Discovering Partial Periodic High Utility Itemsets in Temporal Databases

T. Yashwanth Reddy[1], R. Uday Kiran[⋆2,3], Masashi Toyoda[2], P. Krishna Reddy[1], and Masaru Kitsuregawa[2,4]

[1] International Institute of Information Technology, Hyderabad, India
[2] The University of Tokyo, Tokyo, Japan
[3] National Institute of Information and Communications Technology, Tokyo, Japan
[4] National Institute of Informatics, Tokyo, Japan
yashwanth.t@research.iiit.ac.in,
{uday_rage,toyoda,kitsure}@tkl.iis.u-tokyo.ac.jp and pkreddy@iiit.ac.in

Abstract. High Utility Itemset Mining (HUIM) is an important model with many real-world applications. Given a (non-binary) transactional database and an external utility database, the aim of HUIM is to discover all itemsets within the data that satisfy the user-specified *minimum utility* (*minUtil*) constraint. The popular adoption and successful industrial application of HUIM has been hindered by the following two limitations: (*i*) HUIM does not allow external utilities of items to vary over time and (*ii*) HUIM algorithms are inadequate to find recurring customer purchase behavior. This paper introduces a flexible model of Partial Periodic High Utility Itemset Mining (PPHUIM) to address these two problems. The goal of PPHUIM is to discover only those interesting high utility itemsets that are occurring at regular intervals in a given temporal database. An efficient depth-first search algorithm, called PPHUI-Miner (Partial Periodic High Utility Itemset-Miner), has been proposed to enumerate all partial periodic high-utility itemsets in temporal databases. Experimental results show that the proposed algorithm is efficient.

Keywords: Data mining · pattern mining · utility itemset mining · periodic itemsets.

1 Introduction

High Utility Itemset Mining (HUIM) is an important model in data mining. HUIM algorithms discover all interesting itemsets whose *utility* (profit) in a transactional database is no less than the user-specified *minimum utility (minUtil)* constraint. The utility of an *itemset* is the summation of its utilities in all the transactions. The classic application of HUIM is market-basket analysis. HUIM has many other applications, such as website click stream analysis, cross-marketing and bio-medical applications [3]. HUIM has also inspired several other

⋆ Corresponding author

important data mining tasks such as high-utility occupancy pattern mining [4] and high-utility periodic pattern mining [1].

The popular adoption and successful industrial application of HUIM has been hindered by the following two obstacles: (*i*) Most previous studies on HUIM implicitly assume that the external utilities of the items do not change over time in the entire database. However, this is the seldom in real-world applications. In many applications, items' external utilities can vary with respect to time. For example, the prices of items in an eCommerce application can raise and/or fall depending on supply and demand. (*ii*) In many applications, high utility itemsets that are occurring at regular intervals can provide useful information to the users. For instance, in an eCommerce store, customers buy certain items (e.g. diapers and soaps) on a weekly or monthly basis. The knowledge pertaining to such periodically purchased high utility itemsets can facilitate an eCommerce application to improve its sales. Unfortunately, most studies on HUIM fail to discover such periodically occurring high utility itemsets in the data.

This paper makes an effort to address the above mentioned two issues. This paper introduces a novel model of Partial Periodic High Utility Itemset (PPHUI) in temporal databases. A temporal database not only facilitate multiple transactions to appear the same timestamp, but also facilitates irregular time gaps between the consecutive transactions. Partial Periodic High Utility Itemset Mining (PPHUIM) allows items' external utility values to vary overtime. Thus, addressing the first obstacle of HUIM. The PPHUIM tries to address the second obstacle of HUIM by finding partial periodically occurring high utility itemsets in temporal databases. A fast algorithm, called Partial Periodic High Utility Itemset-Miner (PPHUI-Miner), has been introduced to discover all PPHUIs by proposing new pruning techniques. Experimental results demonstrate that PPHUI-Miner is not only memory and runtime efficient, but also highly scalable as well.

The rest of the paper is organized as follows. Related work is presented in Section 2. Section 3 introduces the proposed model of PPHUIM. The proposed is presented in Section 4. Experimental results are reported in Section 5. Section 6 provides conclusions.

2 Related Work

High utility itemset mining: Yao et al. [12] described HUIM by taking into account the importance of items and their occurrence $frequency$ in every transaction. Since then, several algorithms have been proposed to discover high utility itemsets in transactional databases [2, 7–9, 11] and sequence databases [14]. To circumvent the fact that the utility is not anti-monotonic and to find all high utility itemsets, several HUIM algorithms (e.g. Two-Phase [9] and UP-Growth+) have employed *Transaction Weighted Utilization* (TWU) to reduce the search space. The TWU measure represents an upper bound on the *utility* of itemsets. Recently, alorithms like EFIM [13] introduced by proposing tighter measures to calculate upper bound on the utility of itemsets than TWU.

Periodic high utility itemset mining: Tanbeer et al. [10] have introduced a model to find periodic-frequent itemsets in transactional databases. Philippe et al. [1] have extended the model [10] to discover full periodic high utility itemsets in a transactional database (i.e., a database in which transactions occur at a fixed time interval). This model discovers all periodic itemsets within the transactional database that satisfy the user specified minimum utility ($minUtil$), minimum average periodicity ($minAvgPer$), maximum average periodicity ($maxAvgPer$), minimum period ($minPer$) and maximum period ($maxPer$). This model suffers from the following limitations: (i) If an itemset has one instance where *period* (or inter-arrival time) is more than the user-specified $maxPer$, the corresponding itemset is considered as an uninteresting itemset. (ii) This model assumes time gap between two consecutive transactions is constant, which is not the case in real-world databases and It requires too many input parameters from the user.

A model has been proposed in [6] to find partial periodic itemsets in temporal databases. It can overcome limitations of model proposed in [1]. However, the model [6] disregards the importance of the items and their occurrence $frequency$ in every transaction.

The proposed model of PPHUI mining does not suffer from any of the above mentioned limitations. A part from extracting PPHUI from a given trasanactional database, the proposed model is different from the model proposed in [1] as that model employs different measures to find periodic high utility itemsets.

3 Proposed Model

Let $I = \{i_1, i_2, \cdots, i_m\}$, $m \geq 1$, be a set of items. Let $X \subseteq I$ be an itemset. An itemset containing k items is known as k-itemset. A transaction $T_{tid} = (tid, ts, Y)$ is a triplet, where $tid \in R^+$ represents the transactional identifier, $ts \in R^+$ represents the timestamp of corresponding transaction and $Y \subseteq I$ is an itemset. A temporal database, denoted as TDB, represents a set of transactions. That is, $TDB = \{T_1, T_2, \cdots, T_n\}, 1 \leq n$. Let $p(i_j, tid)$ denote the **external utility** of an item $i_j \in I$ in a transaction whose transaction identifier is tid. Let $P(i_j) = \{p(i_j, 1),\ p(i_j, 2), \cdots,\ p(i_j, n)\}$ denote the set of all external utility values of i_j in the data. The (external) **utility database**, UD, is the set of external utility values of all items in I. That is, $UD = \bigcup_{i_j \in I} P(i_j)$. Every item $i_j \in T_{tid}$ has a positive number $q(i_j, tid)$, called its **internal utility**. The internal utility of an item generally represents its $frequency$ in a transaction and external utility represents cost/profit of item in a transaction.

Example 1. Let $I = \{a, b, c, d, e, f, g, h, i, j\}$ be the set of items. The set of items 'd' and 'f', i.e., $\{d, f\}$ (or df, in short) is an itemset. This itemset contains two items. Therefore, it is a 2-itemset. A temporal database generated from I is shown in Table 1. This database contains 8 transactions. The minimum and maximum timestamps of the transactions in this database are 1 and 12, respectively. It can be observed that temporal databases not only allow multiple transactions

Table 1. Temporal database

tid	ts	items
1	1	$(a,1),(b,2),(c,1)$
2	3	$(a,2),(b,2),(e,2),(h,1)$
3	4	$(c,1),(d,3),(f,2)$
4	6	$(b,1),(d,2),(e,3),(f,1),(g,2),(h,3)$
5	7	$(c,3),(f,1),(g,1)$
6	7	$(i,1),(j,3)$
7	9	$(a,1),(b,1),(d,2),(f,1),(g,2)$
8	12	$(c,3),(d,1),(e,1),(f,2),(g,2)$

Table 2. External utility database

tid	a	b	c	d	e	f	g	h	i	j
1	200	100	50	0	0	0	0	0	0	0
2	50	100	0	0	100	0	0	100	0	0
3	0	0	200	200	0	200	0	0	0	0
4	0	200	0	200	150	300	100	200	0	0
5	0	0	100	0	0	150	50	0	0	0
6	0	0	0	0	0	0	0	0	40	20
7	150	300	0	200	0	300	200	0	0	0
8	0	0	50	200	300	50	200	0	0	0

to share a common timestamp, but also encourage irregular time gaps between the consecutive transactions. Thus, a temporal database generalizes a transactional database by taking into account the temporal occurrence information of the transaction. Table 2 shows the external utilities (or prices/profit) of all items in every transaction. Let the currency of these prices be Japanese Yen (¥). The external utility of an item d in the third transaction, i.e., $p(d,3) = 200$¥. The internal utility of an item d in the third transaction T_3, i.e., $q(d,3) = 3$.

Definition 1. *(**Utility of an item in a transaction**). The utility of an item i_j in a transaction T_{tid} denoted as $u(i_j, T_{tid}) = p(i_j, T_{tid}) \times q(i_j, T_{tid})$.*

Definition 2. *(**Utility of an itemset in a transaction**) The utility of an itemset X in a transaction T_{tid}, denoted as $u(X, T_{tid}) = \Sigma_{i \in X} u(i, T_{tid})$.*

Definition 3. ***(Utility of an itemset in a database)** The utility of an itemset X in the database TDB, denoted as $u(X) = \Sigma_{T_{tid} \in g(X)} u(X, T_{tid})$, where $g(X)$ is the set of transactions containing X.*

Example 2. Continuing the previous example, the utility of 'd' in third transaction T_3, i.e., $u(d, T_3) = p(d, T_3) \times q(d, T_3) = 200 \times 3 = 600$¥. The utility of itemset df in T_3, $u(df, T_3) = u(d, T_3) + u(f, T_3) = 600$¥ $+ 400$¥ $= 1000$¥. In Table 1, the itemset df has appeared in the transactions T_3, T_4, T_7 and T_8. Therefore, $g(x) = \{T_3, T_4, T_7, T_8\}$. The *utility* of df in each of these three transactions: $u(df, T_3) = 1000$¥, $u(df, T_4) = 700$¥, $u(df, T_7) = 700$¥ and $u(df, T_8) = 300$¥. Therefore, the *utility* of df in the database, $u(df) = 2700$¥.

Definition 4. *(**Periodic appearance of X**.) Let $TS^X = \{ts_a^X, ts_b^X, \cdots, ts_c^X\}$, $ts_{min} \leq ts_a^x \leq ts_b^x \leq ts_c^x \leq ts_{max}$, be an ordered list of timestamps in which X appeared in TDB. The terms ts_{min} and ts_{max} represent the minimal and maximal timestamps in TDB. Let $ts_j^X, ts_k^X \in TS^X$, $ts_{min} \leq ts_j^X \leq ts_k^X \leq ts_{max}$, denote any two consecutive timestamps in TS^X. The time difference between ts_k^X and ts_j^X is referred to an **inter-arrival time** of X, and denoted as iat_p^X, $p \geq 1$. That is, $iat_p^X = ts_k^X - ts_j^X$. Let $IAT^X = \{iat_1^X, iat_2^X, ..., iat_{|TS^X|-1}^X\}$, be the list of all inter-arrival times of X in TDB. An inter-arrival time of X is said to be **periodic** (or interesting) if it is no more than the user-specified*

maximum-inter arrival time (maxIAT). That is, an $iat_k^X \in IAT^X$ *is said to be* ***periodic*** *if* $iat_k^X \leq maxIAT$.

Example 3. In Table 1, the itemset *df* has appeared in the transactions T_3, T_4, T_7 and T_8. Therefore, the set of timestamps of these four transactions, i.e., $TS^{df} = \{4, 6, 9, 12\}$. The inter-arrival times of '*df*' are: $iat_1^{df} = 6-4 = 2$, $iat_2^{df} = 9-6 = 3$, $iat_3^{df} = 12-9 = 3$. Thus, $IAT^{df} = \{iat_1^{df}, iat_2^{df}, iat_3^{df}\} = \{2, 3, 3\}$. If the user-specified $maxIAT = 3$, then iat_1^{df} is considered interesting (or periodic) occurrence of *df* within the database because $iat_1^{df} \leq maxIAT$. Similarly, iat_2^{df} and iat_3^{df} are also periodic occurrences of *df*.

Definition 5. ***(Periodic-Support of itemset).*** *Let* $\overline{IAT^X} \subseteq IAT^X$ *be the set of all inter-arrival times that have value no more than maxIAT. That is,* $\overline{IAT^X} \subseteq IAT^X$ *such that if* $\exists iat_k^X \in IAT^X : iat_k^X \leq maxIAT$, *then* $iat_k^X \in \overline{IAT^X}$. *The periodic-support of X, denoted as* $PS(X) = |\overline{IAT^X}|$.

Example 4. Continuing with the previous example, $\overline{IAT^{df}} = \{iat_1^{df}, iat_2^{df}, iat_3^{df}\}$. Therefore, the *periodic support* of '*df*', i.e., $PS(df) = |\overline{IAT^{df}}| = 3$. In other words, the itemset '*df*' has appeared 3 times periodically within the data.

The *periodic-support*, as defined above, determines the number of periodic occurrences of an *itemset* in the database. An inter-arrival time of an itemset can be expressed in percentage of $(ts_{max} - ts_{min})$. The *periodic-support* of an itemset also can be expressed in percentage of $|TDB|-1$, where $|TDB|-1$ represents the maximum *periodic-support* an itemset can have in the database.

Definition 6. ***(Partial Periodic High Utility Itemset X.)*** *An itemset X is a Partial Periodic High Utility Itemset (PPHUI) if* $u(X) \geq minUtil$ *and* $PS(X) \geq minPS$, *where minUtil and minPS represent the user-specified* minimum utility *and* minimum periodic-support, *respectively.*

Example 5. If the user-specified $minUtil = 1500$¥, $maxIAT = 6$ and $minPS = 2$, then the itemset '*df*' is a PPHUI because $u(df) \geq minUtil$ and $PS(df) \geq minPS$. All PPHUIs generated from Table 1 are shown in Table 4.

Problem Statement: Given a temporal database (TDB), an external utility database (UD) and the user-specified *minUtil*, *maxIAT* and *minPS*, the problem of finding PPHUIs involve discovering all itemsets in *TDB* whose *utility* and *periodic-support* is no less than the user-specified *minUtil* and *minPS*, respectively.

4 Proposed Approach

The problem is to develop an efficient approach for discovering partial periodic high utility itemsets (PPHUIs) in Temporal Database subject to *minUtil*, *minPS* and *maxIAT* constraints. Given n data items, a naïve way to find

Table 3. TWU values of items in Table 1

item	***f***	***d***	***g***	***b***	***e***	***c***	***a***	h	i	j
TWU	6750	6250	5550	4950	3900	3300	2800	2750	100	100
PS	4	3	3	3	2	3	2	1	0	0

Table 4. PPHUIs generated from Table 1 at $minUtil = 2000, PS = 2, maxIAT = 6$

Itemset	d	f	g	ab	cf	dg	df	fg	dfg
Utility	1600	1250	1250	1150	1300	2200	2700	2100	2900
PS	3	4	3	2	2	2	3	3	3

PPHUIs is to mine set of all possible $2^n - 1$ combinations of items and test for $minUtil$, $minPS$ and $maxIAT$ constraints. Notably, such an approach suffers from exponential complexity. The basic idea is to define pruning techniques based on Transaction Weighted Utilization (TWU), Periodic Support (PS) and Remaining Utility and proposed efficient approach to mine PPHUI. We briefly explain these techniques and discuss the proposed approach.

i. Pruning using TWU:

We carry out the pruning based on TWU [9]. The notion of TWU is defined as follows.

Definition 7. *(**Transaction Weighted Utilization (TWU)**) The transaction utility (TU) of a transaction T_{tid} is the sum of the utility of all items in T_{tid}. i.e. $TU(T_{tid}) = \Sigma_{x \in T_{tid}} u(x, T_{tid})$. The transactional-weighted utilization (TWU) of an itemset X is defined as the sum of the transaction utility of transactions containing X, i.e. $TWU(X) = \Sigma_{T_c \in g(X)} TU(T_c)$.*

Example 6. Consider the first transaction in Table 1. The *transaction utility* of T_1, denoted as $TU(T_1)$, is the total revenue generated by all its items. That is, $TU(T_1) = u(a, 1) + u(b, 2) + u(c, 1) = 200 + 200 + 50 = 450$¥. In other words, the first transaction has generated the revenue of 450¥, Similarly, the *transaction utility* of T_2, T_3, T_4, T_5, T_6, T_7 and T_8 are 600, 1200, 2150, 500, 100, 1750 and 1150 respectively. Consider the item 'd,' which is appearing in the transactions T_3, T_4, T_7 and T_8. The TWU of d, i.e., $TWU(d) = TU(T_3) + TU(T_4) + TU(T_7) + TU(T_8) = 6250$¥.

The pruning rule based on TWU is as follows. It can be observed that the TWU of item conveys the crucial information that it is equivalent to atmost utility that an item can generate by combining with other items in the database. TWU measure can be used to identify the items, whose supersets may generate PPHUIs. We ignore the extensions of items $i_j \in I$ whose $TWU(i_j) < minUtil$.

ii. Pruning using PS:

We prune the itemsets based on the value of PS. *Periodic-Support* has **anti-monotonic property** that is an itemset cannot have PS greater than PS

of its subsets. So, we can ignore extensions of those items/itemsets X, whose $PS(X) < minPS$.

iii. Pruning using Remaining Utility:

We carry out the pruning based on the notion of *Remaining Utility*. We define the notion of *Remaining Utility* and define the notion of *utility list*.

Definition 8. *(**Remaining utility**). Let $\succ$ be any total order on items from I and X be the itemset. The remaining utility of X in a transaction T_{tid} is defined as $rU(X, T_{tid}) = \Sigma_{i \in T_{tid} \wedge i \succ x \forall x \in X} u(i, T_{tid})$.*

Definition 9. *(**Utility-list**) Let $\succ$ be any total order on items from I. The utility-list of an itemset X in a database D is denoted as $UL(X)$ and defined as a set of tuples such that there is a tuple $< tid, ts, iutil, rutil >$ for each transaction T_{tid} containing X. The iutil element of a tuple is utility of X in T_{tid}. i.e., u(X,T_{tid}). The rutil element of a tuple is the remaining utility (see Definition 8).*

Example 7. For this example $\succ$ be lexicographical order i.e. ($i \succ h \succ g \succ f \succ e \succ d \succ c \succ b \succ a$) Remaining utility of df in T_4 is $rU(df, T_4) = u(g, T_4) + u(h, T_4) = 2 \times 100 + 3 \times 200 = 800$.

For pruning, we use **Remaining utility** measure to overestimate the utility value of *itemset*. Let X be an itemset. If $\Sigma iutil + \Sigma rutil < minUtil$, where $iutil$, $rutil \in ul(X)$, X and its extensions are low utility. So, such patterns can be pruned. The proof that the sum of $iutil$ and $rutil$ values of utility list an itemset X is an upper bound on the utility of X and its extensions is provided in [8].

The proposed **PPHUI-Miner** employs depth-first search of set enumeration tree and prunes patterns based on preceding pruning techniques.

Algorithm 1 PPHUI-Miner

1: **Input**: TDB: a temporal database; UD: a external utility database; $minUtil$: a user-specified minimum utility constraint; $maxIAT$: a user-specified period constraint; $minPS$: a user-specified periodic-support constraint.
2: **Output**: A set of partial periodic high-utility itemsets.
3: Let α denote the itemset that needs to be extended. Initially, set $\alpha = \emptyset$;
4: Scan TDB to compute $TWU(\{i_j\})$, $PS(\{i_j\})$ for each items $i_j \in I$;
5: $I^* = \{i_j | i_j \in I \wedge TWU(i_j) \geq minUtil \wedge PS(i_j) \geq minPS\}$;
6: Let us call I^* as candidate items;
7: Let $\succ$ be the total order of TWU descending values on candidate items;
8: Scan TDB to build the utility list(UL) of each item $i_j \in I^*$;
9: $Primary(\alpha) = \{i_j | i_j \in I^* \wedge \forall x \in \alpha, i_j \succ x\}$;
10: Search (UL, α, $Primary(\alpha)$, $minUtil$, $maxIAT$, $minPS$);

The Approach: PPHUI-Miner presented in Algorithms 1, 2. We first scan the database to measure TWU and PS values for all items within the database.

Table 3 shows the TWU amd PS values determined for all items after scanning the database. Next, we prune the items in the list that have PS value less than $minPS$ and/or TWU value less than $minUtil$. The remaining items in the list are considered as **candidate items** and sorted in TWU descending order of items. After finding **candidate items** and establishing $\succ$ total order (i.e., TWU descending order of items), the utility list (refer Definition 9) by scanning the database second time. The $Primary(\alpha)$ contains the candidate items, which are $\succ$ than every item in α.

After building the utility lists of candidate items, we call recursive search with α and UL utility list of candidate items. Next, we expand search by combining α with $Primary(\alpha)$ one by one using DFS technique. If $i_x \in Primary(\alpha)$, we build utility list of $\beta(\alpha \cup i_x)$. We check utility and periodic support of β from the above utility list. Then we have two cases: (i) if β is PPHUI, then $Primary(\beta)$ is generated and β is further extended by calling recursive search (ii) if β is not PPHUI, it may fail to satisfy either $minPS$ or $minUtil$ values. In the former case (i.e., when β fails to satisfy $minPS$), we stop performing depth-first search on α. In the latter case (i.e., when β fails to satisfy only $minUtil$), we calculate its remaining utility value. If this value is greater than $minUtil$, we continue exploring β same as in first case. If remaining utility of β is less than $minUtil$, then we stop exploring that branch in the DFS tree.

Algorithm 2 The search procedure

1: **Input**: α: an itemset; UL: utility lists of candidate items; $UL(\alpha)$: utility list of α; $Primary(\alpha)$: *Extension items of* α; $minUtil$; $maxIAT$; $minPS$.
2: **Output**: A set of periodic high-utility itemsets.
3: **for** $\forall$ itemsets β=$\alpha \cup i_j$, $i_j \in Primary(\alpha)$; **do**
4: Calculate utility list of β from utility lists of α and i_j;
5: Calculate utility and periodic support of β from utility list above;
6: **if** $U(\beta) + rU(\beta) \geq minUtil \wedge PS(\beta) \geq minPS$ **then**
7: **if** $U(\beta) \geq minUtil$ **then**
8: Output β;
9: **end if**
10: generate itemset $Primary(\beta)$;
11: Search(β, UL, $UL(\beta)$, $Primary(\beta)$, $minUtil$, $maxIAT$, $minPS$);
12: **end if**
13: **end for**

5 Experimental Results

Since there exists no algorithm to find PPHUIs in temporal databases, we only evaluate the proposed PPHUI-Miner algorithm using both synthetic and real-world databases. Please note that we are not comparing the proposed PPHUI-Miner algorithm against the Periodic High Utility Mining (PHM) algorithm. It is because PHM employs different measures to find interesting itemsets.

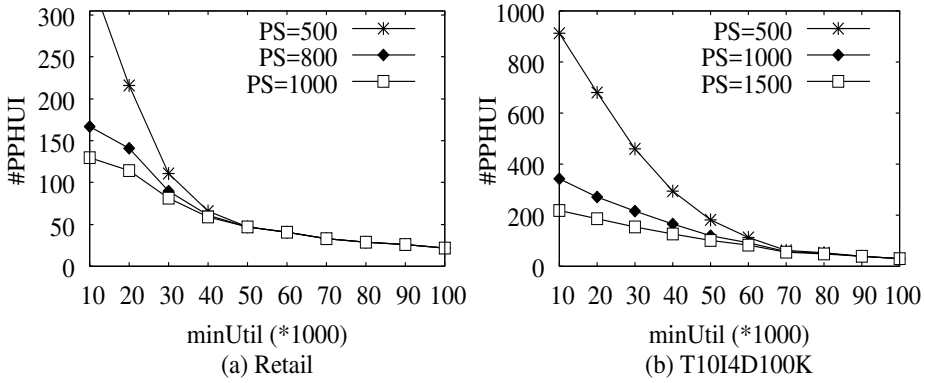

Fig. 1. PPHUI generated in Retail and T10I4D100k databases.

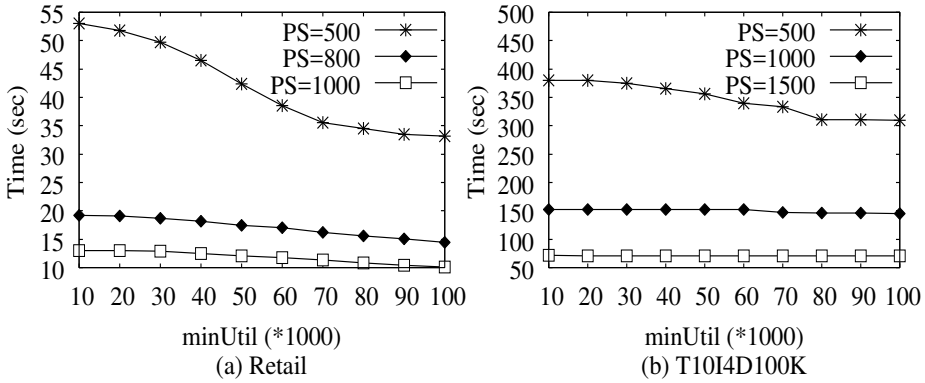

Fig. 2. Time taken by PPHUI-miner for Retail and T10I4D100k databases.

The algorithms, PHM and PPHUI-Miner, were written in C++ and executed on i5 1.5 GHz processor, with 16GB ram. The experiments have been conducted using both synthetic (T10I4D100K) and real-world (Retail) databases. The Retail and T10I4D100K databases are available on SPMF toolkit.

The $maxIAT$ value for Retail database is fixed at 500 and for T10I4D100K database is fixed at 1000. We are not reporting results by varying $maxIAT$ value due page limitation. But in general, we observed that increase in $maxIAT$ increases number of PPHUIs generated [5].

Fig. 1(a) and Fig. 1(b) show the number of PPHUIs generated by PPHUI-Miner in different databases at different $minUtil$ and $minPS$ values. It can be observed that increase in $minUtil$ and/or $minPS$ results in the decrease of PPHUIs as many itemsets fail to satisfy the increased $minUtil$ and/or $minPS$ values. Fig. 2(a) and Fig. 2(b) show the runtime requirements of PPHUI-Miner in different databases at different $minUtil$ amd $minPS$ values. It can be observed that increase in $minUtil$ and/or $minPS$ results in the decrease of runtime for PPHUI-Miner algorithm. It is because many itemsets fail to satisfy the increased

minUtil and/or *minPS* values. Similar behaviour is observed in case of memory consumption, but due page limitation we are not including graphs of memory consumption. Overall, it can be observed from the results that PPHUI-Miner algorithm can efficiently discover PPHUIs in very large databases even at low *minUtil* and *minPS* values.

6 Conclusions and Future work

In this paper, we have studied the problem of finding partial periodic high utility itemsets in temporal databases. A fast algorithm has also been presented to find all PPHUIs. The proposed approach employs pruning techniques to improve efficiency (or computational cost). As a part of future work, we looking to develop more efficient algorithms to discover Partial Periodic High Utility itemsets in other databases like uncertain database.

References

1. Fournier-Viger, P., Lin, J.C.W., Duong, Q.H., Dam, T.L.: Phm: Mining periodic high-utility itemsets. In: Industrial Conference on Data Mining. pp. 64–79 (2016)
2. Fournier-Viger, P., Wu, C.W., Zida, S., Tseng, V.S.: Fhm: Faster high-utility itemset mining using estimated utility co-occurrence pruning. In: ISMIS. pp. 83–92 (2014)
3. Gan, W., Lin, J.C., Fournier-Viger, P., Chao, H., Hong, T., Fujita, H.: A survey of incremental high-utility itemset mining. Wiley Interdiscip. Rev. Data Min. Knowl. Discov. **8**(2) (2018)
4. Gan, W., Lin, J.C.W., Fournier-Viger, P., Chao, H.C., Philip, S.Y.: Huopm: High-utility occupancy pattern mining. IEEE Transactions on Cybernetics (2019)
5. Kiran, R.U., Reddy, T.Y., Fournier-Viger, P., Toyoda, M., Reddy, P.K., Kitsuregawa, M.: Efficiently finding high utility-frequent itemsets using cutoff and suffix utility. In: PAKDD (2019)
6. Kiran, R.U., Shang, H., Toyoda, M., Kitsuregawa, M.: Discovering partial periodic itemsets in temporal databases. In: Proc of the 29th SSDBM. p. 30. ACM (2017)
7. Lan, G.C., Hong, T.P., Tseng, V.S.: An efficient projection-based indexing approach for mining high utility itemsets. KAIS **38**(1), 85–107 (2014)
8. Liu, M., Qu, J.: Mining high utility itemsets without candidate generation. In: Proc of the 21st ACM CIKM. pp. 55–64. ACM (2012)
9. Liu, Y., Liao, W.k., Choudhary, A.: A two-phase algorithm for fast discovery of high utility itemsets. In: PAKDD. pp. 689–695 (2005)
10. Tanbeer, S.K., Ahmed, C.F., Jeong, B.S., Lee, Y.K.: Discovering periodic-frequent patterns in transactional databases. In: PAKDD. pp. 242–253. Springer (2009)
11. Tseng, V.S., Shie, B.E., Wu, C.W., Yu, P.S.: Efficient algorithms for mining high utility itemsets from transactional databases. TKDE **25**(8), 1772–1786 (2013)
12. Yao, H., Hamilton, H.J., Butz, C.J.: A foundational approach to mining itemset utilities from databases. In: SIAM. pp. 482–486 (2004)
13. Zida, S., Fournier-Viger, P., Lin, J.C.W., Wu, C.W., Tseng, V.S.: Efim: a fast and memory efficient algorithm for high-utility itemset mining. KAIS **51**(2), 595–625 (2017)
14. Zida, S., Fournier-Viger, P., Wu, C.W., Lin, J.C.W., Tseng, V.S.: Efficient mining of high-utility sequential rules. In: MLDM. pp. 157–171 (2015)