IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON UTILITY PATTERN MINING: THEORETICAL ANALYTICS AND APPLICATIONS

Received February 29, 2020, accepted March 7, 2020, date of publication March 23, 2020, date of current version April 21, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2982415

Incrementally Updating the Discovered High Average-Utility Patterns With the Pre-Large Concept

JIMMY MING-TAI WU[1], QIAN TENG[1], JERRY CHUN-WEI LIN[2], (Senior Member, IEEE), AND CHIEN-FU CHENG[3]

[1]School of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China
[2]Department of Computer Science, Electrical Engineering and Mathematical Sciences, Western Norway University of Applied Sciences, 5063 Bergen, Norway
[3]Department of Computer Science and Information Engineering, Tamkang University, New Taipei City 25137, Taiwan

Corresponding author: Jerry Chun-Wei Lin (jerrylin@ieee.org)

This work was supported by the Western Norway University of Applied Sciences, Bergen, Norway.

ABSTRACT High average-utility itemset mining (HAUIM) is an extension of high-utility itemset mining (HUIM), which provides a reliable measure to reveal utility patterns by considering the length of the mined pattern. Some research has been conducted to improve the efficiency of mining by designing a variety of pruning strategies and effective frameworks, but few works have focused on the maintenance algorithms in the dynamic environment. Unfortunately, most existing works of HAUIM still have to rescan databases multiple times when it is necessary. In this paper, the pre-large concept is used to update the discovered HAUIs in the newly inserted transactions and reduce the time of the rescanning process. To further improve the performance of the developed algorithm, two new upper-bounds are also proposed to decrease the number of candidates for HAUIM. Experiments were performed to compare the previous Apriori-like method and the proposed APHAUP algorithm with the two new upper-bounds in terms of the number of maintenance patterns and runtime in several datasets. The experimental results show that the proposed APHAUP algorithm has excellent performance and good potential to be applied in real applications.

INDEX TERMS Pre-large, high average-utility itemset mining, dynamic database, lead partial upper bound, incremental.

I. INTRODUCTION

Knowledge discovery in database (KDD) [1]–[6] is a process for extracting practical, novel, and potentially useful knowledge from the original data while shielding users from the tedious details of the original data. Algorithms in KDD can be applied to many different applications, but in the KDD process, security [7]–[11] and optimization [12] are the primary considerations, which have also been considered as an emerging topics in recent decades. The Association rule mining (ARM) [1] was the first significant algorithm to identify frequent itemsets (FIs) based on minimum support and generate association rules (ARs) with the minimum confidence threshold. The first Apriori method utilizes the generate-and-test approach for mining ARs through a level-wise process. The Apriori algorithm [1] follows the downward closure property (DC), and thus many unpromising candidates can be ignored and reduced. To accelerate the mining performance for discovering the set of FIs, some efficient data structures have been studied and proposed, such as FP-tree and FP-growth mining [13]. Moreover, many extensions have provided the information needed in different knowledge domains, like HUIM [14]–[16] or HAUIM [17], [18], based on the Apriori-like DC property. Besides, the existing algorithms have focused on processing the static databases. When the capacity of the database is modified (such as transaction insertion), the discovered knowledge becomes useless, and the batch model is then processed to process the entire database to retrieve the required information.

However, frequent itemset mining (FIM) only considers the frequency of an item and assumes that all items occur at most once in a transaction. Thus, FIM only reflects whether an item occurs in a transaction. Many interesting factors, such as weight and the unit profit of the items, are not considered in FIM. High-utility itemset mining (HUIM) [14], [16], [19] has been considered to be an effective decision-making

The associate editor coordinating the review of this manuscript and approving it for publication was Philippe Fournier-Viger.

method for displaying profitable products and itemsets. That is, the quantity and utility of the itemsets are concerned to reveal the high-utility itemsets (HUIs). An item(set) is called a HUI when its utility exceeds the predefined minimum utility threshold. However, HUIMs is not a trivial task compared to the traditional FIM since it does not hold the anti-monotonic property. In other words, the utility of an itemset may be greater, equal, or smaller than the utility of its supersets. To overcome this problem, Liu *et al.* subsequently implemented a transaction weighted utilization (TWU) model [19] to retain the transaction-weighted downward closure (TWDC) property. Therefore, it can further reduce the search space for revealing HUIs. It first discovers the set of the high transaction-weighted utilization itemsets (HTWUIs), where each itemset in the HTWUIs holds the upper bound utility value on the pattern by ensuring the correctness and completeness of the final HUIs. Furthermore, the traditional TWU model relies on the generate-and-test approach; thus a huge amounts of candidates is produced and evaluated. To solve this limitation by maintaining the promising 1-itemsets in the tree structure, a high-utility pattern (HUP)-tree [20] was proposed by Lin *et al.* to decrease the computational costs. Then, the UP-growth+ approach [21] was introduced with the UP-tree structure to efficiently reveal the HUIs. Liu *et al.* [22] subsequently presented a new utility-list (UL) to extract the *k*-HUIs quickly based on the simple join operation. Several methods [15], [20], [23]–[26] have been extensively studied and discussed, and most of them focused on the TWU model to reveal HUIs.

Even though HUIM can find more information in the decision-making proces rather than the ARM, the utility of the pattern is increased along with the size of the itemset, especially when the length of the pattern is very long. For example, any combination with caviar can also be considered as a HUI in a supermarket, which is not a realistic case in practice. High average-utility itemset mining (HAUIM) [17] was investigated to evaluate high average-utility patterns by considering the length of an itemset. An itemset is considered to be a HAUI while the value of utility divides the length of the itemsets larger than the minimum utility threshold. This approach offers another way to fairly estimate the utility of the pattern. Hong *et al.* developed the TPAU algorithm [17] in a level-wise way to reveal satisfactory HAUIs. This approach applies the average-utility upper bound value (*auub*) to hold the downward closure property in order to discover the set of the high-average-utility-upper-bound itemsets (HAUUBIs). To improve mining performance, Lin *et al.* proposed the high average utility pattern (HAUP) tree structure [27] to maintain the 1-HAUUBIs in the compressed tree structure. An average-utility-list (AU-list) structure was also proposed through a simple join operation to generate the *k*-itemsets. Numerous extensions of HAUIM [27], [28] have also been explored and discussed to enhance the mining efficiency of HAUIs. The limitation of the existing HUIM or HAUIM algorithms were built on the maintenance of the static database, but the storage sizes have changed dynamically in realistic situations. In this article, we utilize the pre-large concept [29] for efficiently handling incremental mining while some transactions are inserted. Furthermore, we propose two new upper-bounds to improve mining efficiency. Several contributions of the developed algorithm are summarized below.

1) The pre-large Apriori-based APHAUI is presented for transaction insertion, which is used to update the discovered HAUIs effectively in the maintenance progress.
2) A strict upper-bound called the partial upper bound (*pub*) is designed to reduce the upper-bound utility value of the itemsets. Besides, a smaller upper-bound called *lpub* is also proposed to reduce the size of candidates in the search space.
3) An equation is specified here to ensure that an additional database scan is not unnecessarily performed. However, based on the designed APHAUI algorithm, the up-to-date HAUIs can still be maintained efficiently.
4) A linked-list structure is utilized in the designed algorithm to ensure that each transaction is scanned at most one time, thus reducing the number of multiple database scans in the maintenance progress.

II. LITERATURE REVIEW

This section introduces related works about HAUIM and dynamic data mining.

A. HIGH (AVERAGE-)utility ITEMSET MINING

The fundamental association-rule mining algorithm is based on two minimum thresholds to find the relationship between the itemsets, thereby mining the association rules, and the original algorithm is called Apriori [1]. However, Apriori only works with binary databases; therefore, other factors, such as weight, interestingness, importance, and quantity, are ignored in traditional ARM. To find more meaningful information from the discovered knowledge, HUIM (high-utility itemset mining) [16], [19], [30] was designed to evaluate two factors, unit profit and the number of items, to reveal useful and meaningful utility patterns in the database. An itemset is known to be a high utility itemset (HUI) if the utility of the itemset is not smaller than the pre-defined minimum utility threshold (count). Traditional HUIM cannot, however, solve the problem of combined explosions. Therefore, the search space is too large to find the required HUIs.

To solve this problem and limitation, a TWU model [19] that maintains the high transaction-weighted utilization itemsets (HTWUIs) is proposed to hold the downward closure property in the mining progress. A HTWUI contains the upper-bound utility on the itemset, and therefore the transaction-weighted downward closure (TWDC) property is maintained for the correctness and completeness of the discovered HUIs. Lin *et al.* then proposed a HUP-tree [20] to establish a compact tree structure for maintaining the 1-HTWUIs, which reduces the computational cost compared to the level-wise approach. Moreover, utility pattern

(UP)-growth+ [21] was proposed to apply the UP-tree structure and several pruning strategies for mining HUIs to accelerate the mining efficiency. Besides, a utility-list (UL)-structure and the HUI-Miner algorithm [15] were designed to generate the *k*-candidate HUIs easily through a straightforward join operation. Several HUIM [15], [22]–[26] algorithms have been extensively discussed, and most of them focused on the TWU model to reduce the number of candidates for mining the HUIs.

During HUIM, the value of the utility increases with the length of the itemset. Thus, any combination with an itemset with a very high utility value, for example, the caviar or diamond, is also considered as a HUI, which is not reasonable in real situations. High average-utility itemset mining (HAUIM) [17], [27] was extended from HUIM regarding the length of the itemset for evaluation. The first algorithm in HAUIM was TPAU [17], which uses *auub* model to keep the downward closure property and utilizes the Apriori-like approach to find the set of HAUIs in a level-wise manner. Lin *et al.* then presented a HAUP-tree [27] to keep the 1-itemsets in a compressed tree structure, thereby significantly reducing the computational cost. Several extensions of HAUIM [18], [28] were respectively studied and most of them relied on the *auub* model to find the set of HAUIs.

B. DYNAMIC DATA MINING

For traditional pattern mining, including ARM [1], HUIM [16], [19], and sequential pattern mining [4], they focused on mining the meaningful patterns from the static databases. In the case of the operations for the dynamic environment, such as insertion, deletion, or modification, for the batch-model algorithms, even if there is a small change (i.e., tiny transactions 1-5 are inserted into the database), they must rescan the updated database each time for pattern maintenance. This is a costly progress since the discovered knowledge in the previous stage cannot be re-used, and the database is needed for multiple database scans to obtain the updated information. Cheung *et al.* suggested the Fast UPdate (FUP) concept [31] to handle dynamic data mining for transaction insertion to reveal the up-to-date frequent itemsets. According to the discovered information in the original database and inserted transaction, it can be divided into four cases for later maintenance. This concept has been utilized in different domains for knowledge discovery, such as ARM [32], sequential pattern mining [33], HUIM [34], and HAUIM [35]. Although the FUP-based approach can handle dynamic data mining, it still needs to rescan the database in some cases, which is still inefficient for knowledge maintenance.

To better maintain the discovered knowledge and avoid the multiple database scans, Hong *et al.* developed the pre-large concept [29], [36], which is used to set up two thresholds for knowledge maintenance. These two thresholds (upper and lower) are used to maintain not only the large itemsets (the pattern count is no less than the default upper threshold) but also the pre-large itemsets (the pattern count is between the upper and lower thresholds). Thus, the pre-large itemsets will be maintained as the buffer to avoid multiple database scans. An equation is also defined to determine whether the number of the newly inserted transactions is less than the safety bound. Thus, some cases for database rescans can be avoided, but the completeness and correctness of the discovered knowledge can still be maintained. There is also an equation to determine whether the number of inserted transactions is less than the safety bound. Figure 1 shows the cases of the pre-large concept.

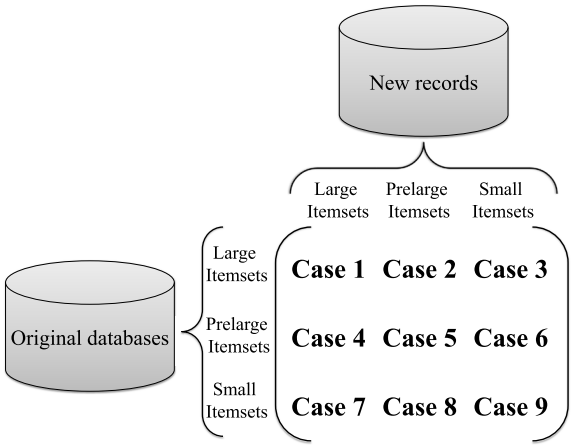


FIGURE 1. Nine cases of the pre-large concept.

In cases 1, 5, 6, 8 and 9, the final results cannot be affected. Furthermore, for cases 2 and 3, the amount of discovered knowledge can be reduced and some new knowledge for cases 4 and 7 may be arisen. The itemsets are easily handled for cases 2, 3 and 4 as the pre-large itemsets are kept. The safety bond (f) of the pre-large concept is defined below.

$$f = \lfloor \frac{(S_u - S_l) \times TU^D}{1 - S_u} \rfloor, \tag{1}$$

in which f represents the safety bound by evaluating two thresholds. S_l is set as the lower support threshold, S_u is set as the upper threshold. Moreover, TU^D is the transaction utility in D.

III. PROPOSED INCREMENTAL APRIORI FRAMEWORK FOR HAUI

This section describes the detailed algorithm of the proposed method (Apriori-based HAUP with pre-large concept, named APHAUP in this paper). The APHAUP is based on Apriori-like approach [1] with the new upper bound values to early prune the unpromising candidates and reduce the search space for revealing the satisfied itemsets in the updating progress. First, to reduce the size of promising itemsets, a strict upper-bound called partial upper bound (*pub*) is designed. In the traditional Apriori-like approach, the satisfied *k*-itemsets are then used to generate the (*k*+1)-itemsets as the candidate patterns in the level-wise manner. In APHAUP, an itemset called *pubi* (high *pub* itemset), which means its utility is larger than the *pub* is developed here.

Second, APHAUP selects a subset from *pubi*, called *lpubi* (lead-*pubi*), for later mining progress. In fact, APHAUP produces these two sets at the same time, but it applies a smaller upper-bound than *pub*. Therefore, *lpubis* is a subset for *pubis* in the proposed method. This smaller upper-bound is called *lpub* (lead-*pub*). The proposed *lpubi* can further reduce the size of the candidate itemsets effectively. Third, to make sure that each transaction will be scanned at most one time in each round, a linked list structure is proposed. Finally, the incremental updating process is proposed to maintain the discovered HAUIs, and the detailed definitions and algorithms for each part will be described in the following subsections.

A. PARTIAL UPPER BOUND, PUB AND HIGH PARTIAL UPPER BOUND ITEMSET, PUBI

For most existing algorithms [17], [27] in HAUIM, *auub* was widely applied to reveal HAUIs. The maximal utility from a transaction is definitely larger than the average utility from any itemset and its superset in this transaction. However, traditional *auub* is too large thus many unpromising itemsets are kept. This is because *auub* does not consider the itemset itself. In other words, it is assumed that there are two itemsets A and B in a transaction. The value of *auub* from this transaction for A and B are the same, even though $A \neq B$. The proposed *pub* was designed to solve this limitation and the detailed definition is given below.

Definition 1: An active item (*ai*) and active itemset (*ais*) in the proposed Apriori process are defined in the designed algorithm.

In Apriori-like approach [1], the discovered large k-itemsets are used to generate (k+1)-itemsets as the candidate patterns for evaluation. This means that if an item does not exist in one of the large k-itemsets, then it is definitely not be included in its supersets such as (k+1)-itemsets. In HAUIM, APHAUP follows the traditional Apriori-like approach, and in each iteration, it generates the candidate *pubis* where the length is n from the *pubis* included n-1 items.

Assuming that APHAUP handles the process to estimate *pubis* where the length is n, then an active item (*ai*) in this period indicates that this item at least exists in one of the *pubi* where the length is (n-1). The definition of ais_n (active itemset of the candidate *pubis* with the length is n) is defined as:

$$ais_n = \left\{ ai \in x \,\middle|\, x \in pubis^{n-1} \right\}, \tag{2}$$

where $pubis^{n-1}$ is the *pubis* in which the length is (n-1).

For example, assume that the process of APHAUP shows the *pubis*, {1,2}, {1,3}, and {2,3} in the previous iteration. The active itemset of the candidate *pubis* where the length is 3 is $ais_3 = \{1, 2, 3\}$.

The physical meaning of active items is that the following process of APHAUP only considers the items in the *ais*. This implies the closure property of *pubi*. If an itemset is a *pubi*, then all of its subsets is also a *pubi*.

Definition 2: The remaining maximal utility of the active itemset in a transaction is denoted as (*rmua*).

Assume an itemset $I = \{i_1, i_2, \ldots, i_n\}$ in a transaction $T = \{i_t, u_t\}$. $I_t = \{i_{t1}, i_{t2}, \ldots, i_{tm}\}$ is the purchase items in this transaction, $U_t = \{u_t(i_{t1}) = u_{t1}, u_{t2}, \ldots, u_{tm}\}$ is the corresponding utility for each item in this transaction, and the active itemsets is *ais*. The *rmua* of i in t denoted as $rmua(i, t)$, which is defined as:

$$rmua(i, t) = max\left\{ u_t(i) \,\middle|\, i \in (ais \cap i_t) \setminus i \right\} \tag{3}$$

Definition 3: The partial maximal utility upper bound is denoted as (*pub*) in a transaction.

Assume an itemset $I = \{i_1, i_2, \ldots, i_n\}$ in a transaction $T = \{i_t, u_t\}$. $I_t = \{i_{t1}, i_{t2}, \ldots, i_{tm}\}$ is the purchase items in this transaction, $U_t = \{u_t(i_{t1}) = u_{t1}, u_{t2}, \ldots, u_{tm}\}$ is the corresponding utility for each item in this transaction, and the active itemsets is *ais*. The *pub* of i in t is denoted as pb_i^t, which is defined as:

$$pb_i^t = \begin{cases} \dfrac{u(i,t) + m \times rmua(i,t)}{|i| + m}, & \text{if } rmua(i,t) > au(i,t) \\ \dfrac{u(i,t) + rmua(i,t)}{|i| + 1}, & \text{if } 0 < rmua(i,t) \leq au(i,t) \\ 0, & \text{if } rmua(i,t) = 0, \end{cases} \tag{4}$$

where m is the number of $(ais \cap i_t) \setminus i$.

Definition 4: The partial maximal utility upper bound is denoted as (*pub*) in a transaction dataset.

Assume an itemset i and a transaction dataset D. The *pub* of i in D is denoted as pb_i, and defined as:

$$pb_i = \sum_{t \in D} pb_i^t \tag{5}$$

Lemma 1: The anti-monotonicity property of *pub* in a transaction.

Assume an itemset $I = \{i_1, i_2, \ldots, i_n\}$ in a transaction $T = \{i_t, u_t\}$, active itemsets = *ais*, and a superset of I, $I' = \{i_1, i_2, \ldots, i_n, i_{n+1}, \ldots, i_o\}$, in which $\forall x \in \{i_{n+1}, \ldots, i_o\} \rightarrow x \in ais \setminus I$, such that $tmu(I, T) \geq pb_i^t \geq au(I', T)$.

Proof: $\because I \subset I' \therefore$ if $i' \in i_t \rightarrow i \in i_t$ and if $i \notin i_t \rightarrow i' \notin i_t$. 1)

1) if $i \notin i_t \rightarrow tmu(i, t) = pb_i^t = au(i', t) = 0$.
2) if $i \in i_t$ and $i' \notin i_t \rightarrow au(i', t) = 0$,
   - if $pb_i^t = \frac{u(i,t)+m\times rmua(i,t)}{|i|+m}$:
     $pb_i^t = \frac{u(i,t)+m\times rmua(i,t)}{|i|+m} \leq \frac{|i|tmu(i,t)+m\times tmu(i,t)}{|i|+m} = tmu(i, t)$.
   - if $pb_i^t = \frac{u(i,t)+rmua(i,t)}{|i|+1}$:
     $pb_i^t = \frac{u(i,t)+rmua(i,t)}{|i|+1} \leq \frac{|i|tmu(i,t)+tmu(i,t)}{|i|+1} = tmu(i, t)$.
   - if $pb_i^t = 0$:
     $pb_i^t = 0 < tmu(i, t)$.

   $\rightarrow tmu(i, t) \geq pb_i^t \geq au(i', t)$.
3) if $i \in i_t$ and $i' \in i_t$,
   $\because$ 2) $\therefore tmu(i, t) \geq pb_i^t$.
   - if $rmua(i, t) > au(i, t)$:
     $pb_i^t = \frac{u(i,t)+m\times rmua(i,t)}{|i|+m} =$

$\frac{\sum_{w=1\sim n} u(i_w,t)+m\times rmua(i,t)}{|i|+m}$
$\because rmua(i,t) \geq u(i_z,t) \forall i_z \in \{i_{n+1},\ldots,i_o\}$
$\therefore \frac{\sum_{w=1\sim n} u(i_w,t)+m\times rmua(i,t)}{|i|+m} \geq \frac{\sum_{w=1\sim o} u(i_w,t)}{n+o-n} = \frac{\sum_{w=1\sim o} u(i_w,t)}{o} = au(i',t)$
.

- if $rmua(i,t) \leq au(i,t)$:
  $pb_i^t = \frac{u(i,t)+rmua(i,t)}{|i|+1}$
  $\because rmua(i,t) \leq au(i,t)$
  $\therefore \frac{u(i,t)+rmua(i,t)}{|i|+1} \geq \frac{u(i,t)+m\times rmua(i,t)}{|i|+m} \geq au(i',t)$.
- if $rmua(i,t) = 0$:
  if $i' \in i_t$, $rmua(i,t)$ is impossible as 0.

$\rightarrow tmu(i,t) \geq pb_i^t \geq au(i',t)$.

Lemma 2: The downward closure property of *pub* is given below. Assume an itemset $I = \{i_1, i_2, \ldots, i_n\}$ in a transaction dataset $D = \{t_1, t_2, \ldots, t_z\}$, active itemsets = *ais*, and a superset of i, $I' = \{i_1, i_2, \ldots, i_n, i_{n+1}, \ldots, i_o\}$, in which $\forall x \in \{i_{n+1}, \ldots, i_o\} \rightarrow x \in ais \setminus i$, such that $auub(i) \geq pb_i \geq au(i')$.

Proof: According to Lemma 1, $tmu(i,t) \geq pb_i^t \geq au(i',t), \forall t \in d$. Accumulate all of the results from each transaction in t, $\rightarrow auub(i) \geq pb_i \geq au(i')$.

The physical meaning of the downward closure property of *pub* is that if the *pb* of an itemset is less than the predefined threshold, then the average-utility of its superset is definitely less than the threshold. Therefore, if there is an itemset in which *pb* is less than the threshold, then it would not be used to combine the new candidate itemset in the Apriori-like process.

Definition 5: High partial upper bound itemset is denoted as (*pubi*). Thus, if an itemset i is a high partial upper bound itemset (*pubi*) and a predefined threshold is r, then it indicates that $pb_i \geq r$.

By Lemma 2, the proposed APHAUP maintains a set of *pubi* to generate the candidate itemsets. This technique has ability to effectively reduce the search space.

B. LEAD PARTIAL UPPER BOUND LEAD-PUB AND LEAD HIGH PARTIAL UPPER BOUND ITEMSET LEAD-PUBI

In this section, a subset of *pubi* is introduced to reduce the search space of the candidate itemsets. This subset is called the lead partial upper bound (*lead-pub*). A predefined order of items in an itemset needs to be set to apply this technique. Before giving the formal definitions, the concept of *lead-pub* is described first. Assume that a predefined item order l is given and a candidate itemset $I_c = \{i_1, i_2, \ldots, i_n\}$ and follows the order l. In the Apriori-like process, this candidate itemset must be made up of a set of *pubi*. This set of *pubi* is $\{\{i_1, i_2, \ldots, i_{n-1}\}, \{i_1, i_2, \ldots, i_{n-2}, i_n\}, \ldots, \{i_2, i_3, \ldots, i_n\}\}$. The first of this set is one of the *lead-pubis*. Due to this extra limitation, a smaller upper bound than *pub* can be defined and described in the following section, which can further reduce the size of the candidate itemsets.

Definition 6: The remaining maximal utility of active itemset in a transaction is denoted as (*hrmua*).

Assume that an itemset $I = \{i_1, i_2, \ldots, i_n\}$, a transaction $T = \{i_t, u_t\}$. $I_t = \{i_{t1}, i_{t2}, \ldots, i_{tm}\}$ is the purchase items in this transaction, $U_t = \{u_t(i_{t1}) = u_{t1}, u_{t2}, \ldots, u_{tm}\}$ is the corresponding utility for each item in this transaction, and the active itemsets is *ais* and a predefined item order $L = \{i_1^l, i_2^l, \ldots, i_p^l\}$. The *lrmua* of i in t is denoted as $lrmua(i,t)$, which is defined as follows.

Assume that $i_n = i_w^l$, and set an itemset $s = \{i_{w+1}^l, i_{w+2}^l, \ldots, i_p^l\}$, we can obtain that:

$$lrmua(i,t) = max\left\{u_t(i) \,\middle|\, i \in (ais \cap i_t \cap s) \setminus i\right\}. \quad (6)$$

Definition 7: The lead partial maximal utility upper bound is denoted as (*lead-pub*) in a transaction dataset.

Assume an itemset $I = \{i_1, i_2, \ldots, i_n\}$ in a transaction $T = \{i_t, u_t\}$. $i_t = \{i_{t1}, i_{t2}, \ldots, i_{tm}\}$ is the purchase items in this transaction, $U_t = \{u_t(i_{t1}) = u_{t1}, u_{t2}, \ldots, u_{tm}\}$ is the corresponding utility for each item in this transaction, the active itemsets is *ais*, and a predefined item order $L = \{i_1^l, i_2^l, \ldots, i_p^l\}$. The *lead-pub* of i in t is denoted as lpb_i^t, which is defined as:

$$lpb_i^t = \begin{cases} \frac{u(i,t)+m\times rmua(i,t)}{|i|+m}, \\ \quad \text{if } lrmua(i,t) > au(i,t) \\ \frac{u(i,t)+rmua(i,t)}{|i|+1}, \\ \quad \text{if } 0 < lrmua(i,t) \leq au(i,t) \\ 0, \quad \text{if } lrmua(i,t) = 0 \end{cases} \quad (7)$$

where m is the number of $(ais \cap i_t \cap s) \setminus i$ (the definition of s is same as Definition 6).

Definition 8: The lead partial maximal utility upper bound is denoted as (*lead-pub*) in a transaction dataset.

Assume that there is an itemset i and a transaction dataset D, where the *lead-pub* of i in D is denoted as lpb_i, which is defined as:

$$lpb_i = \sum_{t \in D} lpb_i^t \quad (8)$$

Definition 9: The lead high partial upper bound itemset is denoted as (*lead-pubi*). Thus, if an itemset i is a lead high partial upper bound itmeset (*lead-pubi*) and a predefined threshold is r, then $pb_i \geq r$.

The lead high partial upper bound itemset *lead-pubi* has a similar definition to *pubi*. Therefore, *lead-pubi* also has the downward closure property such as *pubi*. The set of *lead-pubi* is a subset of the set of *pubi* due to the strict limitation than *pubi*. It can further reduce the size of the candidate itemsets in the proposed APHAUP.

IV. APRIORI-BASED HAUP WITH PRE-LARGE CONCEPT, APHAUP

This section describes details about the proposed APHAUP. It includes two parts: the first is the general scan for the input dataset and the second part is the incremental process of the proposed APHAUP method. APHAUP utilizes the proposed

rescan threshold to select the maintenance progress automatically for incremental datasets. The proposed APHAUP also applies the new proposed upper bounds (*pub* and *lead-pub*), and it maintains two itemsets (*pubi* and *lead-pubi*) to reveal all of the HAUsI in a transaction dataset. The detailed pseudocodes are respectively given in Algorithms 1 and 2.

Different from traditional HAUI mining, the incremental process needs to set a pre-large itemsets threshold. In the beginning, safety bound is set in line 3. APHAUP applies *auub* to initialize *pubi* and *lead-pubi*. At this time, all of the itemsets in *pubi* and *lead-pubi* have obtained 1-items. The APHAUP also calculates the average utility for all the itemsets (with 1-items) at the same time. Then, it is determined whether the average utility is larger than the high average utility threshold or the pre-large utility threshold. This progress is performed in lines 4-23. There is a while loop for each iteration to reveal the high average itemsets and pre-large itemsets in lines 24-47. In lines 26-28, each itemset in *lead-pubis* is combined with the itemsets in *pubis* to generate the candidate itemsets. Then, APHAUP scans the dataset again to obtain the utility information for each itemset in the candidate itemsets. Finally, at the end of this while loop, APHAUP updates the set of *pubis*, *lead-pubis*, P, and H. If *pubis* and *lead-pubis* are all empty set, then the loop is going to be stopped and the process will output the HAUIs, pre-large itemsets and the rescan threshold.

Algorithm 2 is the incremental process of the proposed APHAUP framework. While an incremental dataset is added in the original dataset, APHAUP performs the incremental process to speed up the mining algorithm. First, APHAUP maintains a rescan threshold r. After performing Algorithm 1, a new rescan threshold would be set to control the process to rescan the whole dataset again or perform the incremental algorithm. This means that if the size of the incremental dataset is less than r, then it is impossible for the itemsets whose average utility is less than the pre-large threshold in Algorithm 1 to be a HAUI after updating the utility information. APHAUP updates the value r in line 2 and it is checked whether the value is larger than 0. If the value is less than 0, then APHAUP performs Algorithm 1 to rescan the new whole dataset to update the new P, H, and r (lines 9-11); if not, then the incremental process performs Algorithm 1 for the incremental dataset D' to obtain the partial P' and H' (lines 4), and a rescan itemsets S is generated (line 5). S contains all of the itemsets that are in $(P \cup H)$ and not in $(P' \cup H')$. In the end of the updating process in lines 6, APHAUP checks the utility information of S in D'. In line 7, APHAUP uses P', H' and S to update P and H. In line 13, the updating process outputs the updated P, H, and r.

V. EXPERIMENTAL RESULTS

In this paper, a new upper-bound, called the partial maximal utility upper-bound (*pub*) is proposed. In the new algorithm (APHAUP) proposed in this paper, a smaller upper-bound than *pub* is applied called lead high partial maximal utility upper-bound (*lpub*). The proposed *lpub* can effectively reduce the size of the candidate itemsets. This section describes two implementations of the proposed APHAUP,

Algorithm 1 Apriori-Based HAUP With Pre-Large Concept

Input: a transaction dataset, D,
a high average utility threshold, at,
and a pre-large utility threshold, pt.
Output: a set of pre-large itemsets, P,
a set of HAUI, H,
and a rescan threshold, r.

1: set $H = \emptyset$ for HAUIs;
2: set $P = \emptyset$ for PreLarge Itemsets;
3: count rescan threshold r for D by formula (1);
4: **for** each transaction t in D **do**
5: find maximal utility m in t;
6: **for** each item i in t **do**
7: $f[i] = A[i] + m$;
8: $au(i) = au(i) + u(i, t)$;
9: **end for**
10: **end for**
11: **if** $au(i) \geq at$ **then**
12: $H \leftarrow i$;
13: **else if** $au(i) \geq pt$ **then**
14: $P \leftarrow i$;
15: **end if**
16: set *pubis* $\leftarrow \emptyset$;
17: set *lead-pubis* $\leftarrow \emptyset$;
18: **for** each item i **do**
19: **if** $A[i] \geq pt$ **then**
20: *pubis* $\leftarrow i$;
21: *lead-pubis* $\leftarrow i$;
22: **end if**
23: **end for**
24: **while** *pubis* $\neq \emptyset$ and *lead-pubis* $\neq \emptyset$ **do**
25: set $C = \emptyset$ for the candidate itemsets;
26: **for** each itemset e in *lead-pubis* **do**
27: search (combine) *pubis* by e to generate candidate itemsets $\rightarrow C$
28: **end for**
29: set *pubis* $\leftarrow \emptyset$;
30: set *lead-pubis* $\leftarrow \emptyset$;
31: **for** each transaction t in D **do**
32: scan utility information for each candidate itemset in t;
33: **end for**
34: **for** each candidate itemset c in C **do**
35: **if** $pb_c \geq p$ **then**
36: *pubis* $\leftarrow c$;
37: **end if**
38: **if** $lpb_c \geq p$ **then**
39: *lead-pubis* $\leftarrow c$;
40: **end if**
41: **if** $au(c) \geq at$ **then**
42: $H \leftarrow c$;
43: **else if** $au(c) \geq pt$ **then**
44: $P \leftarrow c$;
45: **end if**
46: **end for**
47: **end while**
48: **return** P, H, r;

Algorithm 2 Incremental Process for APHAUP

Input: a incremental transaction dataset, D',
a set of pre-large itemsets, P,
a set of HAUI, H,
a remaining rescan threshold, r,
a high average utility threshold, at
and a pre-large utility threshold, pt.
Output: a set of updated pre-large itemsets, P,
a set of updated HAUI, H,
and updated rescan threshold, r.

1: inc = the size of the input D';
2: update $r = r - inc$;
3: **if** $r \geq 0$ **then**
4: apply Algorithm 1 for D', at and pt obtain P', H';(ignore r')
5: $S \leftarrow (P \cup H) \cap \sim (P' \cup H')$;
6: scan utility information for S in D';
7: update P, H from P, H, P', H', S
8: **else**
9: update $D \leftarrow D + D'$;
10: rescan whole dataset by Algorithm 1 for D, t and p;
11: update P, H, r;
12: **end if**
13: **return** P, H, r;

TABLE 1. Characteristics of used datasets.

Dataset	#|D|	#|I|	AvgLen	Type
retail	88,162	16,470	10	sparse
foodmart	21,557	1,559	4	sparse
BMS	59,602	497	2.5	sparse
mushroom	8,124	119	23	dense
chess	3,196	75	37	dense
accidents	34,018	468	34	dense

one is with the *lpub* and another is without the *lpub*. Besides, the proposed APHAUP was separated into two different experiments, one is performed for the incremental process and utilized the pre-large concept while another is used to rescan the whole dataset when the size of the dataset is increased with some new transactions. The experimental results will show the performance of APHAUP with six different real datasets [37] using different parameter settings. The datasets and their characteristics are shown in Table 1. All the algorithms were implemented in Java language and executed on a computer equipped with an Intel(R) Core (TM) i5-5257U 2.7 GHz processor and 8 GB main memory, running on the MacOS Mojave operating system.

A. RUNTIME WITH DIFFERENT INCREMENTAL SIZE

In this section, the runtimes of the proposed APHAUP with different scales of sizes for the incremental data in six real datasets are compared. The experimental results are shown in Figures 2 to 7. Among them, Apriori(A) is the algorithm that applies Algorithm 1 to scan the whole dataset when the incremental data is inserted in the original dataset but

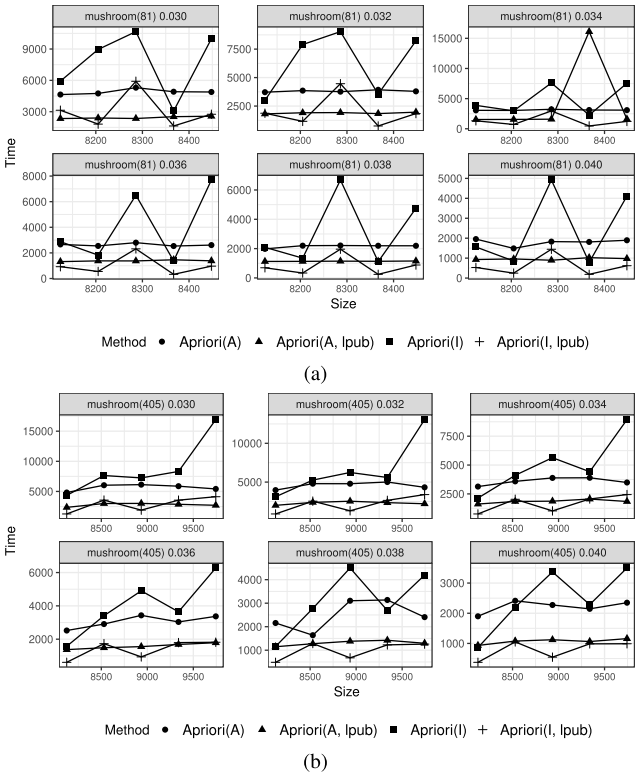


FIGURE 2. The runtimes for APHAUP in the mushroom dataset with two different incremental size.

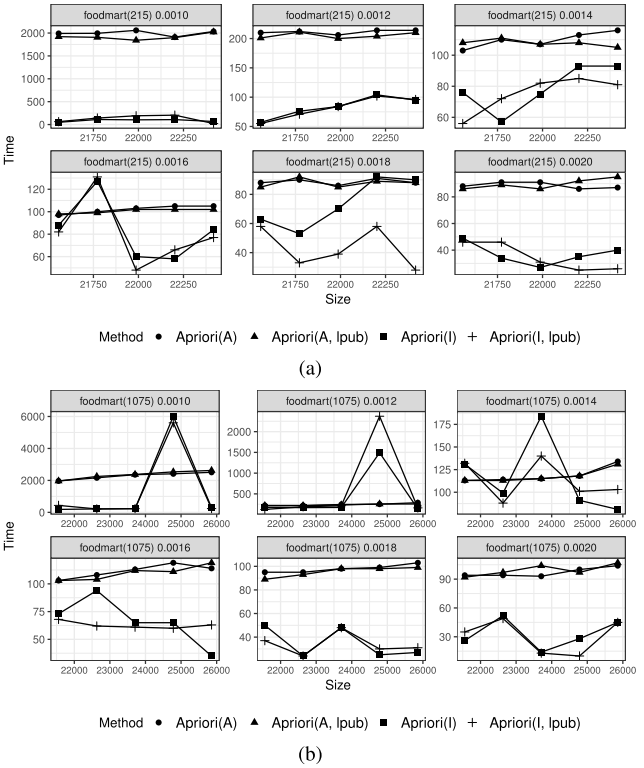


FIGURE 3. The runtimes for APHAUP in the foodmart dataset with two different incremental size.

does not use *lpub*, Apriori(I) is the algorithm that is the proposed APHAUP with the incremental process (Algorithm 2)

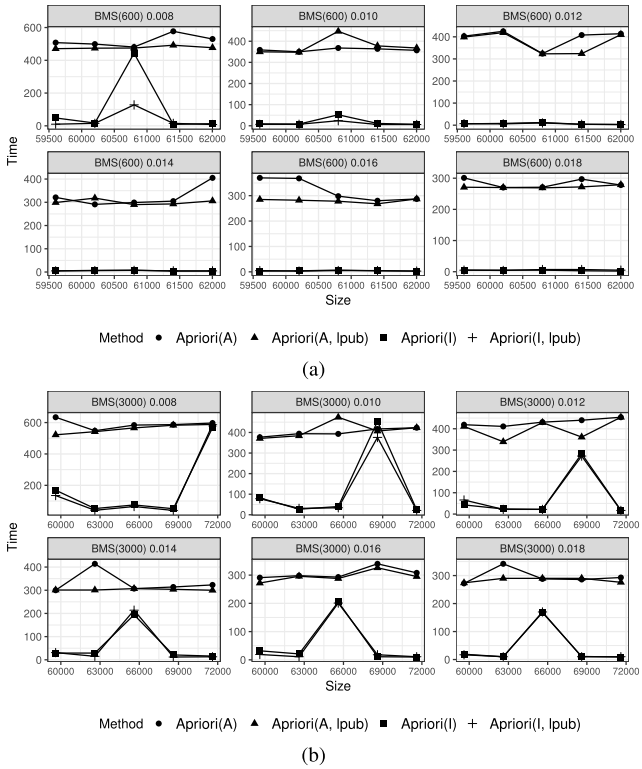


FIGURE 4. The runtimes for APHAUP in the BMS dataset with two different incremental size.

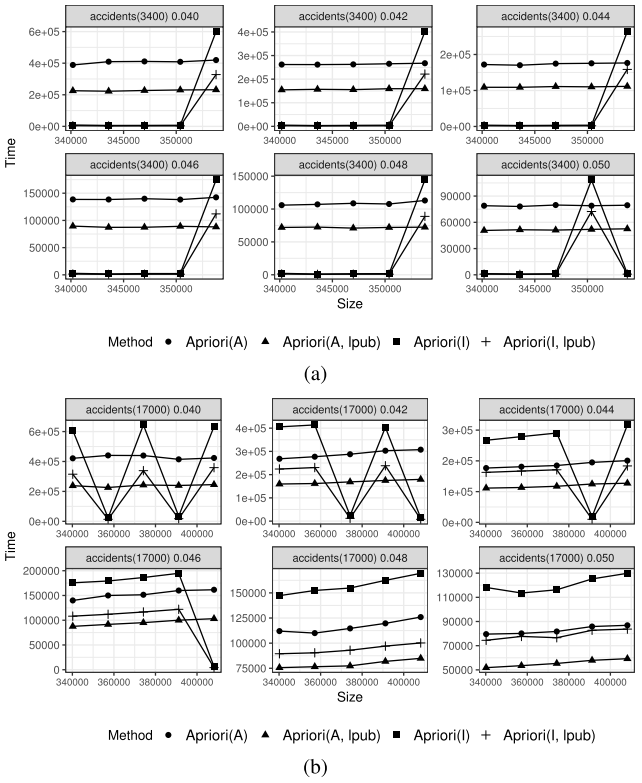


FIGURE 5. The runtimes for APHAUP in the accidents dataset with two different incremental size.

but does not use *lpub*, Apriori(A,*lpub*) is the algorithm that applies Algorithm 1 to scan the whole dataset when the incremental data is inserted in the original dataset, Apriori(I,*lpub*) is the algorithm that is the proposed APHAUP with the incremental process (Algorithm 2), the numbers on the top of the figures are the minimal support (minimal average utility threshold), and the numbers in brackets are the different scales of sizes for the incremental data.

First, we focus on the comparisons of Apriori(A) and Apriori(A,*lpub*). This shows the good performance of the proposed lead high partial upper bound. In most real datasets, Apriori(A,*lpub*) has obtained better runtime than the Apriori(A), especially in datasets that has a high density or a huge number of transactions, such as the mushroom and the accidents datasets. However, in some sparse datasets such as the foodmart and the BMS datasets, the Apriori(A,*lpub*) has no obvious advantage over Apriori(A). From this, it can be concluded that a dense database is suitable for Apriori(A,*lpub*), but for sparse databases, the difference between the two algorithms is not obvious. The reason for this result is that in the sparse datasets, the Apriori(A,*lpub*) does not reduce the size of candidate itemsets but result in additional calculations. Thus, it needs more computational cost to find the required information. Second, Aprioir(I) (incremental mode) could handle incremental situations and reduce the runtime effectively. Apriori(I) suffered from many database rescans (rescanning the whole dataset) in a large scale incremental environment. For example, in Figure 4(a), Apriori(I) almost performs the updating progress immediately. In Figure 4(b), the Apriori(I) rescanned the whole dataset several times because the size of incremental data is larger than the rescan threshold. It is worth to notice that the runtime of the rescan process for the Apriori(I) is always larger than Apriori(A). The reason is since Apriori(I) needs to apply a pre-large threshold, and it causes the proposed Apriori(I) to maintain more itemsets during the mining process. Moreover, it needs to perform the mining process for the new utility information and the original one to obtain the updated utility information for maintenance. For the same reason, the runtime of the rescan process of Apriori(I,*lpub*) is always larger than Apriori(A,*lpub*). In some cases, the performance of Apriori(I) is worse than Apriori(A) (in Figure 2). However, the performance of Apriori(I,*lpub*) is still better than Apriori(I).

Next, the influence of different threshold settings is discussed. There are two bad influences for the incremental-based algorithms. The first one is performing the rescanning process (includes the first time rescanning). Due to the threshold used for the pre-large itemsets, the incremental-based algorithms reveal more itemsets in the rescanning process. When the applied upper bound is not tight enough, there will be a lot of candidates needed for evaluation. This is the limitation of the incremental-based algorithms. Therefore, it spends more time than the conventional mining method. The second one is the updating process for some uncertain itemsets (ex. check a pre-large itemset becomes a large itemset or not), especially for the situation with the large

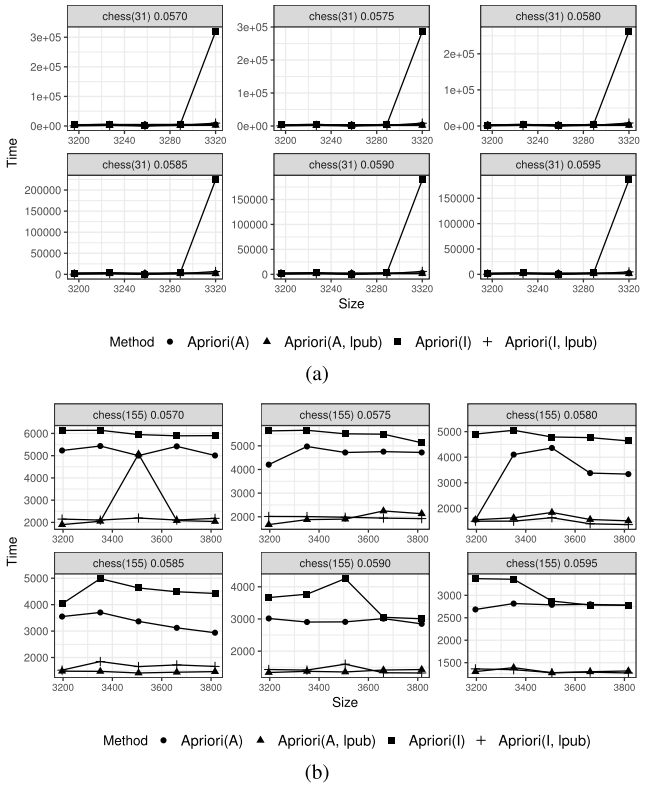


FIGURE 6. The runtimes for APHAUP in the chess dataset with two different incremental size.

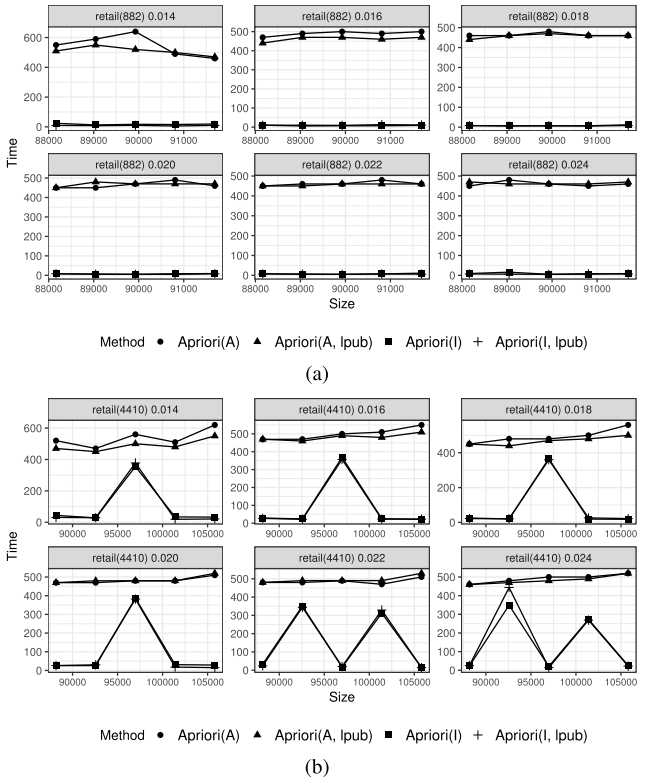


FIGURE 7. The runtimes for APHAUP in the retail dataset with two different incremental size.

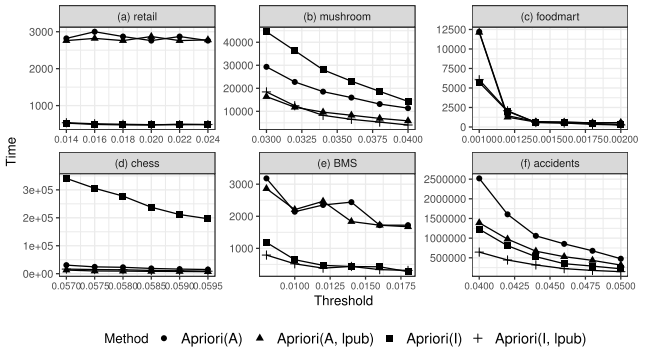


FIGURE 8. The runtimes for APHAUP with different minimal support.

scale of large itemsets and pre-large itemsets. It might spend more computational costs than the ordinary scanning process. From Figure 4(a), it represents the size of incremental data is 600, and Figure 4(b) represents the size of incremental data is 3,000. As we can see from the Figure 4(a), the minimal support (minimal average utility threshold) is set as 0.008, the rescanning process will be preformed after the third insertion operation of the new dataset. Due to Formula 1, a bigger minimal support threshold causes a larger safety bound. Thus, if we set a small threshold for applying the incremental-based algorithm, the gap of the pre-large threshold and minimal average utility threshold should be increased. Otherwise, if the size of the incremental data is large (such as in Figure 4(b)), the frequent rescanning process will cause longer maintaining time and the total CPU time will be longer than the normal mining process. Obviously, the proposed novel upper-bound can effectively relieve the punishment of the incremental-based algorithms. In accidents and chess datasets, the performance of incremental Apriori without *lpub* is distinctly worse than the other algorithms.

B. RUNTIME WITH DIFFERENT MINIMAL SUPPORT (HIGH AVERAGE UTILITY THRESHOLD)

This section compares the runtime for the proposed APHAPU with different minimal supports (high average utility threshold) in six real datasets. The experimental results are shown in Figure 8. Among the figures, the data on the horizontal axis represents the different minimal supports.

The experimental results in this section show the runtime of the original database. It is similar to the results in the previous section. Apriori(A,*lpub*) and Apriori(I,*lpub*) are both good for some datasets, such as the mushroom, chess, and accidents datasets. It should be noticed that these datasets are dense type, as the same conclusion in the previous section. Furthermore, it can be observed that with the increasing of the minimal support, the runtime of the algorithm is gradually reduced. This is because the larger the minimal support, the fewer candidates will be produced. Note that the performance of Apriori(I) in mushroom and chess datasets is very poor. The reason is that since the Apriori(I) applies a loose upper-bound and produces too many candidate itemsets in the newly inserted transactions (shown in Figure 9 and discussed in the following session), each incremental process needs to

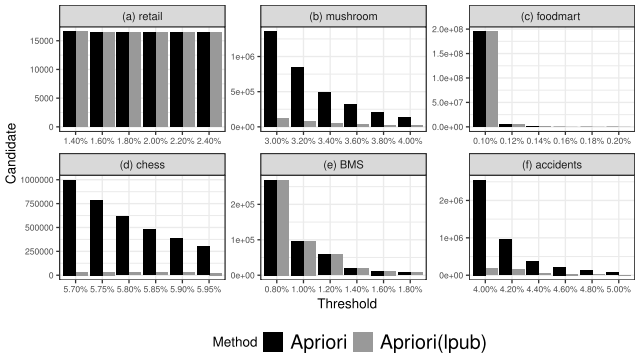


FIGURE 9. The numbers of the candidate itemsets for APHAUP with different minimal support.

evaluate a huge number of candidate itemsets. In this case, rescan the whole dataset might spend less CPU time than the incremental-based algorithms.

C. THE NUMBER OF CANDIDATES FOR PROPOSED METHOD AND FUP-BASED MODEL

In the final part of the experimental results, the numbers of candidates depending on whether using *lpub* in six real datasets are shown in Figure 9. The definition of a candidate is an itemset that needs to be calculated for the utility value in the whole dataset. Due to the benefit of lead partial upper bound (*lpub*), there are few itemsets needed to check for the utility values in the updated dataset. Comparing with Apriori-like approach, the cost of rescanning the whole updated dataset is not huge for the Apriori(lpub) in some datasets. It is very suitable to be applied in a stream environment. On the other hand, the traditional upper-bound cannot select potential candidate itemsets precisely, especially for mushroom and chess datasets. For those datasets, there is a huge number of itemsets between two upper-bounds. Thus, the traditional Apriori-like approach needs to perform the evaluating process for a lot of unpromising itemsets no matter in the original dataset or in the newly inserted dataset. Thus, Apriori(I) has the worst performance in mushroom and chess datasets and shown in the previous section.

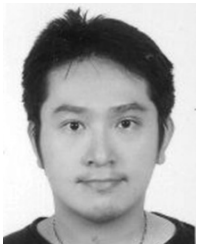
VI. CONCLUSION

In this paper, we design an incremental insertion algorithm based on the pre-large concept for high average-utility itemset mining. Furthermore, we proposed two new upper-bounds to reduce the size of candidates, respectively called *pub* and *lead-pub* in the developed APHAUP algorithm. The results of experiments showed that APHAUP with *lead-pub* can significantly reduce the execution time for updating the discovered HAUIs compared to APHAUP with *pub* in dense datasets. Moreover, the number of determined candidates is much less than APHAUP with *pub*. It also showed that the pre-large concept has the potential ability to improve the maintenance performance for updating the discovered HAUIs. For the further research issues, we will try to apply the pre-large concept for more domains and applications in knowledge discovery. Besides, we will also develop a new upper-bound based on the pre-large concept in order to further enhance the mining performance. In real-world situations, transaction deletion and modification are also significant. They should also be considered in the dynamic situations for the maintaining the discovered HAUIs.

REFERENCES

[1] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proc. Int. Conf. Very Large Data Bases*, vol. 1215, 1994, pp. 487–499.
[2] M.-S. Chen, J. Soo Park, and P. S. Yu, "Efficient data mining for path traversal patterns," *IEEE Trans. Knowl. Data Eng.*, vol. 10, no. 2, pp. 209–221, Mar. 1998.
[3] Z.-H. Deng and S.-L. Lv, "Fast mining frequent itemsets using nodesets," *Expert Syst. Appl.*, vol. 41, no. 10, pp. 4505–4512, Aug. 2014.
[4] W. Gan, J. C. W. Lin, P. Fournier-Viger, H. C. Chao, and P. S. Yu, "A survey of parallel sequential pattern mining," *ACM Trans. Knowl. Discovery Data*, vol. 3, no. 3, pp. 1–34, 2019.
[5] Z. Ling, T. Zengrui, and N. Metawa, "Data mining-based competency model of innovation and entrepreneurship," *J. Intell. Fuzzy Syst.*, vol. 37, no. 1, pp. 35–43, Jul. 2019.
[6] Z. Zhao, C. Li, X. Zhang, F. Chiclana, and E. H. Viedma, "An incremental method to detect communities in dynamic evolving social networks," *Knowl.-Based Syst.*, vol. 163, pp. 404–415, Jan. 2019.
[7] C.-M. Chen, B. Xiang, Y. Liu, and K.-H. Wang, "A secure authentication protocol for Internet of vehicles," *IEEE Access*, vol. 7, pp. 12047–12057, 2019.
[8] T.-Y. Wu, C.-M. Chen, K.-H. Wang, C. Meng, and E. K. Wang, "A provably secure certificateless public key encryption with keyword search," *J. Chin. Inst. Eng.*, vol. 42, no. 1, pp. 20–28, Jan. 2019.
[9] L. Ni, F. Tian, Q. Ni, Y. Yan, and J. Zhang, "An anonymous entropy-based location privacy protection scheme in mobile social networks," *EURASIP J. Wireless Commun. Netw.*, vol. 2019, no. 1, pp. 1–19, Dec. 2019.
[10] X. Wang, S.-J. Ji, Y.-Q. Liang, H.-F. Leung, and D. K. W. Chiu, "An unsupervised strategy for defending against multifarious reputation attacks," *Int. J. Speech Technol.*, vol. 49, no. 12, pp. 4189–4210, Dec. 2019.
[11] C.-M. Chen, Y. Huang, K.-H. Wang, S. Kumari, and M.-E. Wu, "A secure authenticated and key exchange scheme for fog computing," *Enterprise Inf. Syst.*, pp. 1–16, Jan. 2020, doi: 10.1080/17517575.2020.1712746.
[12] Z. Meng, J.-S. Pan, and K.-K. Tseng, "PaDE: An enhanced differential evolution algorithm with novel control parameter adaptation schemes for numerical optimization," *Knowl.-Based Syst.*, vol. 168, pp. 80–99, Mar. 2019.
[13] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining frequent patterns without candidate generation: A frequent-pattern tree approach," *Data Mining Knowl. Discovery*, vol. 8, no. 1, pp. 53–87, Jan. 2004.
[14] W. Gan, C.-W. Lin, P. Fournier-Viger, H.-C. Chao, V. Tseng, and P. Yu, "A survey of utility-oriented pattern mining," *IEEE Trans. Knowl. Data Eng.*, early access, Sep. 20, 2019, doi: 10.1109/TKDE.2019.2942594.
[15] M. Liu and J. Qu, "Mining high utility itemsets without candidate generation," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2012, pp. 55–64.
[16] H. Yao, H. J. Hamilton, and C. J. Butz, "A foundational approach to mining itemset utilities from databases," in *Proc. SIAM Int. Conf. Data Mining*, Apr. 2004, pp. 215–221.
[17] T.-P. Hong, C.-H. Lee, and S.-L. Wang, "Effective utility mining with the measure of average utility," *Expert Syst. Appl.*, vol. 38, no. 7, pp. 8259–8265, Jul. 2011.
[18] G. C. Lan, T. P. Hong, and V. S. Tseng, "Efficiently mining high average-utility itemsets with an improved upper-bound strategy," *Int. J. Inf. Technol. Decis. Making*, vol. 11, no. 5, pp. 1009–1030, 2012.
[19] Y. Liu, W. K. Liao, and A. Choudhary, "A two-phase algorithm for fast discovery of high utility itemsets," in *Proc. Pacific–Asia Conf. Adv. Knowl. Discovery Data Mining*, 2005, pp. 689–695.
[20] C.-W. Lin, T.-P. Hong, and W.-H. Lu, "An effective tree structure for mining high utility itemsets," *Expert Syst. Appl.*, vol. 38, no. 6, pp. 7419–7424, Jun. 2011.
[21] V. S. Tseng, B.-E. Shie, C.-W. Wu, and P. S. Yu, "Efficient algorithms for mining high utility itemsets from transactional databases," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 8, pp. 1772–1786, Aug. 2013.

[22] J. Liu, K. Wang, and B. C. M. Fung, "Direct discovery of high utility itemsets without candidate generation," in *Proc. IEEE 12th Int. Conf. Data Mining*, Dec. 2012, pp. 984–989.
[23] J. Liu, K. Wang, and B. C. M. Fung, "Mining high utility patterns in one phase without generating candidates," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 5, pp. 1245–1257, May 2016.
[24] J. M.-T. Wu, J. C.-W. Lin, and A. Tamrakar, "High-utility itemset mining with effective pruning strategies," *ACM Trans. Knowl. Discovery Data*, vol. 13, no. 6, pp. 1–22, Dec. 2019.
[25] S. J. Yen and Y. S. Lee, "Mining high utility quantitative association rules," in *Proc. Int. Conf. Data Ware Housing Knowl. Discovery*, 2007, pp. 283–292.
[26] S. Zida, P. Fournier-Viger, J. C.-W. Lin, C.-W. Wu, and V. S. Tseng, "EFIM: A fast and memory efficient algorithm for high-utility itemset mining," *Knowl. Inf. Syst.*, vol. 51, no. 2, pp. 595–625, May 2017.
[27] C. W. Lin, T. P. Hong, and W. H. Lu, "Efficiently mining high average utility itemsets with a tree structure," in *Proc. Asian Conf. Intell. Inf. Database Syst.* Hue City, Vietnam: Springer, 2010, pp. 131–139.
[28] J. C.-W. Lin, S. Ren, P. Fournier-Viger, and T.-P. Hong, "EHAUPM: Efficient high average-utility pattern mining with tighter upper bounds," *IEEE Access*, vol. 5, pp. 12927–12940, 2017.
[29] T.-P. Hong, C.-Y. Wang, and Y.-H. Tao, "A new incremental data mining algorithm using pre-large itemsets1," *Intell. Data Anal.*, vol. 5, no. 2, pp. 111–129, Mar. 2001.
[30] A. Erwin, R. P. Gopalan, and N. Achuthan, "Effcientmining of high utility itemsets from large datasets," in *Proc. Pacific–Asia Conf. Adv. Knowl. Discovery Data Mining*. Suzhou, China: Springer, 2008, pp. 554–561.
[31] D. W. Cheung, J. Han, V. T. Ng, and C. Y. Wong, "Maintenance of discovered association rules in large databases: An incremental updating technique," in *Proc. 12th Int. Conf. Data Eng.*, 1996, pp. 106–114.
[32] T. P. Hong, C. W. Lin, and Y. L. Wu, "Incrementally fast updated frequent pattern trees," *Expert Syst. Appl.*, vol. 34, no. 4, pp. 2424–2435, 2008.
[33] C.-W. Lin, T.-P. Hong, W.-Y. Lin, and G.-C. Lan, "Efficient updating of sequential patterns with transaction insertion," *Intell. Data Anal.*, vol. 18, no. 6, pp. 1013–1026, Oct. 2014.
[34] C.-W. Lin, G.-C. Lan, and T.-P. Hong, "An incremental mining algorithm for high utility itemsets," *Expert Syst. Appl.*, vol. 39, no. 8, pp. 7173–7180, Jun. 2012.
[35] T.-P. Hong, C.-H. Lee, and S.-L. Wang, "An incremental mining algorithm for high average-utility itemsets," in *Proc. 10th Int. Symp. Pervas. Syst., Algorithms, Netw.*, 2009, pp. 421–425.
[36] C.-W. Lin, T.-P. Hong, and W.-H. Lu, "The pre-FUFP algorithm for incremental mining," *Expert Syst. Appl.*, vol. 36, no. 5, pp. 9498–9505, Jul. 2009.
[37] P. Fournier-Viger, J. C. W. Lin, A. Gomariz, T. Gueniche, A. Soltani, Z. Deng, and H. T. Lam, "The SPMF open-source data mining library version 2," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*. Riva del Garda, Italy: Springer, 2016, pp. 36–40.



JIMMY MING-TAI WU received the Ph.D. degree in computer science and engineering from National Sun Yat-sen University, Kaohsiung, Taiwan. He was an Assistant Professor with the School of Computer Science and Technology, Harbin Institute of Technology-Shenzhen, China. He worked in an IC design company in Taiwan as a Firmware Developer and an Information Technology Manager for two years. He was also a Research Scholar with the Department of Computer Science and Information Engineering, National University of Kaohsiung, Kaohsiung, and with the Department of Computer Science, College of Engineering, University of Nevada, Las Vegas. He is currently an Assistant Professor with the College of Computer Science and Engineering, Shandong University of Science and Technology. His current research work is related to data mining, big data, cloud computing, artificial intelligence, evolutionary computation, machine learning, and deep learning.

QIAN TENG is currently pursuing the master's degree with the School of Computer Science and Engineering, Shandong University of Science and Technology, Qiandao, China. Her research area is big data, and at present the main research is utility pattern mining.

JERRY CHUN-WEI LIN (Senior Member, IEEE) received the Ph.D. degree from the Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan. He is currently a Full Professor with the Department of Computer Science, Electrical Engineering and Mathematical Sciences, Western Norway University of Applied Sciences, Bergen, Norway. He is also the Director of IKELab. He has published more than 300 research articles in refereed journals (the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, IEEE TRANSACTIONS ON CYBERNETICS, IEEE SYSTEMS JOURNAL, ACM TKDD, and ACM TDS) and international conferences (IEEE ICDE, IEEE ICDM, PKDD, and PAKDD). By Google Scholar, his publications have been cited by more than 3,000 times with H-index 33 and i10-index 113. He has filed and held 25 invention patents and two of them are U.S. patents. His research interests include data mining, soft computing, artificial intelligence and machine learning, and privacy-preserving and security technologies. He is the Fellow of IET and a Senior Member of ACM. He also serves as the Editor-in-Chief of *International Journal of Data Science and Pattern Recognition* and an Associate Editor of IEEE ACCESS and *Journal of Internet Technology*. He is the project Co-Leader of well-known SPMF: An Open-Source Data Mining Library, which is a toolkit offering multiple types of data mining algorithms. He is also the Founder and the Leader of PPSF project. Moreover, he has been awarded as the Most Cited Chinese Researcher by Elsevier, in 2018.

CHIEN-FU CHENG received the Ph.D. degree in computer science from National Chiao-Tung University, Hsinchu, Taiwan, in 2008. He is currently a Full Professor with the Department of Computer Science and Information Engineering and with the Graduate Institute of Networking and Multimedia, Tamkang University, New Taipei, Taiwan. He published many articles in several prestigious conferences and journals, such as the IEEE LCN, IEEE MASS, IEEE VTC, IEEE WCNC, IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE INTERNET OF THINGS JOURNAL, IEEE SENSORS JOURNAL, and IEEE COMMUNICATIONS LETTERS. His current research interests include wireless communication and mobile computing, wireless ad hoc and sensor networks, distributed computing, and fault tolerant computing. He served as a TPC member of more than 50 communication and computer conferences, such as the IEEE GLOBECOM, IEEE ICC, IEEE INFOCOM, and IEEE WCNC. He is currently an Associate Editor for IEEE ACCESS. For more details, please refer to his website (http://mail.tku.edu.tw/cfcheng/eng.html).