vary widely. Intuitively, this pattern should not be a periodic pattern. To avoid finding patterns having periods that vary widely, our solution is to combine the average periodicity measure with other periodicity measure(s). The following measures are combined with the average periodicity to achieve this goal. First, we define the *minimum periodicity* of an itemset as $minper(X) = min(ps(X))$ to avoid discovering itemsets having some very short periods. But this measure is not reliable since the first and last period of an itemset are respectively equal to 1 or 0 if the itemset respectively appears in the first or the last transaction of the database. For example, the last period of itemset {e} is 0, because it appears in the last transaction (T_7), and thus its minimum periodicity is 0. Our solution to this issue is to exclude the first and last periods of an itemset from the calculation of the minimum periodicity. Moreover, if the set of periods is empty as a result of this exclusion, the minimum periodicity is defined as ∞. In the rest of this paper, we consider this definition. Second, we consider the *maximum periodicity* of an itemset $maxper(X)$ as defined in the previous section. The rationale for using this measure in combination with the average periodicity is that it can avoid discovering periodical patterns that do not occur for long periods of time. In terms of calculation costs, a reason for choosing the minimum periodicity, maximum periodicity and average periodicity as measure is that they can be calculated very efficiently for an itemset X by scanning the list of transactions $g(X)$ only once. That is, calculating these measures do not require to store the set of periods $ps(X)$ in memory. Conversely, other measures such as the standard deviation would require to calculate all periods of an itemset beforehand. Thus, we define the concept of periodic high-utility itemsets by considering the minimum periodicity, maximum periodicity and average periodicity measures.

Definition 9 (Periodic high-utility itemsets). Let *minutil, minAvg, maxAvg, minPer* and *maxPer* be positive numbers, provided by the user. An itemset X is a *periodic high-utility itemset* if and only if $minAvg \leq avgper(X) \leq maxAvg$, $minper(X) \geq minPer$, $maxper(X) \leq maxPer$, *and* $u(X) \geq minutil$.

Table 3. The set of PHUIs in the running example

Itemset	u(X)	$\|g(X)\|$	Minper(X)	Maxper(X)	Avgper(X)
$\{b\}$	22	3	1	3	1.75
$\{b, e\}$	31	3	1	3	1.75
$\{b, c, e\}$	37	3	1	3	1.75
$\{b, c\}$	28	3	1	3	1.75
$\{a\}$	25	4	1	2	1.4
$\{a, c\}$	34	4	1	2	1.4
$\{c, e\}$	27	4	1	3	1.4

For example, if $minutil = 20$, $minPer = 1$, $maxPer = 3$, $minAvg = 1$, and $maxAvg = 2$, the complete set of PHUIs is shown in Table 3. To develop an efficient algorithm for mining PHUIs, it is important to design efficient pruning strategies. To use the periodicity measures for pruning the search space, the following theorems are presented.

Lemma 2 (Monotonicity of the average periodicity). Let X and Y be itemsets such that $X \subset Y$. It follows that $avgper(Y) \geq avgper(X)$.

Proof. The average periodicities of X and Y are respectively $avgper(X) = |D|/(|g(X)| + 1)$ and $avgper(Y) = |D|/(|g(Y)| + 1)$. Because $X \subset Y$, it follows that $g(Y) \subseteq g(X)$. Hence, $avgper(Y) \geq avgper(X)$. □

Lemma 3 (Monotonicity of the minimum periodicity). Let X and Y be itemsets such that $X \subset Y$. It follows that $minper(Y) \geq minper(X)$.

Proof. Since $X \subset Y$, $g(Y) \subseteq g(X)$. If $g(Y) = g(X)$, then X and Y have the same periods, and thus $minper(Y) = minper(X)$. If $g(Y) \subset g(X)$, then for each transaction $T_x \in g(X) \setminus g(Y)$, the corresponding periods in $ps(X)$ will be replaced by a larger period in $ps(Y)$. Thus, any period in $ps(Y)$ cannot be smaller than a period in $ps(X)$. Hence, $minper(Y) \geq minper(X)$. □

Lemma 4 (Monotonicity of the maximum periodicity). Let X and Y be itemsets such that $X \subset Y$. It follows that $maxper(Y) \geq maxper(X)$ [14].

Theorem 3 (Maximum periodicity pruning). Let X be an itemset appearing in a database D. X and its supersets are not PHUIs if $maxper(X) > maxPer$. Thus, if this condition is met, the search space consisting of X and all its supersets can be discarded.

Proof. By definition, if $maxper(X) > maxPer$, X is not a PHUI. By Lemma 4, supersets of X are also not PHUIs.

Theorem 4 (Average periodicity pruning). Let X be an itemset appearing in a database D. X is not a PHUI as well as all of its supersets if $avgper(X) > maxAvg$, or equivalently if $|g(X)| < (|D|/maxAvg) - 1$. Thus, if this condition is met, the search space consisting of X and all its supersets can be discarded.

Proof. By definition, if $avgper(X) > maxAvg$, X is not a PHUI. By Lemma 2, supersets of X are also not PHUIs. The pruning condition $avgper(X) > maxAvg$ is rewritten as: $|D|/(|g(X)|+1) > maxAvg$. Thus, $1/(|g(X)|+1) > maxAvg/|D|$, which can be further rewritten as $|g(X)| + 1 < |D|/maxAvg$, and as $|g(X)| < (|D|/maxAvg) - 1$. □

3.2 The Algorithm

The proposed PHM algorithm is a utility-list based algorithm, inspired by the FHM algorithm [4], where the utility-list of each itemset X is annotated with two additional values: $minper(X)$ and $maxper(X)$. The main procedure of PHM (Algorithm 1) takes a transaction database as input, and the $minutil$, $minAvg$, $maxAvg$, $minPer$ and $maxPer$ thresholds. The algorithm first scans the database to calculate $TWU(\{i\})$, $minper(\{i\})$, $maxper(\{i\})$, and $|g(\{i\})|$ for each item $i \in I$. Then, the algorithm calculates the value $\gamma = (|D|/maxAvg) - 1$ to be later used for pruning itemsets using Theorem 4. Then, the algorithm identifies the set I^* of all items having a TWU no less than $minutil$, a maximum periodicity no greater than $maxPer$, and appearing in no less than γ transactions (other items are ignored since they cannot be part of a PHUI by Theorems 1, 3 and 4). The TWU values of items are then used to establish a total order $\succ$ on items, which is the order of ascending TWU values (as suggested in [11]). A database scan is then performed. During this database scan, items in transactions are reordered according to the total order $\succ$, the utility-list of each item $i \in I^*$ is built and a structure named EUCS (Estimated Utility Co-Occurrence Structure) is built [4]. This latter structure is defined as a set of triples of the form $(a, b, c) \in I^* \times I^* \times \mathbb{R}$. A triple (a,b,c) indicates that TWU($\{a, b\}$) $= c$. The EUCS can be implemented as a triangular matrix (as shown in Fig. 1 for the running example), or as a hash map of hash maps where only tuples of the form (a, b, c) such that $c \neq 0$ are kept. After the construction of the EUCS, the

Algorithm 1. The PHM algorithm

```
input : $D$: a transaction database,
        $minutil$, $minAvg$, $maxAvg$, $minPer$ and $maxPer$: the thresholds
output: the set of periodic high-utility itemsets
Scan $D$ once to calculate $TWU(\{i\})$, $minper(\{i\})$, $maxper(\{i\})$, and $|g(\{i\})|$
  for each item $i \in I$;
$\gamma \leftarrow (|D|/maxAvg) - 1$;
$I^* \leftarrow$ each item $i$ such that TWU($i$) $\geq minutil$, $|g(\{i\})| \geq \gamma$ and
  $maxper(\{i\}) \leq maxPer$;
Let $\succ$ be the total order of TWU ascending values on $I^*$;
Scan $D$ to build the utility-list of each item $i \in I^*$ and build the $EUCS$
  structure;
Search ($\emptyset$, $I^*$, $\gamma$, $minutil$, $minAvg$, $minPer$, $maxPer$, $EUCS$, $|D|$);
```

Item	a	b	c	d
b	25			
c	61	54		
d	33	45	53	
e	47	54	76	45

Fig. 1. The EUCS

Item	a	b	c	d
b	1			
c	4	3		
d	2	2	3	
e	2	3	4	2

Fig. 2. The ESCS

depth-first search exploration of itemsets starts by calling the recursive procedure *Search* with the empty itemset $\emptyset$, the set of single items I^*, γ, $minutil$, $minAvg$, $minPer$, $maxPer$, the EUCS structure, and $|D|$.

The *Search* procedure (Algorithm 2) takes as input an itemset P, extensions of P having the form Pz meaning that Pz was previously obtained by appending an item z to P, γ, $minutil$, $minAvg$, $minPer$, $maxPer$, the EUCS, and $|D|$. The search procedure performs a loop on each extension Px of P. In this loop, the average periodicity of Px is obtained by dividing $|D|$ by the number of elements in the utility list of Px plus one (by Lemma 1). Then, if the average periodicity of Px is in the $[minAvg, maxAvg]$ interval, the sum of the *iutil* values of the utility-list of Px is no less than $minutil$ (cf. Property 1), the minimum/maximum periodicity of Px is no less/not greater than $minPer/maxPer$ according to the values stored in its utility-list, then Px is a PHUI and it is output. Then, if the sum of *iutil* and *rutil* values in the utility-list of Px are no less than $minutil$, the number of elements in the utility list of Px is no less than γ, and $maxper(Px)$ is no greater than $maxPer$, it means that extensions of Px should be explored (by Theorems 1, 3 and 4). This is performed by merging Px with all extensions Py of P such that $y \succ x$ to form extensions of the form Pxy containing $|Px| + 1$ items. The utility-list of Pxy is then constructed by calling the *Construct* procedure (cf. Algorithm 3), to join the utility-lists of P, Px and Py. This latter procedure is mainly the same as in HUI-Miner [11], with the exception that periods are calculated during utility-list construction to obtain $maxPer(Pxy)$ and $minPer(Pxy)$ (not shown). Then, a recursive call to the *Search* procedure with Pxy is done to calculate its utility and explore its extension(s). The *Search* procedure starts from single items, recursively explores the search space of itemsets by appending single items, and only prunes the search space using Theorems 1, 3 and 4. Thus, it can be easily seen that this procedure is correct and complete to discover all PHUIs.

Algorithm 2. The *Search* procedure

```
input : P: an itemset, ExtensionsOfP: a set of extensions of P, $\gamma$, minutil,
        minAvg, minPer, maxPer, the EUCS structure, $|D|$
output: the set of periodic high-utility itemsets
1  foreach itemset $Px \in ExtensionsOfP$ do
2      $avgperPx \leftarrow |D|/(|Px.utilitylist| + 1)$;
3      if $SUM(Pxy.utilitylist.iutils) \geq minutil \wedge$
       $minAvg \leq avgperPx \leq maxAvg \wedge Px.utilitylist.minp \geq$
       $minPer \wedge Px.utilitylist.maxp \leq maxPer \wedge$ then output $Px$;
4      if $SUM(Px.utilitylist.iutils)+SUM(Px.utilitylist.rutils) \geq minutil \wedge$
       $avgperPx \geq \gamma$ and $Px.utilitylist.maxp \leq maxPer$ then
5          $ExtensionsOfPx \leftarrow \emptyset$;
6          foreach itemset $Py \in ExtensionsOfP$ such that $y \succ x$ do
7              if $\exists(x, y, c) \in EUCS$ such that $c \geq minutil)$ then
8                  $Pxy \leftarrow Px \cup Py$;
9                  $Pxy.utilitylist \leftarrow$ Construct $(P, Px, Py)$;
10                 $ExtensionsOfPx \leftarrow ExtensionsOfPx \cup \{Pxy\}$;
11             end
12         end
13         Search ($Px$, $ExtensionsOfPx$, $\gamma$, $minutil$, $minAvg$, $minPer$, $maxPer$,
           $EUCS$, $|D|$);
14     end
15 end
```

Furthermore, in the implementation of PHM, two additional optimizations are included, which are briefly described next. **Optimization 1. Estimated Average Periodicity Pruning (EAPP).** The PHM algorithm creates a structure called EUCS to store the TWU of all pairs of items occurring in the database, and this structure is used to prune any itemset Pxy containing a pair of items {x,y} having a TWU lower than *minutil* (Line 7 of the search procedure). The strategy EAPP is a novel strategy that uses the same idea but prune itemsets using the average periodicity instead of the utility. During the second database scan, a novel structure called ESCS (Estimated Support Co-occurrence Structure) is created to store $|g(\{x, y\})|$ for each pair of items {x,y} (as shown in Fig. 2). Then, Line 7 of the search procedure is modified to prune itemset Pxy if $|g(\{x, y\})|$ is less than γ by Theorem 4. **Optimization 2. Abandoning List Construction early (ALC).** Another strategy introduced in PHM is to stop

Algorithm 3. The Construct procedure

```
input  : $P$: an itemset, $Px$: the extension of $P$ with an item $x$, $Py$: the
         extension of $P$ with an item $y$
output: the utility-list of $Pxy$
1  $UtilityListOfPxy \leftarrow \emptyset$;
2  foreach $tuple\ ex \in Px.utilitylist$ do
3      if $\exists ey \in Py.utilitylist\ and\ ex.tid = exy.tid$ then
4          if $P.utilitylist \neq \emptyset$ then
5              Search element $e \in P.utilitylist$ such that $e.tid = ex.tid.$;
6              $exy \leftarrow (ex.tid, ex.iutil + ey.iutil - e.iutil, ey.rutil)$;
7          end
8          else
9              $exy \leftarrow (ex.tid, ex.iutil + ey.iutil, ey.rutil)$;
10         end
11         $period_{exy} \leftarrow calculatePeriod$(exy.tid,UtilityListOfPxy);
12         $UpdateMinPerMaxPer(UtilityListOfPxy, period_{exy})$;
13         $UtilityListOfPxy \leftarrow UtilityListOfPxy \cup \{exy\}$;
14     end
15 end
16 return $UtilityListPxy$;
```

constructing the utility-list of an itemset if a specific condition is met, indicating that the itemset cannot be a PHUI. By Theorem 4, an itemset Pxy cannot be a PHUI, if it appears in less than $\gamma = (|D|/maxAvg) - 1$ transactions. The strategy ALC consists of modifying the Construct procedure (Algorithm 3) as follows. The first modification is to initialize a variable max with the value γ in Line 1. The second modification is to the following lines, where the utility-list of Pxy is constructed by checking if each tuple in the utility-lists of Px appears in the utility-list of Py (Line 3). For each tuple not appearing in Py, the variable max is decremented by 1. If max is smaller than γ, the construction of the utility-list of Pxy can be stopped because $|g(Pxy)|$ will not be higher than γ. Thus Pxy is not a PHUI by Theorem 4, and its extensions can also be ignored.

4 Experimental Study

We performed an experimental study to assess the performance of PHM. The experiment was performed on a computer with a sixth generation 64 bit Core i5 processor running Windows 10, and equipped with 12 GB of free RAM. We compared the performance of the proposed PHM algorithm with the state-of-the-art FHM algorithm for mining HUIs. All memory measurements were done using the Java API. The experiment was carried on four real-life datasets commonly used in the HUIM litterature: *retail*, *mushroom*, *chainstore* and *foodmart*. These datasets have varied characteristics and represents the main types of data typically encountered in real-life scenarios (dense, sparse and long transactions). Let

$|I|$, $|D|$ and A represents the number of transactions, distinct items and average transaction length of a dataset. *retail* is a sparse dataset with many different items ($|I| = 16{,}470$, $|D| = 88{,}162$, $A = 10{,}30$). *mushroom* is a dense dataset with long transactions ($|I| = 119$, $|D| = 8{,}124$, $A = 23$). *chainstore* is a dataset that contains a huge number of transactions ($|I| = 461$, $|D| = 1{,}112{,}949$, $A = 7.23$). *foodmart* is a sparse dataset ($|I| = 1{,}559$, $|D| = 4{,}141$, $A = 4.4$). The *chainstore* and *foodmart* datasets are real-life customer transaction databases containing real external and internal utility values. The *retail* and *mushroom* datasets contains synthetic utility values, generated randomly [11,15]. In the experiment, PHM was run on each dataset with fixed $minper$ and $minAvg$ values, while varying the $minutil$ threshold and the values of the $maxAvg$ and $maxper$ parameters. In these experiments, the values for the periodicity thresholds have been found empirically for each dataset (as they are dataset specific), and were chosen to show the trade-off between the number of periodic patterns found and the execution time. Note that results for varying the $minper$ and $minAvg$ values are not shown because these parameters have less influence on the patterns found than the other parameters. Thereafter, the notation $PHM\ V\text{-}W\text{-}X\text{-}Y$ represents the PHM algorithm with $minper = V$, $maxper = W$, $minAvg = X$, and $maxAVG = Y$. Figure 3 compares the execution times of PHM for various parameter values and FHM. Figure 4, compares the number of PHUIs found by PHM for various parameter values, and the number of HUIs found by FHM.

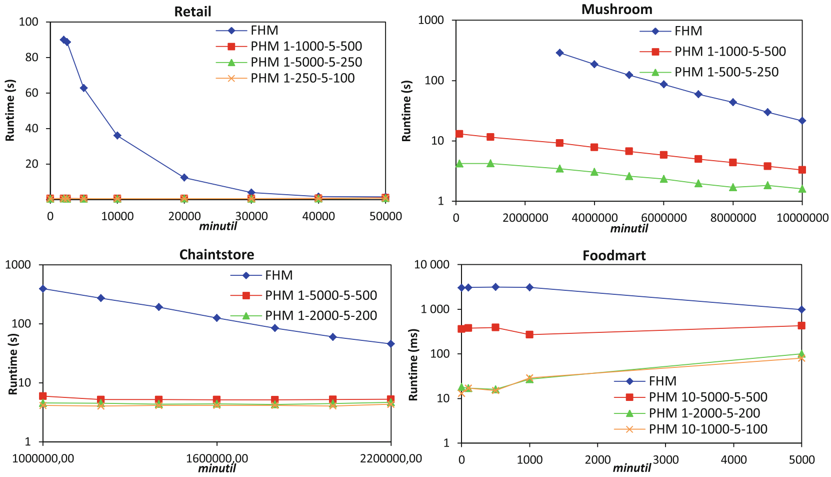

Fig. 3. Execution times

It can first be observed that mining PHUIs using PHM can be much faster than mining HUIs. The reason for the excellent performance of PHM is that it prunes a large part of the search space using its designed pruning strategies based on the maximum and average periodicity measures. For all datasets, it can

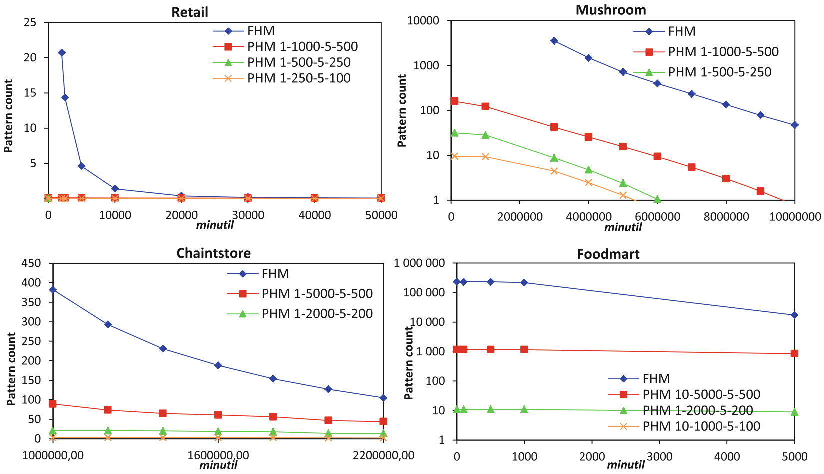

Fig. 4. Number of patterns found

be found that a huge amount of HUIs are non periodic, and thus pruning non periodic patterns leads to a massive performance improvement. For example, for the lowest *minutil*, *maxPer* and *maxAvg* values on these datasets, PHM is respectively up to 214, 127, 100 and 230 times faster than FHM. In general, the more the periodicity thresholds are restrictive, the more the gap between the runtime of FHM and PHM increases. A second observation is that the number of PHUIs can be much less than the number of HUIs (see Fig. 4). For example, on *retail*, 20,714 HUIs are found for $minutil = 2,000$. But only 110 HUIs are PHUIs for PHM 1-1000-5-500, and only 7 for PHM 1-250-5-150. Some of the patterns found are quite interesting as they contain several items. For example, it is found that items with product ids 32, 48 and 39 are periodically bought with an average periodicity of 16.32, a minimum periodicity of 1, and a maximum periodicity of 170. Huge reduction in the number of patterns are also observed on the other datasets. These overall results show that the proposed PHM algorithm is useful as it can filter a huge amount of non periodic HUIs encountered in real datasets, and can run faster. Memory consumption was also compared, although detailed results are not shown as a figure due to space limitations. It was observed that PHM can use up to 10 times less memory than FHM depending on how parameters are set. For example, on *chainstore* and $minutil = 1,000,000$, FHM and PHM 1-5000-5-500 respectively consumes 1,631 MB and 159 MB of memory.

5 Conclusion

This paper explored the problem of mining periodic high-utility itemsets (PHUIs). An efficient algorithm named PHM (Periodic High-utility itemset Miner) was proposed to efficiently discover PHUIs using novel minimum and average periodicity measures. An extensive experimental study with real-life datasets has shown that PHM can be more than two orders of magnitude faster than FHM, and discover more than two orders of magnitude less patterns by filtering non periodic HUIs. The source code of the PHM algorithm and datasets can be downloaded as part of the SPMF open source data mining library [5] http://www.philippe-fournier-viger.com/spmf/.

For future work, we will consider designing alternative algorithms to mine PHUIs. In particular, interesting possibilities are to integrate length constraints in PHUI mining [6], and to use a database projection and merging approach as proposed in the EFIM [17] algorithm for HUI mining. Lastly, more complex types of patterns such as periodic sequential rules could also be explored [16].

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proceedings of the International Conference Very Large Databases, pp. 487–499 (1994)
2. Amphawan, K., Lenca, P., Surarerks, A.: Mining top-k periodic-frequent pattern from transactional databases without support threshold. In: Proceedings of the 3rd International Conference on Advances in Information Technology, pp. 18–29 (2009)
3. Amphawan, K., Surarerks, A., Lenca, P.: Mining periodic-frequent itemsets with approximate periodicity using interval transaction-ids list tree. In: Proceeding of the 2010 Third International Conference on Knowledge Discovery and Data Mining, pp. 245–248 (2010)
4. Fournier-Viger, P., Wu, C.-W., Zida, S., Tseng, V.S.: FHM: faster high-utility itemset mining using estimated utility co-occurrence pruning. In: Proceedings of the 21st International Symposium on Methodologies for Intelligent Systems, pp. 83–92 (2014)
5. Fournier-Viger, P., Gomariz, A., Gueniche, T., Soltani, A., Wu, C., Tseng, V.S.: SPMF: a Java open-source pattern mining library. J. Mach. Learn. Res. (JMLR) **15**, 3389–3393 (2014)
6. Fournier-Viger, P., Lin, C.W., Duong, Q.-H., Dam, T.-L.: FHM+: faster high-utility itemset mining using length upper-bound reduction. In: Proceedings of the 29th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, p. 12. Springer (2016)
7. Lan, G.C., Hong, T.P., Tseng, V.S.: An efficient projection-based indexing approach for mining high utility itemsets. Knowl. Inform. Syst. **38**(1), 85–107 (2014)
8. Kiran, R.U., Reddy, P.K.: Mining rare periodic-frequent patterns using multiple minimum supports. In: Proceedings of the 15th International Conference on Management of Data (2009)
9. Kiran, R.U., Kitsuregawa, M., Reddy, P.K.: Efficient discovery of periodic-frequent patterns in very large databases. J. Syst. Softw. **112**, 110–121 (2015)
10. Song, W., Liu, Y., Li, J.: BAHUI: fast and memory efficient mining of high utility itemsets based on bitmap. Int. J. Data Warehous. Min. **10**(1), 1–15 (2014)
11. Liu, M., Qu, J.: Mining high utility itemsets without candidate generation. In: Proceedings of the 22nd ACM International Conference Information and Knowledge Management, pp. 55–64 (2012)

12. Liu, Y., Liao, W., Choudhary, A.: A two-phase algorithm for fast discovery of high utility itemsets. In: Proceedings of the 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 689–695 (2005)
13. Surana, A., Kiran, R.U., Reddy, P.K.: An efficient approach to mine periodic-frequent patterns in transactional databases. In: Proceedings of the 2011 Quality Issues, Measures of Interestingness and Evaluation of Data Mining Models Workshop, pp. 254–266 (2012)
14. Tanbeer, S.K., Ahmed, C.F., Jeong, B.S., Lee, Y.K.: Discovering periodic-frequent patterns in transactional databases. In: Proceedings of the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 242–253 (2009)
15. Tseng, V.S., Shie, B.-E., Wu, C.-W., Yu, P.S.: Efficient algorithms for mining high utility itemsets from transactional databases. IEEE Trans. Knowl. Data Eng. **25**(8), 1772–1786 (2013)
16. Zida, S., Fournier-Viger, P., Wu, C.-W., Lin, J.C.W., Tseng, V.S.: Efficient mining of high utility sequential rules. In: Proceedings of the 11th International Conference Machine Learning and Data Mining, pp. 1–15 (2015)
17. Zida, S., Fournier-Viger, P., Lin, J.C.-W., Wu, C.-W., Tseng, V.S.: EFIM: a highly efficient algorithm for high-utility itemset mining. In: Proceedings of the 14th Mexican International Conference on Artificial Intelligence, pp. 530–546