Received April 17, 2020, accepted May 5, 2020, date of publication May 12, 2020, date of current version May 26, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2994059

Mining Correlated High Utility Itemsets in One Phase

BAY VO[1], LE V. NGUYEN[2], VINH V. VU[3], MI T. H. LAM[2], THUY T. M. DUONG[2], LY T. MANH[2], THUY T. T. NGUYEN[2], LOAN T. T. NGUYEN[4,5], AND TZUNG-PEI HONG[6,7]

[1]Faculty of Information Technology, Ho Chi Minh City University of Technology (HUTECH), Ho Chi Minh City 700000, Vietnam
[2]Faculty of Information Technology, Ho Chi Minh City University of Food Industry, Ho Chi Minh City 700000, Vietnam
[3]Institute of Research and Development, Duy Tan University, Da Nang 550000, Vietnam
[4]School of Computer Science and Engineering, International University, Ho Chi Minh City 700000, Vietnam
[5]Vietnam National University, Ho Chi Minh City 700000, Vietnam
[6]Department of Computer Science and Information Engineering, National University of Kaohsiung, Kaohsiung 811, Taiwan
[7]Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung 804, Taiwan

Corresponding author: Loan T. T. Nguyen (nttloan@hcmiu.edu.vn)

ABSTRACT High-utility itemset mining (HUIM) in transaction databases has been extensively studied to discover interesting itemsets from users' purchase behaviors. With this, business managers can adjust their sale strategies appropriately to increase profit. HUIM approaches usually focus on the utility values of itemsets, but rarely evaluate the correlation of items in itemsets. Many high-utility itemsets are weakly correlated and have no real meaning. To address this issue, we suggest an algorithm, called CoHUI-Miner, to efficiently find correlated high-utility itemsets. In the proposed algorithm, we use the database projection mechanism to reduce the database size and present a new concept, called the prefix utility of projected transactions, to eliminate itemsets which do not satisfy the minimum threshold in the mining process. Experimental evaluation on two types of datasets from sparse to dense ones shows that CoHUI-Miner can efficiently mine correlated high-utility itemsets with regard to both execution time and memory usage.

INDEX TERMS Correlated high-utility itemset, data mining, high-utility pattern, projected database.

I. INTRODUCTION

Data in a variety of applications are more and more expanded and exploited, especially for business activities, which attracts great attention of scientists. Many methods have been proposed and brought into practical usage. Among them, data mining is an interesting direction to study. It includes frequent itemsets [1]–[5], association rules [6]–[8], high utility itemsets [9]–[11], etc. Frequent pattern mining (FPM) is an interesting problem in the field of data mining, with the target of finding frequent itemsets from a transaction database. Frequent itemsets have been widely studied and play an important role in the context of association rule mining (ARM) [1],[2], [5] for analyzing customers' buying behaviors. However, the limitation of ARM is that it does not consider the benefits of items and the profits of itemsets. Therefore, in 2004 Yao *et al.* [12] defined two types of utilities for items, internal utility and external utility, to mine the itemset utilities from databases. High-utility itemset mining (HUIM) was presented to mine itemsets having high-utility values (or profits) in a transaction database, and was implemented in approaches such as Two-Phase [13], UP-Growth [14], HUI-Miner [15], FHM [16], EFIM [17], and D2HUP [18]. Yun *et al.* [19] proposed the MU-Growth algorithm (Maximum Utility Growth), including two effective candidate pruning techniques in the mining process. In addition, the authors built an MIQ-Tree tree structure that can obtain data with a single-pass. This data structure can be used to find efficient patterns from incremental databases. The experimental results showed that MU-Growth is more effective than other algorithms in terms of execution time and candidate storage. In particular, the algorithm excels when the database contains many long transactions or low minimum utility thresholds. In 2016, Dawar *et al.* [20] presented the UFH algorithm to exploit HUIs efficiently on the sparse dataset. Moreover, on the sparse dataset, some other techniques also presented to discovery useful knowledge such as randomized latent factor model [21], distributed alternative stochastic gradient descent model DASGD [22] or model based on SGD extensions [23]. Recently, Nguyen *et al.* [24] proposed an algorithm that relies on a new tight database format named MEFIM (Modified EFficient high-utility Itemset Mining), which can discover the desired itemsets efficiently. The authors then introduced the iMEFIM algorithm, which is

The associate editor coordinating the review of this manuscript and approving it for publication was Xin Luo.

an improved version of MEFIM. The iMEFIM employs the P-set structure to shrink the amount of transaction scans and accelerate the mining process. The experimental evaluation on several databases showed that iMEFIM is efficient in terms of time and memory.

However, many of the discovered HUIs contain items that have weak correlations. For example, when a customer has just purchased a pair of diamond cores that represent a huge profit and by chance that same transaction also has a tissue product, then this means that {Diamond} is already an HUI, so when combined with the tissue product or any other item in that transaction, then both represent a high utility itemset. However, the correlation between diamond and tissue items is not a strong one in reality. Lin *et al.* [25] thus offered the FDHUP algorithm to efficiently discover discriminative high-utility itemsets with strong frequency affinity. After that, Gan *et al.* [26] proposed an algorithm which considers both the utility and correlation between items in a transaction database, called CoHUIM. However, the number of candidates generated by these algorithms is huge, and it is necessary to scan the original database several times. Therefore, the execution time is relatively long, and the storage space is large. In this paper, we propose a novel algorithm, called CoHUI-Miner, to reduce the space usage requirement, and improve the performance of the mining process for correlated high-utility itemsets (CoHUIs).

A. RESEARCH CONTRIBUTIONS

- It applies the projected database strategy to reduce the database size when exploring the search space, and proposes a new concept called the *prefix utility* of the projected transaction to directly calculate the utility of itemsets.
- It presents a new method, SearchCoHUI, to mine CoHUIs in a single phase without generating candidates.
- Several pruning strategies for mining CoHUIs are applied more efficiently to project the database, such as, TWU-Prune, KULC-prune, Variant of U-Prune and LA-Prune.
- The experimental results on dense, moderate and very dense datasets prove that the improved algorithm we proposed shows better performance than the state-of-the-art CoHUIM algorithm in both runtime and memory usage.

B. ORGANIZATION

The rest of this paper is organized as follows: Section 2 presents influential studies related to CoHUIs, and the important definitions of problem mining for CoHUIs. In Section 3 we propose the CoHUI-Miner algorithm to mine CoHUIs. Section 4 presents experimental evaluations of the proposed algorithm. In Section 5 we discuss the conclusions and give some expansions of the problem.

II. RELATED WORK

In this section, we briefly review prior works both on high-utility itemset mining and on correlated high-utility itemset mining, and the reason for mining correlated high-utility itemsets.

A. HIGH-UTILITY ITEMSET MINING

Recently, many studies have been carried out to mine HUIs, and have been applied to many useful applications that improve the quality of human life. Some previous works on utility mining problems use a two-phase approach and generate a large quantity of candidates, such as Two-Phase [13], CTU-Mine [27], TWU-Mining [28], UP-Growth [14], UP-Growth+ [29], which thus consume more execution time and storage space. To solve this problem, in 2012 Liu and Qu proposed an algorithm, called HUI-Miner [15], to discover HUIs with a list data structure (utility list), and a TWU pruning strategy. The HUI-Miner can reduce the search space because it mines HUIs without scanning the database again. Nonetheless, the performance of HUI-Miner can still be improved. Fournier *et al.* designed the FHM [16] algorithm and defined the Promising Utility Co-Occurrence Structure (PUCS) to reduce the number of candidate itemsets. As a result, FHM can be up to six times faster than HUI-Miner. Following that, many techniques were discovered to effectively exploit HUIs without candidate generation, including d2HUP [30] (proposed by Liu *et al.* in 2016) and EFIM [17] (proposed by Zida *et al.* in 2017). By integrating the two algorithms UP-Growth+ and FHM, Dawar *et al.* [20] proposed a hybrid algorithm UFH to effectively exploit HUIs on sparse databases in 2016. In 2018, Duong *et al.* [31] built a new utility-list buffer (ULB) structure that evolved from the previous utility-list structure and proposed the ULB-Miner (Utility-List Buffer for high utility itemset Miner) algorithm using an effective method for creating utility-list segments to reduce the cost of constructing utility-lists, so HUIs are mined more efficiently both in terms of runtime and memory consumption. Moreover, the problem is also extended to exploit High-Utility Sequential Patterns (HUSP) from a sequence database [32]–[34].

Taking advantage of the global optimization of evolutionary algorithms, many algorithms that exploit HUIs based on GA (Genetic Algorithm), PSO (Particle Swarm Optimization) and Bio-Inspired approaches have been proposed. In 2014, in order to reduce the search space and difficulty in determining the minimum utility threshold, Kannimuthu and Premalatha [35] proposed the HUPE_{UMU}-GRAM and $\text{HUP}_{\text{EWUMU}}$-GARM algorithms based on GA and presented results based on databases containing negative item values. In 2016, Lin *et al.* [36] applied the PSO algorithm to the proposed HUIM-BPSO algorithm by integrating the sigmoid strategy in the TWU update model and using the Or/NOR tree structure to trim invalid candidates to reduce the number of database scans.

Biology focuses on the study of organisms, their interrelationships and environments, so a bio-inspired algorithm is one based on measuring and duplicating the appearance or nature of a biological object. These algorithms achieve high performance but are not guaranteed to explore all the

HUIs. In 2018, Song and Huang [37] proposed a new model based on bio-inspired algorithms to find a set of HUIs. This approach selects the proportion of HUIs detected as the target values of the next population. The three algorithms were a genetic algorithm, particle swarm optimization, and bat algorithm. Extensive tests conducted on publicly available datasets show that the proposed algorithms outperform existing state-of-the-art algorithms in terms of efficiency, quality of results, and convergence speed.

In 2019, Zhang *et al.* [38] presented three new strategies in the HUIM-IGA algorithm, namely the neighbor search strategy to search for HUIs more effectively, the strategy of individual diversity in the population to solve the loss problem of HUIs, and the strategy of adjusting individuals to limit the generation of invalid candidates, with all three combined to ensure that HUIs are preserved. Experiments have shown that the HUIM-IGA algorithm is more effective than previous EC-applied algorithms with regard to time consumption.

B. CORRELATED HIGH-UTILITY ITEMSET MINING

The HUPM algorithm can find HUPs, but many HUPs may be unnecessary due to poor correlations. High-utility pattern mining (HUPM) has the major drawback that it only considers the utility of itemsets but ignores the affinity between items in an itemset. Many studies have worked to calculate the correlation between items in an itemset, with researchers proposing various scales to measure the correlation.

The concept of measurements between items was also proposed by Omiecinski [6] to mine associations in databases. Omiecinski presented three interestingness measures for associations: any-confidence, all-confidence, and bond. These measures describe the strength of items with regard to an association with other related items. He also demonstrated the importance of the downward closure property to both all-confidence and bond metrics, and proposed an algorithm to exploit association rules based on these. However, these measures do not evaluate the correlation among items in a transaction database that contains many unstable and null transactions. In 2010, Wu *et al.* [3] re-evaluated a collection of five null-invariant scales and indicated the stereotype and linear ordering between them. They also presented a new concept, namely the imbalance ratio, to assess the degree of deviation for a dataset and proposed the GAMiner algorithm to exploit frequent and closely related Kulc and Cosine itemsets. In another study, Duan and Street [39] proposed finding maximal fully-correlated itemsets in large databases by building some key properties when choosing the best correlation measure, and gave a complete definition of a fully correlated itemset. Finding the best correlation measure combined with the definition of fully-correlated itemsets eliminated itemsets with irrelevant items in a reasonable time.

In 2011, Ahmed *et al.* [40] proposed an algorithm, high-utility interesting pattern mining (HUIPM) with a frequency affinity, to explore interesting itemsets in which the relations among items are meaningful. They presented the UTFA structure, which is a utility tree based on frequency affinity for the single-pass mining of HUIPs. In 2017, Lin *et al.* [25] proposed the element-information table (EI-table) and frequency-utility tree (FU-tree) structures and some pruning strategies based on them to discovery discriminative high-utility patterns (DHUPs) without candidate generation using a strong frequency affinity HUIM algorithm, called FDHUP. The algorithm performs considerably better than the UIPM algorithm in terms of runtime, memory usage and scalability.

In 2018, Fournier-Viger *et al.* [41] proposed combining the concept of correlation and high-utility itemset mining to find the correlated high-utility itemsets (CHIs). The FCHM (fast correlated high-utility itemset miner) algorithm was presented to mine CHIs. This algorithm has two versions, FCHMall-confidence based on all-confidence and FCHMbound based on the bond measure. The algorithm integrates several strategies to discover CHIs efficiently. The results of their experiment were evaluated using the min_measure and min_util thresholds. This algorithm was also compared to the FHM algorithm for HUIM with regard to runtime and memory usage. More recently, Fournier-Viger *et al.* [42] proposed the mining of LHUIs (local high-utility itemsets) and expanded it to seek PHUIs (peak high-utility itemsets), with the algorithms LHUI-Miner and PHUI-Miner being proposed to find these patterns. However, the set of PHUIs can be quite large, and some items emerging in PHUIs do not contribute much to their peaks. NPHUIs (non-redundant peak high-utility itemsets) were thus proposed to exploit a smaller set of patterns. LHUI, PHUI and NPHUI are completely new issues in HUI mining.

Djenouri *et al.* [43] introduced a highly effective pattern mining approach, namely Clustering-Based Pattern Mining (CBPM). This study included two steps: first, clustering the transaction database into clusters in which highly correlated transactions are grouped together by the K-Means algorithm, and then applying the HUIM algorithm to each cluster to discover relevant HUIs. An experimental evaluation of CBPM showed that it was efficient, especially in a large search space. Moreover, Gan *et al.* [44] proposed an efficient utility mining approach that considers the correlation of items among patterns, called non-redundant CoUPM (Correlated high-Utility Pattern Miner). This algorithm performed on a one-phase basis on the utility-list and revised this structure to contain data on utility and correlation. The downward closure properties of Kulc and remaining utility were applied for pruning in this algorithm. The design and experimental results of this algorithm were compared with those of the HUPM and CoHUIM algorithms.

III. PRELIMINARIES AND PROBLEM STATEMENT

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set containing m different items in a transaction database $D = \{T_1, T_2, \ldots, T_n\}$, where n is the number of records in D. $\forall T_j \in D$, $T_j = \{x_l | l = 1, 2, \ldots, N_j, x_l \in I\}$, where N_j is the number of items in transaction T_j.

An example of transaction database D is given in Table 1 and item profits are given in Table 2.

TABLE 1. A given database.

TID	Transaction	Purchase quantity (Pq)	Utility (U)	Transaction Utility (u_τ)
T_1	a, b, c, d	2, 2, 1, 1	8, 6, 1, 2	17
T_2	a, b, d, e, f	1, 1, 2, 4, 2	4, 3, 4, 12, 2	25
T_3	a, c, d	3, 5, 2	12, 5, 4	21
T_4	b, c, e	2, 5, 2	6, 5, 6	17
T_5	b, c, d, e	3, 3, 2, 7	9, 3, 4, 21	37
T_6	a, b, c, d, e, f	1, 1, 4, 2, 5, 2	4, 3, 4, 4, 15, 2	32
T_7	c, e	4, 5	4, 15	19
T_8	c, d, e	1, 2, 3	1, 4, 9	14

TABLE 2. Profits of items.

Item	a	b	c	d	e	f
Profit \$ per unit (Prf)	4	3	1	2	3	1

Each item x_i in I is assigned a profit value denoted as $Prf(x_i)$. Each item x_i in transaction T_j is assigned a purchase quantity value and denoted as $Pq(x_i, T_j)$. The utility of an item x_i in a transaction T_j is denoted as $u(x_i, T_j)$, and defined as $u\left(x_i, T_j\right) = Prf\ (x_i) * Pq(x_i, T_j)$. The utility of an itemset $X = \{x_1, x_2, \ldots, x_k\} \subseteq T_j$ is denoted as $u\left(X, T_j\right) = \sum_{x_i \in X} u(x_i, T_j)$. The utility of an itemset X in database D is defined as $u\left(X\right) = \sum_{X \subseteq T_j T_j \in D} u(X, T_j)$. The transaction utility in database D is specified as $u_\tau\left(T_j\right) = \sum_{x_i \in T_j} u(x_i, T_j)$.

Definition 1: HUI is an itemset X whose utility is larger than the minimum utility threshold (minUtil) given by the user. HUIs is a set of all HUI in D and is defined as:

$$HUIs = \{X | u(X) >= minUtil\}$$

Definition 2: The transaction–weighted utility of X in D is defined as $twu(X) = \sum_{X \subseteq T_j \in D} u_\tau\left(T_j\right)$. *For example, in Table 1,* $twu\left(ac\right) = u_\tau\left(T_1\right) + u_\tau\left(T_3\right) + u_\tau\left(T_6\right) = 17 + 19 + 32 = 68$. *Table 3 shows twu of 1 - itemsets in D.*

TABLE 3. Transaction-weighted utility and support for 1 - itemsets.

Items	f	a	b	d	e	c
twu	57	95	128	146	144	157
Support	2	4	5	6	6	7

Property 1 [13]: If $twu\left(X\right) < minUtil$ *then no of supersets of X are a HUI.*

Definition 3: The support of X is the quantity of transactions which contain X, denoted as sup(X). For example, in Table 1, $sup(ac) = 3$ *because three transactions* T_1, T_3 *and* T_6 *contain {ac}. The support for each item is shown in Table 3.*

In 2018, the first correlated high-utility itemset mining (CoHUIM) algorithm was proposed by Gan *et al.* [26] to mine items in every HUI that has strong correlation. The algorithm uses a correlation measure and applies the sorted downward closure property of Kulc (abbreviated as Kulc - correlation measure) to efficiently prune unpromising candidates.

Definition 4: (Ordering of items). A total order is built on the increasing support of items in D (in Table 3). For the database given in Table 1, the total order is: $f \prec a \prec b \prec d \prec e \prec c$. *Selecting* $minUtil = 70$ *and above total order* $\prec$, *the database has been changed as shown in Table 4.*

TABLE 4. The ordering of purchases sorted by support.

TID	Transaction	Purchase quantity (Pq)	Utility (U)	Transaction Utility (u_τ)
T_1	a, b, d, c	2, 2, 1, 1	8, 6, 2, 1	17
T_2	a, b, d, e	1, 1, 2, 4	4, 3, 4, 12	23
T_3	a, d, c	3, 2, 5	12, 4, 5	21
T_4	b, e, c	2, 2, 5	6, 6, 5	17
T_5	b, d, e, c	3, 2, 7, 3	9, 4, 21, 3	37
T_6	a, b, d, e, c	1, 1, 2, 5, 4	4, 3, 4, 15, 4	30
T_7	e, c	5, 4	15, 4	19
T_8	d, e, c	2, 3, 1	4, 9, 1	14

There is a large number of HUIs that have items whose correlation is weak, not strong. Many measures have been proposed to evaluate the HUIs' correlation such as affinitive frequency, all-confidence, bond, any-confidence, and Kulczynsky (Kulc). However, Kulc is widely used to assess the inherent correlation of itemsets.

Definition 5: The correlation between items in the itemset $X = \{x_1, x_2, \ldots, x_k\}$ *is defined as* $kulc\left(X\right) = \frac{1}{k}\left(\sum_{x_i \in X} \frac{Sup(X)}{\sup(x_i)}\right)$. *The value of kulc is within the range* [0, 1] *[26].*

$Sup(X)/Sup(x_i)$ is the ratio of the number of appearances in the same transaction to all other items in X to the number of times the item x_i appears during the transaction. If this ratio is high, it proves that x_i is highly associated with all items in X. In contrast, *kulc* is the average of the above ratios, so if the ratio is high, the items in X appear together many times and are highly related.

For example, $kulc\left(ac\right) = \frac{1}{2}\left(\frac{\sup(ac)}{\sup(a)} + \frac{\sup(ac)}{\sup(c)}\right) = \frac{1}{2}\left(\frac{3}{4} + \frac{3}{7}\right) = 0.5893$.

Property 2 [26]: On the total order of all items in a transaction database, assume that $x_1 \prec x_1 \prec \ldots \prec x_k \prec x_{k+1}$, *the values of kulc are sorted in order of downward closure as* $kulc\left(x_1, \ldots, x_{k+1}\right) \leq kulc\left(x_1, \ldots, x_k\right)$.

Definition 6: A correlated high-utility itemset (CoHUI) is an itemset X in D which is $kulc\left(X\right) \geq minCor$ *and* $u\left(X\right) \geq minUtil$). *CoHUIs is a set of all CoHUI in D and is defined as:*

$$CoHUIs = \{X \subseteq I | u\left(X\right) \geq minUtil \wedge kulc(X) \geq minCor\}$$

For instance, in Table 4, $u\left(bde\right) = 75$, $kulc\left(bde\right) = 0.5333$. If the $minUtil = 70$ and the $minCor = 0.52$ then the itemset $\{bde\}$ is a CoHUI.

IV. COHUI-MINER ALGORITHM

A. DATABASE PROJECTION STRATEGY

To reduce the database size while exploring the search space, we use the projected database strategy [17], which significantly reduces the time needed to scan the database. Therefore, the process of mining for CoHUIs becomes more effective. The details of the projection mechanism are presented below.

Definition 7: The item x_i is after itemset X in T_j if x_i is after all items in X. In this, all items in T_j are a set of ascending sorted order support-values and are denoted as $X \prec x_i$.

For instance, in the transaction T_6 in Table 4, $\{a, b, d\} \prec$ e, $\{a, e\} \prec c$.

Definition 8: The projected transaction T_j on itemset X is denoted as $T_j\backslash X$, and $T_j\backslash X = \{x_i \in T | X \prec x_i\}$.

For instance, in Table 4, the projected transaction T_6 on itemset $\{a, e\}$ is $T_6\backslash\{a, e\} = \{c\}$.

Definition 9: The projected database on itemset X is denoted as $D\backslash X$, and $D\backslash X = \{T_j\backslash X | T_j \in D\}$.

For instance, in Table 4, the projected database on itemset $X = \{a, b\}$: itemset X is contained in the three transactions: T_1, T_2, and T_6. Then, $D\backslash\{a, b\}$ is $T_1\backslash\{a, b\} = \{d, c\}$, $T_2\backslash\{a, b\} = \{d, e\}$, and $T_6\backslash\{a, b\} = \{d, e, c\}$.

TABLE 5. The projected database on X = {a, b}.

TID	Transaction
T_1	d, c
T_2	d, e
T_6	d, e, c

The above example shows that the projected database $D\backslash\{a, b\}$ is a lot smaller than the original database, thus increasing the effectiveness of the CoHUI mining process. Because $D\backslash\{a, b\}$ has removed $\{a, b\}$, $U\{a, b\}$ cannot be directly calculated on this projected database. Gan *et al.* [26] put $\{a, b\}$ into the candidate set if $kulc(a, b) \geq minCor$. After a large quantity of candidates was found in phase 1, they continued to scan the original database in phase 2 to compute $U\{a, b\}$. Therefore, this algorithm required a long execution time and consumed a large amount of memory. To solve these issues, we propose the *prefix utility* concept.

Definition 10: The prefix utility of the projected transaction $T_j\backslash X$ is defined as $pru\left(T_j\backslash X\right) = u\left(X, T_j\right)$.

For instance, in Table 4, $pru(T_1\backslash ab) = u(ab, T_1) = 8 + 6 = 14$, $pru(T_2\backslash ab) = u(ad, T_2) = 4 + 3 = 7$, $pru(T_6\backslash ab) = u(ab, T_6) = 4 + 3 = 7$.

Thus, when projecting a transaction T_j on itemset X, the utility of X on T_j will be calculated and saved by the *prefix utility*. Different transactions may result in different prefix utility values. The prefix utility of a transaction will change in different projections. In the projected database $D\backslash X$, the itemset X' is extended from X and its utility is identified based on the utility of X and the *prefix utility* of the transactions in $D\backslash X$. Combined with the *kulc* value, our proposed algorithm helps determine whether X' is CoHUI or not in the projected database without re-scanning the original database. The CoHUI-Miner algorithm is shown in Algorithm 1.

TABLE 6. The projected database on X = {a, b} with prefix utility.

TID	Transaction	Prefix Utility
T_1	d, c	14
T_2	d, e	7
T_6	d, e, c	7

Definition 11: The remaining utility of X in T_j (denoted as $ru(X, T_j)$) is sum of the utility of all items which are after X in T_j by the total order,

$$ru(X, T_j) = \sum_{x_i \in T_j \wedge X \prec x_i} u(x_i, T_j)$$

For instance, in Table 4, $ru(ab, T_1) = u(d, T_1) + u(c, T_1) = 2 + 1 = 3$.

Definition 12: The remaining utility of X in D is defined as

$$ru(X) = \sum_{X \subseteq T_j \in D} ru(X, T_j).$$

For instance, $ru(ab) = ru(ab, T_1) + ru(ab, T_2) + ru(ab, T_6) = 3 + 16 + 23 = 42$.

B. THE COHUI-MINER ALGORITHM

Some pruning strategies are applied to reduce the search space and execution time of the algorithm, as follows. With strategy 1 (TWU-Prune), in the first scan of the database, the *twu* values of the 1-itemsets X are calculated. According to property 1, if $twu(X)$ is smaller than $minUtil$, X and all its supersets are not HUIs and also are not CoHUIs [13] (because a CoHUI's prerequisite is HUI). So, the algorithm stops expanding from X. With strategy 2 (KULC-prune), based on the total order $\prec$ of all items in the database, if an itemset X has $kulc\ (X) < minCor$, every itemset Y expanded from X also has $kulc\ (Y) < minCor$ (for $kulc(Y) \leq kulc(X)$) (property 2). This means that X is not a CoHUI and any extended itemsets from the prefix X are not CoHUIs. With strategy 3 (Variant of U-prune), the use of projection will significantly reduce the size of the database during algorithm implementation. The larger the extended set is, the smaller the database size after the projection is made, so that the search space will be effectively reduced based on the projection database by itemset X. The utility and the remaining utility values of an itemset X'(extended from X) are then calculated. If $u(X') + ru(X') < minUtil$, then all the itemsets Y extended from X' are not HUIs [15] and are also not CoHUIs. With strategy 4 (LA-Prune), in the process of extending itemset X' from X by projection method, the ULA of X' value is initialized by $U\ (X) + RU(X)$. The projection will consider all transactions in the input database. If $X' \not\subset T$, then reduce the ULA value by $pru\ (T) + u_\tau(T)$. If $ULA < minUtil$ then X' is not a CoHUI and all itemsets that extend from X' are not CoHUI. The projection with itemset X' will stop.

Algorithm 1 CoHUI-Miner

Input: D: transaction database; *minCor*; *minUtil*.
Output: CoHUIs: set of all correlated HUIs.

1 Scan database D for calculating $SUP(i)$, $TWU(i)$ and $U(i)$ for every item i in the database.
2 Construct $I_{keep} = \{i \in I | TWU(i) \geq minUtil\}$.
3 Update SUP, U and database D:
$SUP = \{SUP(i) | i \in I_{keep};\ U = \{U(i) | i \in I_{keep}\}$;
Eliminate i from database D if $i \notin I_{keep}$
4 Sort I_{keep} in the increasing order of SUP, and sort items of all transactions in database D with respect to I_{keep}
5 **for** *each 1-item* $X \subseteq I_{keep}$ **do**
6 **if** $U(X) \geq minUtil$ **then** // Correlation values are
7 $CoHUIs \leftarrow X$; //always 1 because $|X| = 1$
8 **end if**
9 *set* $RU(X) = 0$; //remaining utility of X
10 **for** *each transaction* $T \in D$ **do**
11 *set* $j = 0$; $x_j \in T$; *set* $uTemp = u_\tau(T)$;
12 **while** $j < |T|$ ***and*** $x_j \prec X$ **do**
13 *Decrement* $uTemp$ by $u\left(x_j, T\right)$;
14 *Increment* j *by 1;*
15 **end while**
16 **if** $j = |T|$ ***or*** $X \prec x_j$ **then** *Continue;*
17 **else if** $j < |T|$ **then**//Calculate projected transaction
18 Initial $T \backslash X = T.Get(j+1, |T|)$;
19 $pru(T \backslash X) = u\left(x_j, T\right)$;
20 $u_\tau(T \backslash X) = uTemp$;
21 $dbProjectX.add(newTran)$;
22 Increment $RU(X)$ by $uTemp$;
23 **end if**
24 **end for**
25 *SearchCoHUI*$(X, U(X), RU(X), dbProjectX, 1)$;
26 **end for**

Algorithm 2 SearchCoHUI

Input: X: prefix itemset; $U(X)$: utility of X, $RU(X)$: the remain utility of X, $dbProjectX$: projected database with X prefix; k: length of items set X.
Output: itemsets are CoHUIs with X prefix.

1 **for** $i = k$ *to* $|I_{keep}|$ **do**
2 $LastItem = I_{keep}[i]$
// *LastItem is an item that is extended from X itemset.*
3 $X' = X \cup LastItem$;
4 Initial $U(X') = U(X)$; $RU(X') = 0$;
$SUP(X') = 0$; $ULA = U(X) + RU(X)$;
5 **for** each transaction $T \in dbProjectX$ **do**
6 set $j = 0$; $x_j \in T$;
7 set $uTemp = u_\tau(T)$;
8 **while** $j < |T|$ **and** $x_j \prec LastItem$ **do**
9 Decrement $uTemp$ by $u\left(x_j, T\right)$;
10 Increment j by 1;
11 **end while**
12 **if** $j = |T|$ **or** $LastItem \prec x_j$ **then**
13 Decrement $U(X')$ by $pru(T)$;
14 Decrement ULA by $(pru(T) + u_\tau(T))$;
15 **if** $ULA < minUtil$ **then** *return;*//LA-Prune
16 *Continue;*
17 **else**
18 Increment $U(X')$ by $u\left(x_j, T\right)$;
19 Increment $SUP(X')$ by *1;*
20 **if** $j < |T|$ **then**
//Calculate projected transaction with X'
21 $T \backslash X' = T.Get(j+1, |T|)$;
22 $pru(T \backslash X') = pru(T) + u\left(x_j, T\right)$;
23 $u_\tau(T \backslash X') = uTemp$;
24 $dbProjectX'.add(newTran)$;
25 *Increment* $RU(X')$ by $uTemp$;
26 **end if**
27 **end if**
28 **end for**
29 **if** $SUP(X') > 0$ **then**
30 Calculate $kulc(X')$//by the formula in definition 5
31 **if** $kulc(X') \geq minCor$ **then**
32 **if** $U(X') \geq minUtil$ **then**
33 $CoHUIs \leftarrow X'$;
34 **end if**
35 **if** $U\left(X'\right) + RU(X') \geq minUtil$ **then** //U-Prune
36 $SearchCoHUI(X', U\left(X'\right), RU(X'), dbProjectX', k+1)$;
37 **end if**
38 **end if**
39 **end if**
40 **end for**

The procedure of CoHUI-Miner takes as its input D, a transaction database; *minCor*, user-specified minimum correlation threshold; and *minUtil*, a user-specified minimum utility threshold. It mainly performs in one phase. It first scans the database to calculate $sup(i)$, $twu(i)$ and $u(i)$ for each item i in D (line 1), then constructs the I_{keep} set which consists of items whose twu value is not less than minUtil and finally updates SUP, U and database D with respect to I_{keep} (lines 2, 3). After that, I_{keep} is sorted in the increasing order of SUP, and the items of all transactions in database D is sorted w.r.t I_{keep} (line 4). In line 5, each 1-item $X \subseteq I_{keep}$, if $U(X) > minUtil$ then X is a CoHUI (because $kulc(X) = 1$) (dismiss lines 6, 7). From lines 9 to 24, the algorithm creates the projected database from 1-item X. Each projected transaction is assigned a prefix-utility of itemset X (line 19). In line 25, the SearchCoHUI procedure is called to extend the CoHUI set.

The **SearchCoHUI** algorithm takes X : prefix itemset, $U(X)$: utility of X, $RU(X)$: remaining utility of X, $dbProjectX$: projected database with X prefix, and k: length of items set X. This is based a DFS algorithm to find the extended set X' having the prefix X, then calculate $U(X')$, $RU\left(X'\right)$, $support(X')$, $kulc(X')$ and projected database by X'.

The calculation $U(X')$ will be based on $U(X)$ and the *prefix utility* of the transactions. Based on the $U(X')$ and $kulc(X')$ values, we can determine whether X' is a CoHUI or not. With each item in I_{keep} (starting from position k), the procedure finds the extended set X' of X (lines 2, 3). Initializations for $U(X')$, $RU(X')$, $Support(X')$ and $ULA(X')$ are done in line 4. From lines 5 to 28, the algorithm processes each transaction T of the database *dbProjectX* to calculate $U(X')$ (if $X' \subseteq T$ then increase $U(X')$ by $u\left(x_j, T\right)$ and update $RU(X')$, $support\left(X'\right)$; else decrease $U\left(X'\right)$ *by* $pru\,(T)$. At lines 1 and 15, the LA-Prune strategy is applied, if $X' \not\subseteq T$ then the ULA value is decreased by $(pru\,(T) + u_\tau\,(T))$, and if $ULA < minUtil$ then X' is not a CoHUI so stop the projection with X' by return command. Lines 20 to 26 calculate *projected transaction* T on $X'(T \backslash X')$ and update *prefix utility* corresponding, $pru(T \backslash X') = pru(T) + u\left(x_j, T\right)$. In lines 29 to 39, $kulc\left(X'\right)$ is computed and if $kulc(X') \geq minCor$ and $U(X') \geq minUtil$ then X' is a CoHUI. At lines 35 and 36, U-Prune strategy is applied, if $U\left(X'\right) + ru(X') \geq minUtil$, **SearchCoHUI** is called recursively to continue searching for CoHUI with the prefix X'. Otherwise this algorithm is ended.

V. EXPERIMENTS

A. GENERAL SETTINGS

We implemented the CoHUI-Miner algorithm in Java and conducted it on a Dell Precision Tower 3620 with Intel Core i7-7800X CPU @3.5GHz, 32GB of memory, running Windows 10. We used standard datasets downloaded from the SPMF library [45] such as Chainstore, Kosarak, Accident, Mushroom, Chess and Connect. Detailed characteristics of experimental datasets are given in Table 7 with both dense, moderately dense and sparse datasets.

TABLE 7. Dataset characteristics.

Dataset	Transactions	Distinct items (I)	Avg Len (A)	Density (A/I) %
Chess	3,196	75	37	49.3333
Connect	67,557	129	43	33.3333
Mushroom	8,124	119	23	19.3277
Accident	340,183	468	33.8	7.2222
Kosarak	990,000	41,270	8.1	0.0196
Chainstore	1,112,949	46,086	7.3	0.0158

The obtained results show that the CoHUI-Miner algorithm has better execution time than CoHUIM [26] in all datasets from Table 7. The major difference between the two algorithms is that CoHUIM generates the candidate set in phase 1 and rescans the dataset several times in phase 2 to discover CoHUIs, whereas CoHUI-Miner uses the prefix utility of the projected transactions to define the utility of an itemset without generating candidates. Furthermore, we apply two effective pruning strategies, *Variant of U-Prune* and *LA-Prune*, to the projected database to reduce the search space. The CoHUI-Miner algorithm was used on three models to compare with the CoHUIM algorithm:

Basic algorithm: The single-phase CoHUI-Miner algorithm uses the prefix utility value, applies *KULC-prune* and *TWU-Prune* strategies to reduce the search space. The CoHUIM algorithm that we compared also adopts these strategies.

Improved algorithm: To increase the performance of the basic algorithm, the *Variant of U-Prune* and *LA-Prune* strategies are used to evaluate our more efficient proposed algorithm, including CoHUI-Miner$_{RU}$ (Basic algorithm + *U-Prune*), CoHUI -Miner$_{RU+LA}$(Basic algorithm + *U-Prune* + *LA-Prune*).

B. COMPARISON OF RUNTIME AND MEMORY USAGE

Figures 1, 2 and 3 present the runtime effectiveness of CoHUI-Miner on three models, namely CoHUI-Miner (basic algorithm), CoHUI-Miner$_{RU}$ and CoHUI-Miner$_{RU+LA}$. As the density of the dataset increases, the cost of dataset scanning is also increased. Thus, CoHUIM shows that it suffers from excessive dataset scanning in phase 2. The experimental performance in CoHUI-Miner is significantly better than CoHUIM on all dense datasets.

In a very dense database (Fig. 1.), all the models of the CoHUI-Miner algorithm are much more efficient than the CoHUIM in execution time. In particular, the smaller the *minUtil* and *minCor* are, the more obvious the effect is. In some cases, when the Connect database is run on low *minUtil* thresholds, the CoHUIM algorithm could not find out the CoHUI sets. However, CoHUI-Miner still performs effectively and does not need much time. CoHUIM is a two-phase algorithm. In phase 1, the CHUUBI set is built to save candidates which satisfy the *minCor* threshold. Phase 2 rescans the database to calculate the candidates' utility and defines CoHUI from the candidates. Meanwhile, CoHUI-Miner is a single-phase algorithm, which calculates the correlated value between items in itemsets and their utility immediately when determining CoHUI itemsets. CoHUI-Miner is also more efficient than the CoHUIM algorithm because it does not spend a lot of time to determine the utility of candidates in phase 2. In dense databases, the correlation between items is high, and the TWU of itemsets is also large, so the number of candidates is huge. Therefore, the CoHUI-Miner takes much less time than the CoHUIM. The CoHUI-Miner$_{RU}$ algorithm (applying the *Variant of U-Prune* strategy) is better than the basic CoHUI-Miner algorithm at all *minCor* and *minUtil* thresholds. The CoHUI-Miner$_{RU+LA}$ algorithm (applying both *U-Prune* and *LA-Prune* strategies) has the best results in all cases of the different types of databases. This result demonstrates that both pruning strategies play a very meaningful role in exploiting CoHUI.

The execution times of the CoHUI-Miner and CoHUIM algorithms on moderately dense databases are shown in Fig. 2 at various thresholds of *minCor* and *minUtil*. The results show that all the models of CoHUI-Miner are more efficient than the CoHUIM, especially in the situation of small *minCor* and *minUtil*. With regard to the runtime for the Accident dataset and $minCor = 0.62$, the

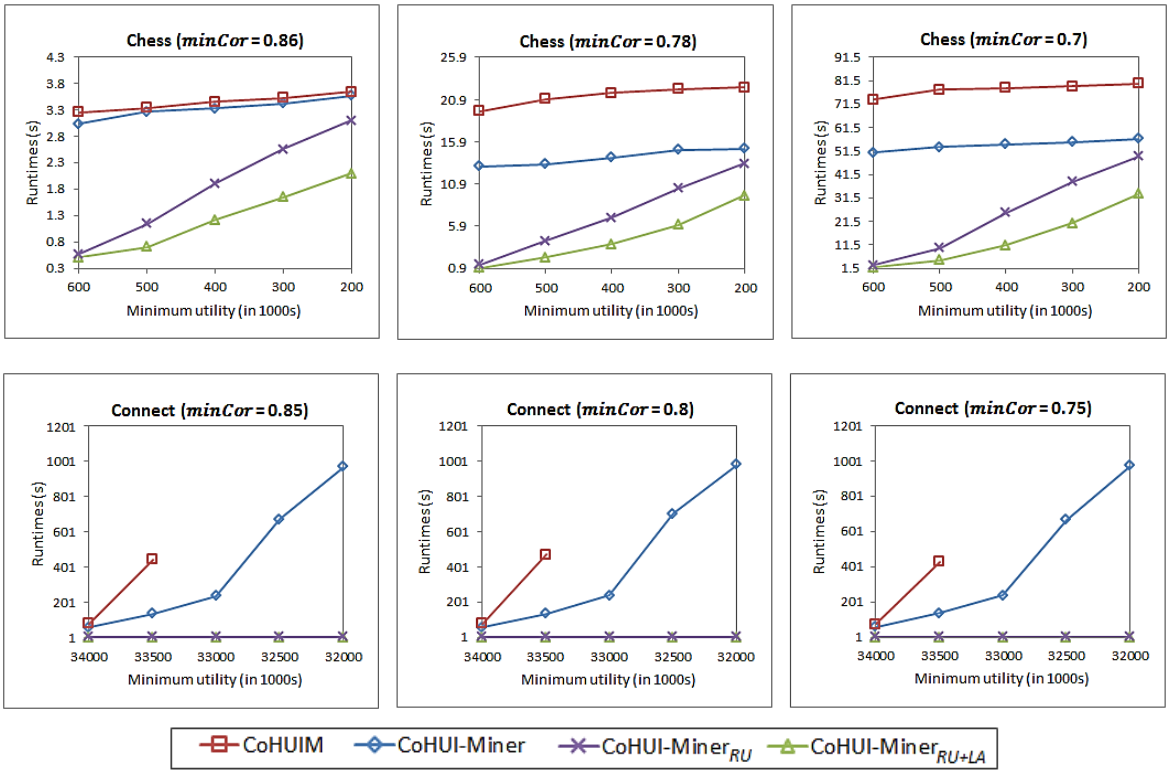

FIGURE 1. Runtime evaluations of the CoHUI-Miner and CoHUIM on very dense datasets.

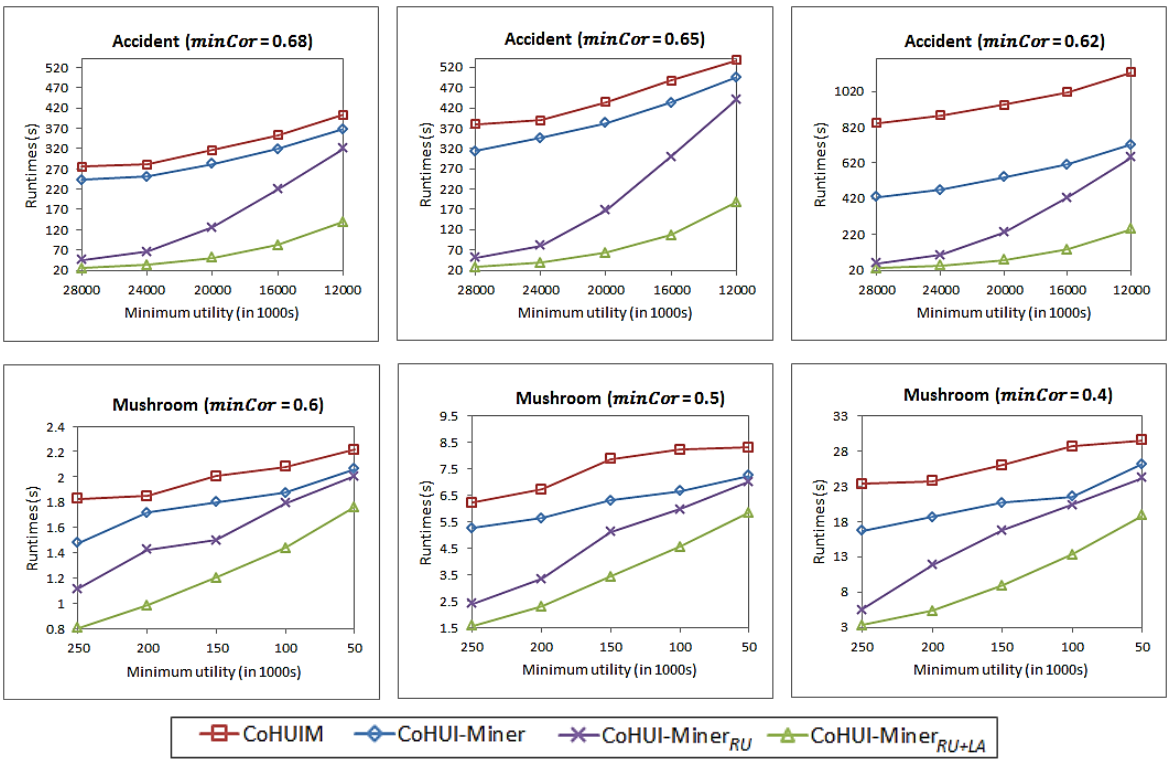

FIGURE 2. Runtime evaluations of the CoHUI-Miner and CoHUIM on moderately dense datasets.

proposed method, basic CoHUI-Miner, is two times faster than the CoHUIM. Moreover, the CoHUI-Miner$_{RU}$ algorithm (using the *U-Prune* strategy) has an effective execution time at high *minUtil* thresholds. For example, at *minUtil* =

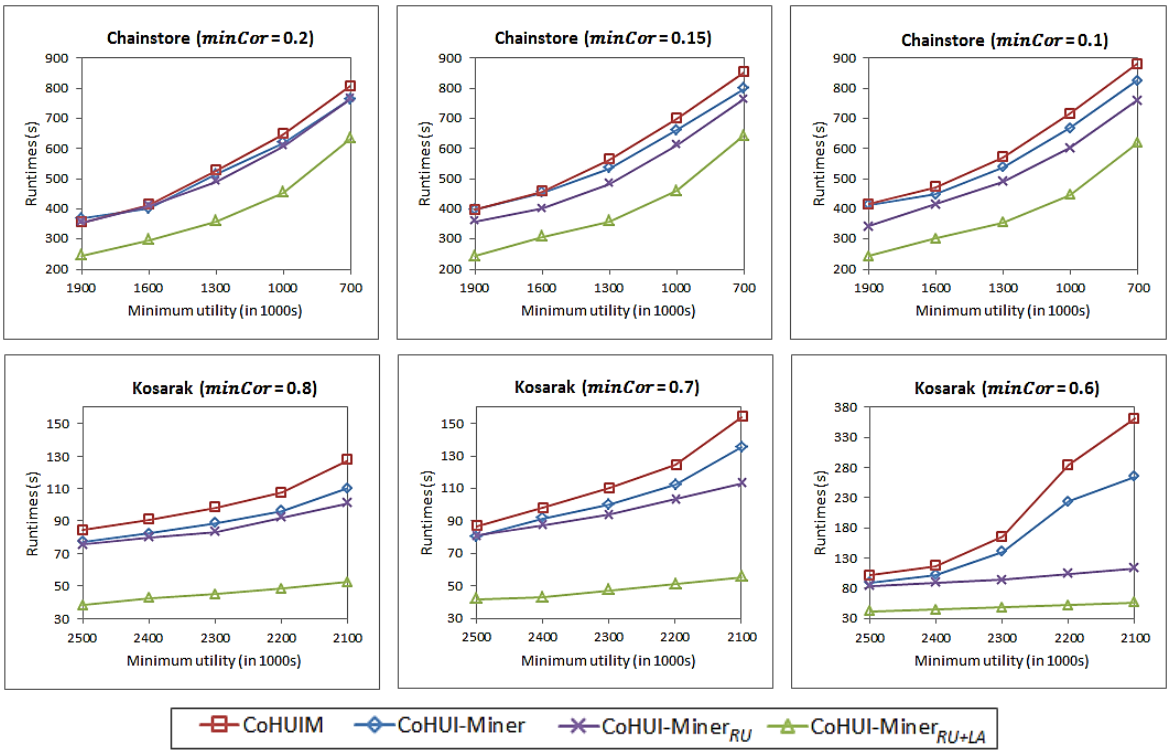

FIGURE 3. Runtime evaluations of the CoHUI-Miner and CoHUIM on very sparse datasets.

28, 000, 000, the execution time of CoHUI-Miner$_{RU}$ is about seven times faster compared to the basic CoHUI-Miner algorithm and about 14 times faster compared to CoHUIM. The execution time of the CoHUI-Miner$_{RU+LA}$ algorithm is the fastest in all case of the *minUtil* thresholds, about 14 times faster than the basic CoHUI-Miner algorithm and 28 times faster than the CoHUIM algorithm. In Fig. 2, showing the runtimes with the Mushroom dataset, also shows the similar results. This experimental results demonstrate that the *U-Prune* and *LA-Prune* strategies combined with the *prefix utility* value are effective on moderately dense datasets.

Figure 3 shows the runtimes of CoHUI-Miner and CoHUIM on very sparse datasets. In these the items' correlation is very weak, so the number of candidates in the CHUUBI is rather small. Although the CoHUIM works in two phases, performing phase 2 to determine the utility for each candidate by rescanning the database will take less time. Therefore, the execution time of basic the CoHUI-Miner algorithm is faster than CoHUIM, but not significantly. However, the combination of the *U-Prune* and *LA-Prune* strategies shows significantly improvedperformance. For the Chainstore dataset, the average execution time of the CoHUI-Miner$_{RU+LA}$ algorithm is faster than those of the basic CoHUI-Miner and CoHUIM algorithms by about 1.2 and 1.5 times, respectively. Moreover,, for the Kosarak dataset, the smaller the *minUtil* threshold is, the more effective CoHUI-Miner$_{RU}$ and CoHUI-Miner$_{RU+LA}$ are.

In Fig. 4, we make a comparison of memory usage between the CoHUI-Miner and CoHUIM algorithms on very dense datasets, such as Chess and Connect. With these datasets, the memory consumptions of all the CoHUI-Miner models are much less than that of the CoHUIM. For instance, with the Chess dataset, $minCor = 0.7$ and all of the *minUtil* thresholds, the memory usage among the models of CoHUI-Miner is not much different, at about 550MB. Meanwhile, the memory usage of the CoHUIM algorithm is 6,800MB, which is 12 times higher. In addition, the memory usage of CoHUI-Miner$_{RU+LA}$ algorithm for Connect is the lowest, while the CoHUIM algorithm is overloaded at $minUtil < 33,500$. This result proves that the CoHUI-Miner uses memory effectively with very dense datasets because these have a large number of candidates, and the application of the projected database strategy and the prefix utility structure has a clear effect on shrinking the search space. Moreover, the *U-Prune* and *LA-Prune* strategies are very effective. These eliminate a large number of unpromising itemset early, and thus the memory usage reduces significantly.

Next, the algorithms are implemented on moderately dense databases (Fig 5.), and all the CoHUI-Miner models still consume less memory than the CoHUIM. On the Accident dataset, the memory usage between the upgraded algorithm models is nearly the same but lower than CoHUIM's at *minCor* thresholds: it is about two times lower with $minCor = 0.68$, 2.5 times lower with $minCor = 0.65$ and three times lower with $minCor = 0.62$ respectively.

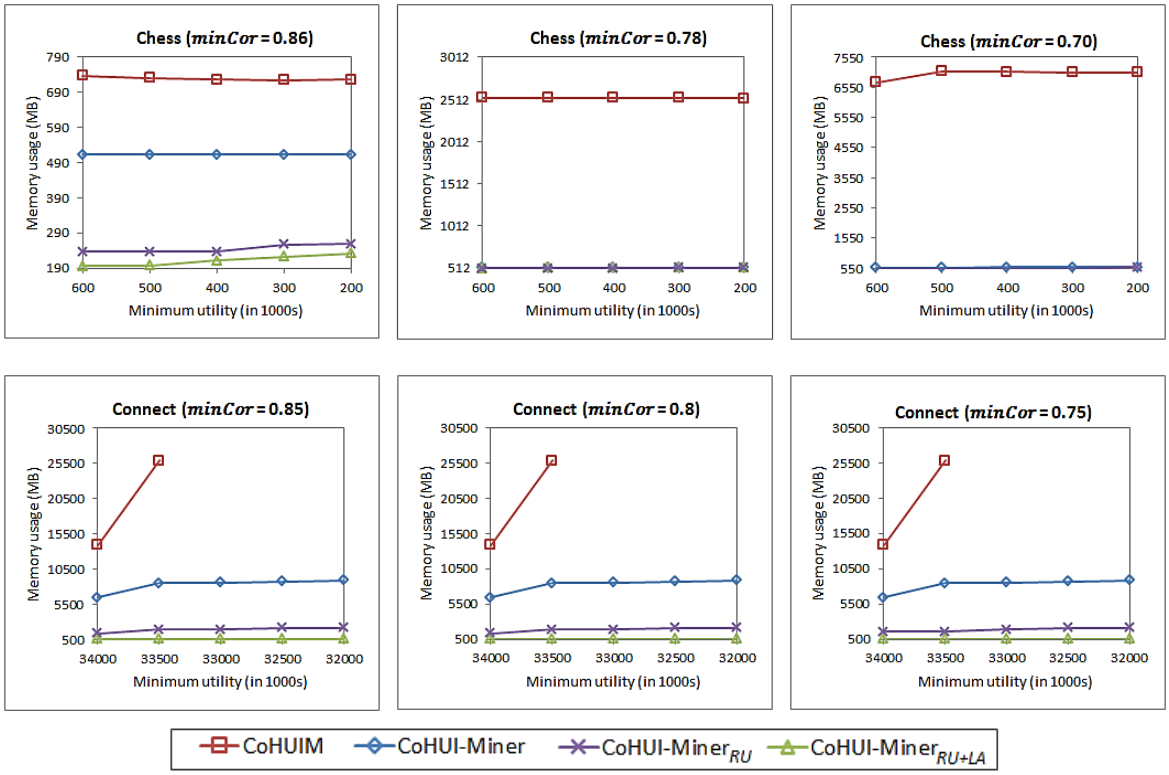

FIGURE 4. Memory consumption of the CoHUI-Miner and CoHUIM on very dense datasets.

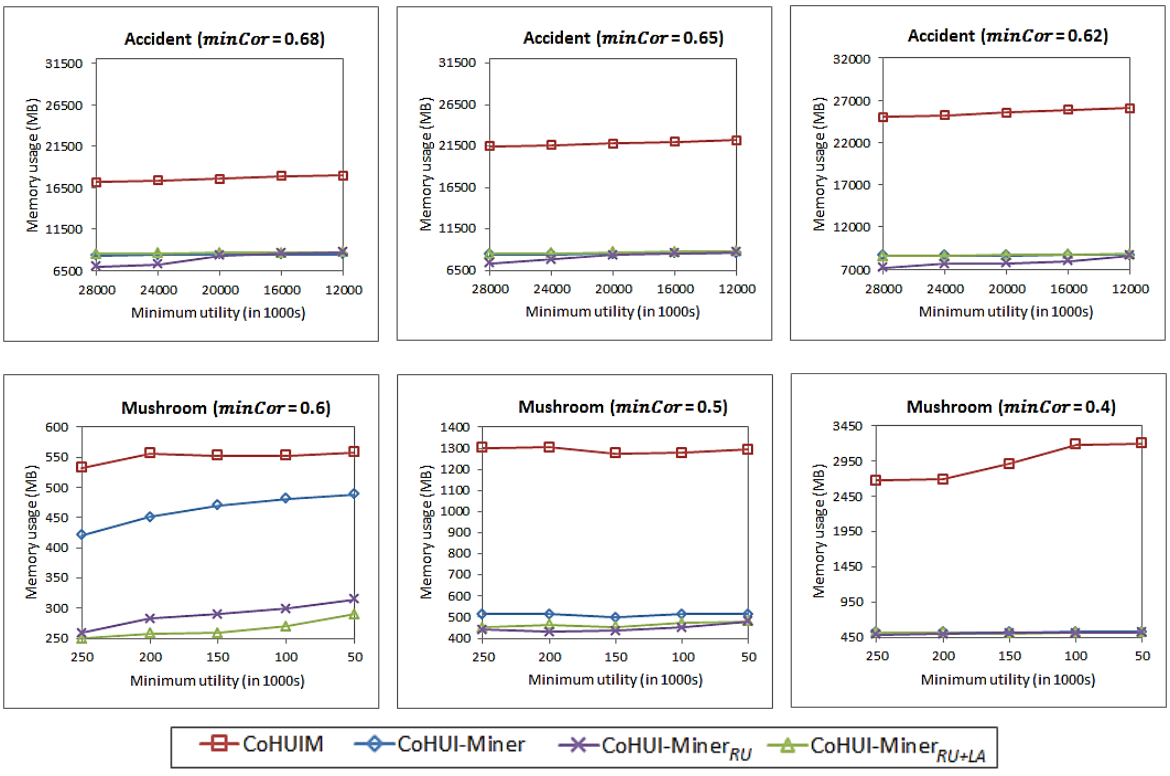

FIGURE 5. Memory consumptions of the CoHUI-Miner and CoHUIM on moderately dense datasets.

Similarly, the memory usage with the Mushroom dataset is also about five times lower. The strategies to reduce the search space and memory usage are still effective, but not as much as with dense databases because the number of candidates in these is significantly lower compared to Connect or Chess.

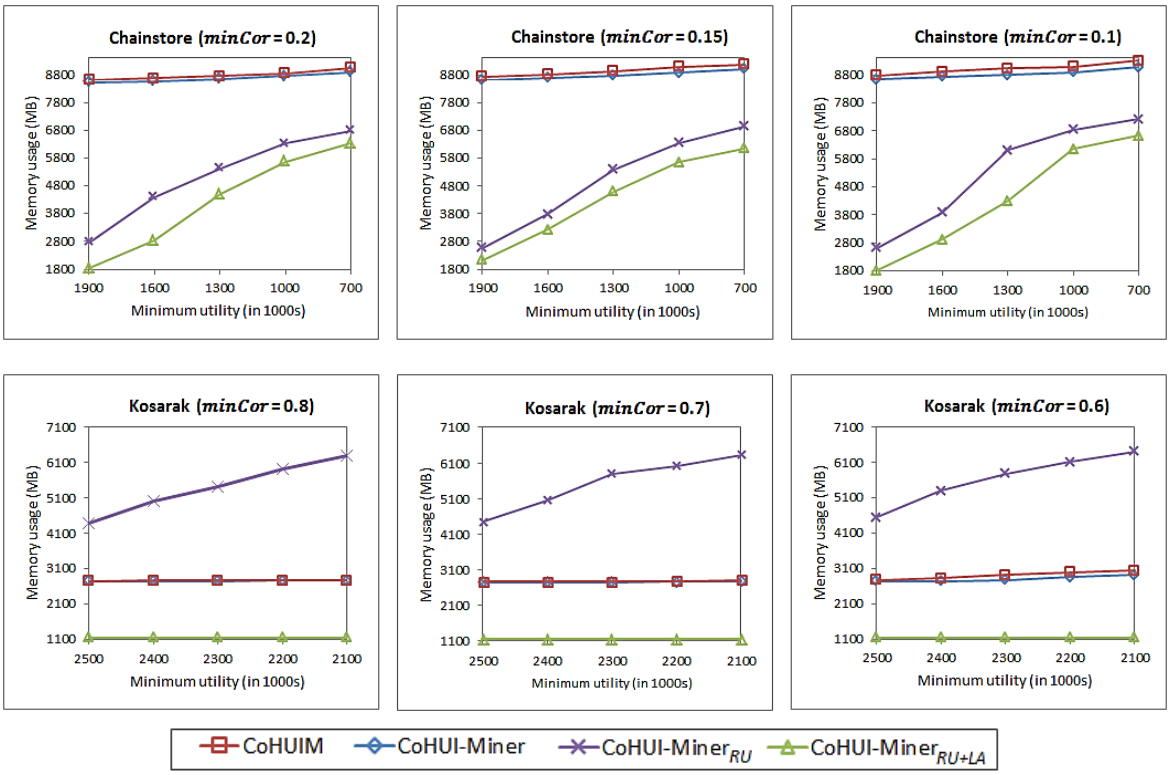

FIGURE 6. Memory consumptions of the CoHUI-Miner and CoHUIM on sparse or very sparse datasets.

In sparse or very sparse datasets (Fig. 6), such as the Chainstore dataset, the memory usage of the basic CoHUI-Miner algorithm is not much different from that of CoHUIM, because in a very sparse dataset the length of each candidate is small, so the CoHUIM algorithm uses only a little memory to store the candidates. However, the *U-Prune* and *LA-Prune* strategies show that the amount of memory usage decreases significantly because these pruning strategies remove many patterns which do not satisfy the CoHUI property. For instance, at $minCor = 0.2$ and $minUtil = 1,900,000$, the CoHUI-Miner$_{RU+LA}$ uses only 1,800MB, the CoHUI-Miner$_{RU}$ uses 2,800MB, while the CoHUIM uses about 8,500MB.

For the Kosarak dataset, similar to the Chainstore dataset, the memory usage of the basic CoHUI-Miner is not much different from that of CoHUIM. On the other hand, the CoHUI-Miner$_{RU+LA}$ algorithm shows the lowest memory usage at all the *minCor* and *minUtil* thresholds.

C. SUMMARY

Based on the experimental results, some observations can be summarized as follows. 1) The projection strategy on the database reduces the database size significantly when exploring the search space because it stores only the information of the needed transactions for calculation in the next level. 2) The new concept of *prefix utility* can be used to calculate the utility values of itemsets quickly in exploiting CoHUIs. As mentioned before, the utility of an itemset is calculated by the sum of the *prefix utility* values and the utility values of the items to be considered. Thus, when the *prefix utility* value is determined, the utility of an itemset can be calculated easily during the processing algorithm. 3) The CoHUI-Miner algorithm runs in one phase and does not generate the candidates. It shows good performance compared to the previous CoHUIM on runtime and memory usage. Through the experimental results, the efficiency of the algorithm has been proven. Especially, with the dense and moderately dense datasets, the performance of the CoHUI-Miner algorithm is much better than the CoHUIM algorithm. 4) Many pruning strategies such as TWU-Prune, KULC-prune, Variant of U-Prune and LA-Prune are applied to mine CoHUIs effectively on the projected database.

VI. CONCLUSIONS

This paper presented a new measure for efficiently mining CoHUIs, in which we calculated the utility values of projected transactions via projected databases. We proposed the prefix utility to calculate the utilities of extended itemsets during the search process. The storage of the prefix utility combined with many pruning strategies during the mining process significantly increases the performance of the proposed algorithm. The CoHUI-Miner is a single-phase algorithm and no candidates were generated during the discovery process for CoHUIs. The experimental results on several databases with different densities, from sparse to very dense, showed that our proposed algorithm was more efficient compared to the previous one.

In the next study, we aim to use a novel data structure to reduce the memory usage, and explore some pruning strategies to increase the performance of the algorithm.

Moreover, we will apply this approach to incremental and dynamic profit databases. On the other hand, we will combine with the optimization problem [46]–[49] to be able to improve the quality of industrial production activities.

REFERENCES

[1] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 1993, pp. 207–216.
[2] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proc. 20th Int. Conf. Very Large Data Bases (VLDB)*, 1994, pp. 487–499.
[3] T. Wu, Y. Chen, and J. Han, "Re-examination of interestingness measures in pattern mining: A unified framework," *Data Mining Knowl. Discovery*, vol. 21, no. 3, pp. 371–397, Nov. 2010.
[4] B. Vo, T. Le, F. Coenen, and T.-P. Hong, "Mining frequent itemsets using the N-list and subsume concepts," *Int. J. Mach. Learn. Cybern.*, vol. 7, no. 2, pp. 253–265, Apr. 2016.
[5] Y. Djenouri, D. Djenouri, J. C.-W. Lin, and A. Belhadi, "Frequent itemset mining in big data with effective single scan algorithms," *IEEE Access*, vol. 6, pp. 68013–68026, 2018.
[6] E. R. Omiecinski, "Alternative interest measures for mining associations in databases," *IEEE Trans. Knowl. Data Eng.*, vol. 15, no. 1, pp. 57–69, Jan. 2003.
[7] L. T. T. Nguyen, B. Vo, L. T. T. Nguyen, P. Fournier-Viger, and A. Selamat, "ETARM: An efficient top-k association rule mining algorithm," *Appl. Intell.*, vol. 48, no. 5, pp. 1148–1160, 2018.
[8] T. Mai, L. T. T. Nguyen, B. Vo, U. Yun, and T.-P. Hong, "Efficient algorithm for mining non-redundant high-utility association rules," *Sensors*, vol. 20, no. 4, p. 1078, 2020.
[9] B. Vo, L. T. T. Nguyen, N. Bui, T. D. D. Nguyen, V.-N. Huynh, and T.-P. Hong, "An efficient method for mining closed potential high-utility itemsets," *IEEE Access*, vol. 8, pp. 31813–31822, 2020.
[10] H. Nam, U. Yun, B. Vo, T. Truong, Z.-H. Deng, and E. Yoon, "Efficient approach for damped window-based high utility pattern mining with list structure," *IEEE Access*, vol. 8, pp. 50958–50968, 2020.
[11] U. Yun, H. Nam, J. Kim, H. Kim, Y. Baek, J. Lee, E. Yoon, T. Truong, B. Vo, and W. Pedrycz, "Efficient transaction deleting approach of pre-large based high utility pattern mining in dynamic databases," *Future Gener. Comput. Syst.*, vol. 103, pp. 58–78, Feb. 2020.
[12] 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. 482–486.
[13] Y. Liu, W. K. Liao, and A. Choudhary, "A two-phase algorithm for fast discovery of high utility itemsets," in *Proc. Pacific-Asia Conf. Knowl. Discovery Data Mining*, 2005, pp. 689–695.
[14] V. S. Tseng, C. W. Wu, B. E. Shie, and P. S. Yu, "UP-Growth: An efficient algorithm for high utility itemset mining," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 253–262.
[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] P. Fournier-Viger, C. W. Wu, S. Zida, and V. S. Tseng, "FHM: Faster high-utility itemset mining using estimated utility co-occurrence pruning," in *Proc. Int. Symp. Methodol. Intell. Syst.*, vol. 8502, 2014, pp. 83–92.
[17] 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.
[18] 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.
[19] U. Yun, H. Ryang, and K. H. Ryu, "High utility itemset mining with techniques for reducing overestimated utilities and pruning candidates," *Expert Syst. Appl.*, vol. 41, no. 8, pp. 3861–3878, Jun. 2014.
[20] S. Dawar, V. Goyal, and D. Bera, "A hybrid framework for mining high-utility itemsets in a sparse transaction database," *Appl. Intell.*, vol. 47, no. 3, pp. 809–827, 2017.
[21] M. Shang, X. Luo, Z. Liu, J. Chen, Y. Yuan, and M. Zhou, "Randomized latent factor model for high-dimensional and sparse matrices from industrial applications," *IEEE/CAA J. Autom. Sinica*, vol. 6, no. 1, pp. 131–141, Jan. 2019.
[22] X. Shi, Q. He, X. Luo, Y. Bai, and M. Shang, "Large-scale and scalable latent factor analysis via distributed alternative stochastic gradient descent for recommender systems," *IEEE Trans. Big Data*, early access, Feb. 11, 2020, doi: 10.1109/TBDATA.2020.2973141.
[23] X. Luo, D. Wang, M. Zhou, and H. Yuan, "Latent factor-based recommenders relying on extended stochastic gradient descent algorithms," *IEEE Trans. Syst., Man, Cybern. Syst.*, early access, Jan. 3, 2019, doi: 10.1109/TSMC.2018.2884191.
[24] L. T. T. Nguyen, P. Nguyen, T. D. D. Nguyen, B. Vo, P. Fournier-Viger, and V. S. Tseng, "Mining high-utility itemsets in dynamic profit databases," *Knowl.-Based Syst.*, vol. 175, pp. 130–144, Jul. 2019.
[25] J. C.-W. Lin, W. Gan, P. Fournier-Viger, T.-P. Hong, and H.-C. Chao, "FDHUP: Fast algorithm for mining discriminative high utility patterns," *Knowl. Inf. Syst.*, vol. 51, no. 3, pp. 873–909, Jun. 2017.
[26] W. Gan, J. C.-W. Lin, P. Fournier-Viger, H.-C. Chao, and H. Fujita, "Extracting non-redundant correlated purchase behaviors by utility measure," *Knowl.-Based Syst.*, vol. 143, pp. 30–41, Mar. 2018.
[27] A. Erwin, R. P. Gopalan, and N. R. Achuthan, "CTU-Mine: An efficient high utility itemset mining algorithm using the pattern growth approach," in *Proc. 7th IEEE Int. Conf. Comput. Inf. Technol. (CIT)*, Oct. 2007, pp. 71–76.
[28] B. Le, H. Nguyen, T. A. Cao, and B. Vo, "A novel algorithm for mining high utility itemsets," in *Proc. 1st Asian Conf. Intell. Inf. Database Syst.*, 2009, pp. 13–17.
[29] 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.
[30] 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.
[31] Q.-H. Duong, P. Fournier-Viger, H. Ramampiaro, K. Nørvåg, and T.-L. Dam, "Efficient high utility itemset mining using buffered utility-lists," *Int. J. Speech Technol.*, vol. 48, no. 7, pp. 1859–1877, Jul. 2018.
[32] J. Yin, Z. Zheng, and L. Cao, "USpan: An efficient algorithm for mining high utility sequential patterns," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 660–668.
[33] J.-Z. Wang, J.-L. Huang, and Y.-C. Chen, "On efficiently mining high utility sequential patterns," *Knowl. Inf. Syst.*, vol. 49, no. 2, pp. 597–627, Nov. 2016.
[34] J. C.-W. Lin, Y. Li, P. Fournier-Viger, Y. Djenouri, and L. S.-L. Wang, "Mining high-utility sequential patterns from big datasets," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2019, pp. 2674–2680.
[35] S. Kannimuthu and K. Premalatha, "Discovery of high utility itemsets using genetic algorithm with ranked mutation," *Appl. Artif. Intell.*, vol. 28, no. 4, pp. 337–359, Apr. 2014.
[36] J. C.-W. Lin, L. Yang, P. Fournier-Viger, T.-P. Hong, and M. Voznak, "A binary PSO approach to mine high-utility itemsets," *Soft Comput.*, vol. 21, no. 17, pp. 5103–5121, Sep. 2017.
[37] W. Song and C. Huang, "Mining high utility itemsets using bio-inspired algorithms: A diverse optimal value framework," *IEEE Access*, vol. 6, pp. 19568–19582, 2018.
[38] Q. Zhang, W. Fang, J. Sun, and Q. Wang, "Improved genetic algorithm for high-utility itemset mining," *IEEE Access*, vol. 7, pp. 176799–176813, 2019.
[39] L. Duan and W. N. Street, "Finding maximal fully-correlated itemsets in large databases," in *Proc. 9th IEEE Int. Conf. Data Mining*, Dec. 2009, pp. 770–775.
[40] C. F. Ahmed, S. K. Tanbeer, B.-S. Jeong, and H.-J. Choi, "A framework for mining interesting high utility patterns with a strong frequency affinity," *Inf. Sci.*, vol. 181, pp. 4878–4894, Nov. 2011.
[41] P. Fournier-Viger, Y. Zhang, J. C.-W. Lin, D.-T. Dinh, and H. B. Le, "Mining correlated high-utility itemsets using various measures," *Log. J. IGPL*, vol. 28, no. 1, pp. 19–32, Jan. 2020.
[42] P. Fournier-Viger, Y. Zhang, J. Chun-Wei Lin, H. Fujita, and Y. S. Koh, "Mining local and peak high utility itemsets," *Inf. Sci.*, vol. 481, pp. 344–367, May 2019.
[43] Y. Djenouri, J. C.-W. Lin, K. Norvag, and H. Ramampiaro, "Highly efficient pattern mining based on transaction decomposition," in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, Apr. 2019, pp. 1646–1649.
[44] W. Gan, J. C.-W. Lin, H.-C. Chao, H. Fujita, and P. S. Yu, "Correlated utility-based pattern mining," *Inf. Sci.*, vol. 504, pp. 470–486, Dec. 2019.
[45] P. Fournier-Viger, A. Gomariz, T. Gueniche, A. Soltani, C.-W. Wu, and V. S. Tseng, "SPMF: A java open-source pattern mining library," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3389–3393, 2014.
[46] X. Luo, M. Zhou, S. Li, L. Hu, and M. Shang, "Non-negativity constrained missing data estimation for high-dimensional and sparse matrices from industrial applications," *IEEE Trans. Cybern.*, vol. 50, no. 5, pp. 1844–1855, May 2020.

[47] X. Luo, M. Zhou, Z. Wang, Y. Xia, and Q. Zhu, "An effective scheme for QoS estimation via alternating direction method-based matrix factorization," *IEEE Trans. Services Comput.*, vol. 12, no. 4, pp. 503–518, Jul. 2019.
[48] A. H. Khan, X. Cao, S. Li, V. N. Katsikis, and L. Liao, "BAS-ADAM: An ADAM based approach to improve the performance of beetle antennae search optimizer," *IEEE/CAA J. Autom. Sinica*, vol. 7, no. 2, pp. 461–471, Mar. 2020.
[49] H. Kazemi and A. Yazdizadeh, "Optimal state estimation and fault diagnosis for a class of nonlinear systems," *IEEE/CAA J. Autom. Sinica*, vol. 7, no. 2, pp. 517–526, Mar. 2020.

BAY VO received the Ph.D. degree in computer science from the University of Science, Vietnam National University, Ho Chi Minh City, Vietnam, in 2011. He is currently an Associate Professor at the Ho Chi Minh City University of Technology, Vietnam. His research interests include association rules, classification, mining in incremental database, distributed databases and privacy preserving in data mining, and soft computing.

LE V. NGUYEN received the M.S. degree in data transmission and computer network from the Posts and Telecommunications Institute of Technology, Vietnam, in 2011. He is currently a Lecturer at the Ho Chi Minh City University of Food Industry, Vietnam. His research interests include association rules, classification, text processing, and mining high utility itemset in data mining.

VINH V. VU received the M.S. degree in information technology from the Ho Chi Minh City University of Technology, Ho Chi Minh city, Vietnam, in 2019. His research interests include association rules, classification, text processing, mining high utility itemset, and privacy preserving in data mining.

MI T. H. LAM received the M.S. degree in mathematical foundation for computers and computing systems from the University of Science, Vietnam National University, Ho Chi Minh City, Vietnam, in 2013. She is currently a Lecturer at the Ho Chi Minh City University of Food Industry, Vietnam. Her research interests include association rules, classification, text processing, and mining high utility itemset in data mining.

THUY T. M. DUONG received the M.S. degree in computer science from the University of Science, Vietnam National University, Ho Chi Minh City, Vietnam, in 2013. She is currently a Lecturer at the Ho Chi Minh City University of Food Industry, Vietnam. Her research interests include association rules, classification, text processing, and mining high utility itemset in data mining.

LY T. MANH received the M.S. degree in information system from Military Technical Academy, Hanoi, Vietnam, in 2010. She is currently a Lecturer at the Ho Chi Minh City University of Food Industry, Vietnam. Her research interests include association rules, classification, text processing, and mining high utility itemset in data mining.

THUY T. T. NGUYEN received the B.S. degree in information technology from the University of Science, Vietnam National University, Ho Chi Minh City, Vietnam, in 2003, and the M.S. degree in business administration from the University of Economics Ho Chi Minh City, in 2013. She is currently a Lecturer at the Ho Chi Minh City University of Food Industry, Vietnam. Her research interests include association rules, classification, text processing, and mining high utility itemset in data mining.

LOAN T. T. NGUYEN received the B.Sc. and M.Sc. degrees in computer science from Vietnam National University, Ho Chi Minh City, Vietnam, in 2002 and 2008, respectively, and the Ph.D. degree in computer science from the Wroclaw University of Science and Technology, Poland, in 2015. From October 2016 to September 2017, she was an ERCIM Postdoctoral Researcher with the University of Warsaw, Poland. She was a Visiting Researcher at NTNU, Norway, in March 2017. Her research interests include association rules, classification, and mining in incremental databases.

TZUNG-PEI HONG received the B.S. degree in chemical engineering from National Taiwan University, in 1985, and the Ph.D. degree in computer science and information engineering from National Chiao-Tung University, in 1992.

He served for the Department of Computer Science, Chung-Hua Polytechnic Institute, from 1992 to 1994, and for the Department of Information Management, I-Shou University, from 1994 to 2001. He was in charge of the whole computerization and library planning with the National University of Kaohsiung, in preparation, from 1997 to 2000, and served as the first Director of the Library and Computer Center, National University of Kaohsiung, from 2000 to 2001, as the Dean of Academic Affairs, from 2003 to 2006, as the Administrative Vice President, from 2007 to 2008, and as the Academic Vice President, in 2010. He is currently a Distinguished and Chair Professor at the Department of Computer Science and Information Engineering and at the Department of Electrical Engineering, National University of Kaohsiung, Taiwan, where he is also the Director of the AI Research Center. He is also a Joint Professor with the Department of Computer Science and Engineering, National Sun Yat-sen University, Taiwan. He has published more than 600 research articles in international/national journals and conferences and has planned more than 50 information systems. He is also a board member of more than 40 journals and the program committee member of more than 500 conferences. His current research interests include knowledge engineering, data mining, soft computing, management information systems, and WWW applications. He received the first National Flexible Wage Award from the Ministry of Education, Taiwan.