IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 2, 2020, accepted August 10, 2020, date of publication August 20, 2020, date of current version September 1, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3018155

Using Tree Structure to Mine High Temporal Fuzzy Utility Itemsets

TZUNG-PEI HONG[1,2], (Senior Member, IEEE), CHENG-YU LIN[2], WEI-MING HUANG[2], KATHERINE SHU-MIN LI[2], (Senior Member, IEEE), LEON SHYUE-LIANG WANG[1], (Member, IEEE), AND JERRY CHUN-WEI LIN[3], (Senior Member, IEEE)

[1]Department of Computer Science and Information Engineering, National University of Kaohsiung, Kaohsiung 811, Taiwan
[2]Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung 804, Taiwan
[3]Department of Computer Science, Electrical Engineering and Mathematical Sciences, Western Norway University of Applied Science, 5020 Bergen, Norway

Corresponding author: Wei-Ming Huang (granthill168@gmail.com)

ABSTRACT Data mining is a critical technology for extracting valuable knowledge from databases. It has been used in many fields, like retail, finance, biology, etc. In computational intelligence, fuzzy logic has been applied in many intelligent systems widely because it is simple and similar to human inference. Fuzzy utility mining combines utility mining and fuzzy logic for getting linguistic utility knowledge. In this paper, we study a more challenging, complicated, but practical topic called temporal fuzzy utility data mining, which considers the temporal periods in transactions, purchased amounts, item profits, and understandable linguistic terms as important factors. Although an Apriori-based algorithm was proposed previously, its execution was not efficient. We thus use a modified tree structure based on the classical frequent-pattern tree to improve its performance. A tree-based mining algorithm is also proposed to mine temporal fuzzy utility itemsets from quantitative transactional databases. The tree structure is built to keep all temporal fuzzy utility 1-itemsets in a database. All the high temporal fuzzy utility itemsets in a database can be obtained by traversing the tree-based structure. The proposed algorithm gets the final results through two phases. In the first phase, a procedure like FP-Growth is used to find the candidate itemsets. In the second phase, the temporal fuzzy utility database is scanned to decide whether the candidate itemsets are desired. Experimental results show that the proposed algorithm is superior to the existing algorithm for temporal fuzzy utility mining in terms of processing time and used memory.

INDEX TERMS Fuzzy set, quantitative database, temporal fuzzy utility mining, tree structure, utility mining.

I. INTRODUCTION

Most data-mining methods [1]–[3] use item frequencies in transaction databases to decide the degrees of importance. They are expected to find useful data and make practical decisions in different domains, such as finance, retail industry, and biology. For example, Enke *et al.* combined data mining with neural networks to forecast the stock market [8]. Chen *et al.* applied data mining methods in the online retail industry [7]. Hirschman *et al.* reviewed data mining methods used in the literature for biology, analyze them, and summarize their accomplishments and challenges [11]. However, frequent itemsets with low prices typically make little contribution to the total benefit of a company; on the contrary, non-frequent itemsets with highly beneficial rates may be desired and worth of promotion. For example, a car gets much higher profit than a motorcycle, even though the former is less sold than the latter. In business, combinations of items that can earn well are more critical than those with high frequencies. Consequently, Yao *et al.* [36] proposed utility mining by simultaneously considering purchased quantities and actual profits to find itemsets with high utility values. An itemset with its utility value larger than a specified minimum utility threshold is regarded as relevant and called a high utility itemset. A disadvantage of Yao *et al.*'s method is that mining steps do not keep the downward-closure property. Hence, it consumes more processing time than mining association rules.

To mine desired utility itemsets efficiently, a two-phase method proposed by Liu *et al.* [24], [25] could have the downward-closure property in the mining process. Upper-bound values for itemsets were designed to keep the

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

monotonic property in Phase 1. Unpromising itemsets were thus pruned as early as possible. In Phase 2, the mined itemsets found in Phase 1 as the candidates were used to decide the actual high utility itemsets by rescanning the database.

Besides, the temporal factor is very critical to the analysis of business behavior. In real situations, the transactions in supermarkets are recorded with check-out time. Furthermore, different products may begin their sales at different times. If all the products are mined using the same duration, it will cause some biases for their real utility associations. The temporal relationship among purchased products is complex and not easily found out in utility mining. Some scholars thus proposed mining approaches to reveal ordered correlation among items from temporal transactions. For example, Lee *et al.* used the common exhibition periods for items in a publication database to obtain corresponding temporal rules [21]. Chang *et al.* considered different exhibition times for all products in a dataset [5] to find temporal association rules. Weng discussed the marked times of products as their on-shelf times [34]. He designed a measurement to find more relevant patterns than before by avoiding generating useless itemsets.

The fuzzy set theory provides good linguistic representation for bridging the gap between computers and human beings. People are perceived as linguistic more than quantitative representation. The fuzzy logic was proposed by Zadeh [38] and had a literal concept analogous to human perception. This concept had also been used in intelligent information systems with many good applications. In data mining, when a transactional database with sold quantitative information is processed, the idea of the fuzzy sets can be used to transform the item quantity into linguistic representation. It uses semantic terms to represent concepts, which better matches the way of human thought. Some fuzzy data mining algorithms were proposed for finding linguistic association rules [6], [13], [39]. During the last years, fuzzy utility mining algorithms [15], [20] have been proposed as well. They transformed the quantity values of purchased items in each transaction into fuzzy terms by using pre-defined membership functions and then designed mining steps to find linguistic knowledge from the fuzzy terms. For example, Lan *et al.* [20] used an upper-bound model based on the fuzzy concept to hold the monotonic property for estimating high fuzzy utility patterns level by level [25].

Huang *et al.* applied the fuzzy sets to handle the temporal utility data mining [15]. They considered both the transaction time and the purchased quantities of items in a temporal quantitative transactional database and designed a temporal fuzzy utility mining algorithm to mine pertinent linguistic itemsets with the temporal property. It provided a two-phase model to hold the monotonic property in solving the problem. Nevertheless, its execution time is slow due to the consideration of the time factor and level-wise processing.

In this paper, a tree-structure algorithm is proposed to overcome the execution efficiency of the previous temporal fuzzy utility mining algorithm [15]. The proposed approach uses the concept of the FP-tree [10] to reduce the multiple scans to a temporal quantitative database, thus speed up the execution. A tree-structure is mainly used to maintain information for candidates in the first phase. The designed method utilizes it to confirm whether the candidate was desired in a second database scan. The experimental results show that the proposed approach can indeed improve performance.

The rest of the paper is organized as follows. The relevant background information is given in Section II. The problem to be solved and some related definitions are described in Section III. The proposed algorithm is stated and explained in Section IV. An example is used to illustrate the proposed algorithm in Section V. Experimental results are shown and discussed in Section VI. We make a conclusion in Section VII.

II. REVIEW OF RELATED WORKS

In this section, some related works on frequent-pattern growth mining, utility mining, and fuzzy utility mining are briefly reviewed.

A. FREQUENT-PATTERN GROWTH MINING

The Apriori algorithm is the most well-known data mining method, which was proposed by Agrawal *et al.* to find association rules [1]–[3]. At each iteration, the Apriori method needs to capture the set of frequent k-itemsets by scanning a database once. Its major drawback is that a substantial amount of candidates are generated during the mining process. If a dataset has n items, then $2^n - 1$ candidates may be generated and examined in the worst case. Therefore, this method needs multiple database scans for deciding frequent itemsets.

FP-Growth [10], called Frequent-Pattern Growth, is a popular algorithm for mining frequent itemsets without generating candidates. To find all frequent itemsets efficiently, this method only scans the dataset twice. It thus shortens computational time to find frequent itemsets. In the beginning, FP-Growth conducts the first database scan is to mine all frequent 1-itemsets. It then scans the database again to remove irrelevant items and compress the whole dataset into a tree structure named FP-Tree to store the frequent items with corresponding information needed for mining. Every node in an FP-Tree has an item and its frequency (support). After the tree is built up, the complete set of frequent itemsets can be found by recursively finding conditional trees and traversing them. The FP-Growth [10] algorithm is stated as follows.

Step 1: Scan the database once to get the frequency value of each item (1-itemset) in the database and keep the items with their values larger than the pre-defined minimum support as the frequent 1-itemsets.

Step 2: Sort the frequent 1-itemsets in descending order of their frequencies and then build the header table according to the order.

Step 3: Scan the database again to remove in each transaction non-frequent items, sort the remaining items according to the order above, and insert them into the FP-Tree.

Step 4: Find the conditional pattern base of each 1-itemset in the header table from the last item to the first one.
Step 5: Use the conditional pattern base of each 1-itemset in the header table to build the conditional sub-tree and then to find frequent itemsets.
Step 6: Repeat Step 5 until no frequent itemsets are found.

B. UTILITY MINING

Frequent itemset mining (FIM) is useful, but it only considers binary databases. Therefore, it has some restrictions. From the perspective of business marketing, the first restriction is that purchasing some quantities of an item in a market is considered equally important to buy a single one. It is not in line with the actual situation in real life. The second restriction is that all items in a market are viewed as equally important. But in a retail store, the sale of a mobile phone is more important than that of a headset since the former has a higher profit than the latter. The third restriction is that frequent itemsets derived by FIM may be less attractive to users because some high-profit item combinations may be hard to find, especially if the items are with low frequencies. The products with high profits but low frequencies in a market may be important as well.

To overcome these restrictions, utility mining has recently evolved into an important research topic. It was first proposed by Yao *et al.* [36]. They designed a utility measure to represent the importance of an item in a dataset. Different from FIM, utility mining simultaneously considers the quantity values of items in a dataset and their profits as critical factors. If the utility values of items satisfy the minimum utility threshold set by a user, they can be viewed as high utility itemsets. However, utility mining is more complicated than FIM because the latter holds the monotonic property, but the former does not. That is, the utility value of an itemset may be larger, smaller, or equal to those of its subsets. Liu *et al.* thus proposed the two-phase method [24], [25] to estimate the set of high utility itemsets efficiently. The designed model, called transaction-weighted utility, provides the monotonic property for the utility measure to reduce the search space in mining. They applied the sum of the utility values of all items in each transaction to recognize the upper-bound value of that transaction. Consequently, the upper-bound value of an itemset in a dataset is the sum of the upper-bound values of transactions, which include the itemset. Other related methods [23], [30] used by this model were proposed to find high utility itemsets.

Several variants of mining high utility itemsets were also proposed. Vo *et al.* presented the MCH-Miner to find high utility itemsets with varying unit profits of items in a dataset by parallel processing [32]. It was assumed that the profit values of all items might change along with item promotion, supply chain cost, or other factors. Besides, a divide-and-conquer strategy was adopted to overcome the performance in the mining process. Nam *et al.* proposed the DHUPL algorithm to find high utility itemsets in an incremental circumstance. A list structure was designed to reveal the different importance of the items. The method used a damped window model to prune itemsets and considered newly inserted data more important than the previous ones [28].

C. FUZZY UTILITY MINING

Fuzzy sets, which are similar to human thought, can easily represent quantitative information when mining quantitative datasets. The item quantities in a dataset are turned into linguistic representations based on fuzzy membership functions defined in advance. The transformed linguistic representation can imply uncertain features which are interposed within a specific range of numbers. For instance, ''three apples and five bananas are sold'' is a transaction in a database. The transformed linguistic terms can be treated as a ''low'' amount of apples to represent people's feelings.

Based on the above concept, several algorithms about fuzzy data mining [12], [22], [26] have been developed. These mining approaches start by turning the item quantity (the number of items sold) into fuzzy terms that are assigned in advance in line with the different membership functions of individual items. For example, Hong *et al.* mined fuzzy association rules to reveal amusing patterns with quantitative information. They extended the Apriori algorithm and used the fuzzy concept to mine interesting linguistic patterns [12]. For example, the mining pattern {*A.Low*, *B.High*} may be mined out and can be interpreted as ''If someone buys a low amount of product *A*, then he/she may also buy a high amount of product *B*.'' Although the results derived from the fuzzy data mining algorithm are useful, its execution cost is high. To solve this problem, a fuzzy frequent pattern tree based on the structure of the FP-Tree was designed by Lin *et al.* to handle fuzzy data efficiently [22]. The mining method based on the tree can find fuzzy frequent itemsets with two database scans.

In contrast to frequent itemset mining, utility mining [24], [25] was designed to find high-utility itemsets which can get more profits than ordinary ones in a dataset. However, high utility itemsets contain just items without quantitative information and are difficult for users to understand their quantitative occurrence relationship. As mentioned before, fuzzy itemsets [18], [35] could represent the linguistic meaning of item quantity relationships, such as the example {*A.Low*, *B.High*}. Thus, fuzzy utility mining [20] has been developed into another new research issue for being more suitable for practical applications. Considering item quantities in a dataset, profit values of the items, and linguistic meaning of the quantities, fuzzy utility mining turns the items with quantitative information into fuzzy sets to derive high fuzzy utility itemsets. To reduce the search space in mining, Lan *et al.* developed an upper-bound method to retain the downward-closure property in fuzzy utility mining. Based on the extension of the concept, Huang *et al.* then considered different transaction periods in a dataset to discover linguistic temporal utility patterns [15].

III. PROBLEM DEFINITIONS

In this section, some critical terms about mining high temporal fuzzy utility itemsets (*HTFUIs*) based on the temporal

TABLE 1. A quantitative database.

Period	*TID*	*Items and Quantities*
P_1	$Trans_1$	{6*A*, 2*C*}
P_1	$Trans_2$	{4*A*, 4*B*}
P_2	$Trans_3$	{1*A*, 1*C*}
P_2	$Trans_4$	{3*A*, 3*B*, 4*C*, 4*D*, 2*E*}
P_3	$Trans_5$	{3*A*, 6*B*, 2*E*}
P_3	$Trans_6$	{3*A*, 4*D*, 2*E*}

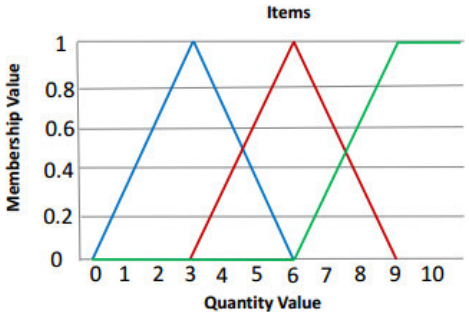

FIGURE 1. Item membership functions.

TABLE 2. Item utilities.

Item	*Profit*
A	2
B	6
C	4
D	2
E	4

fuzzy utility mining problem [15] are described below. Let $I = \{i_1, i_2, i_3, \ldots, i_m\}$ be a finite set of m distinct items in the temporal quantitative transaction database $TQD = \{Trans_1, Trans_2, \ldots, Trans_z\}$, where $Trans_z \in TQD$ and $Trans_y$ is the y-th transaction in TQD. Each transaction $Trans_y$ includes items sold i_m and quantity sold v_{ym}. According to the membership function of the quantitative value v_{ym} of the item i_m, assume that R_{m1} to R_{mh} are the elements in fuzzy set f_{ym}, and u_{ym1} to u_{ymh} are their membership values in f_{ym}. Besides, each item i_m has external utility value, denoted as $s(i_m)$, for the profit of i_m. Let a fuzzy region of an item is denoted as a fuzzy item, and a fuzzy itemset includes at least two fuzzy items with no fuzzy items are originated from the same item. A pre-defined minimum temporal fuzzy utility threshold is set as λ.

Definition 1: A time period set $T = \{p_1, p_2, p_3, \ldots, p_m\}$ in the temporal quantitative transaction database TQD, m is the number of the time periods, and p_m denotes the m-th time period.

For example, assume that three time periods are defined in Table 1 for the running example.

Definition 2: For the quantitative value v_{ym} of an item i_m, the fuzzy utility fu_{ymh} of its h-th fuzzy item R_{mh} in the transaction $Trans_y$ is denoted as

$$fu_{ymh} = \mu_{ymh} * v_{ym} * s(i_m).$$

For example, assume that the membership functions and profits of all items in TQD are defined in Figure 1 and Table 2. The quantity of A in $Trans_2$ in Table 1 are transformed as (0.67/*A.Low*, 0.33/*A.Middle*, 0/*A.High*). Therefore, the fuzzy utility of *A.Low* in $Trans_2$ is calculated as 0.67*4*2 = 5.36.

Definition 3: The transactional fuzzy utility in a transaction $Trans_y$ is denoted as tfu_y and defined as the sum of the fuzzy utility values for all fuzzy items in the transaction $Trans_y$, that is:

$$tfu_y = \sum_{i_m \subseteq Trans_y} fu_{ym}.$$

Definition 4: The fuzzy utility value of a fuzzy itemset X in the transaction $Trans_y$ is denoted as fu_{yX} and defined as the sum of the fuzzy utility values of all items in X and also in $Trans_y$, which is denoted as:

$$fu_{yX} = \mu_{yX} * \sum_{R_{mh} \subseteq X} v_{ym} * s(i_m),$$

where the μ_{yX} can use the smallest membership grade in R_{mh} in X.

Definition 5: The start transaction period of an item i_m, denoted as STP_{im}, is the first occurring time of the transaction period in the TQD.

Definition 6: The last transaction period of an itemset X, denoted as LTP_X, is the last occurring time of transaction period originated from all items of an itemset that are purchased simultaneously last.

A designed upper-bound model [15], which is used to keep the monotonic property, was proposed to find all candidate itemsets to avoid information loss. A set of terms is defined as follows.

Definition 7: For an item i_m, its maximal fuzzy utility mfu_{ym} in $Trans_y$ is represented as

$$mfu_{ym} = max\left\{fu_{yml}, fu_{ym2}, \ldots, fu_{ymh}\right\}.$$

Definition 8: The maximal transactional fuzzy utility of $Trans_y$ is represented as $mtfu_y$. That is:

$$mtfu_y = \sum_{i_m \subseteq Trans_y} mfu_{ym}.$$

Definition 9: The start transaction period STP_{all} of all items is represented as

$$STP_{all} = min_{TP}\{STP_1, STP_2, \ldots, STP_m\}.$$

Definition 10: Let LTP_{all} be the time periods from STP_{all} to the last time period of TQD. The temporal fuzzy utility upper bound ratio of fuzzy itemset X is denoted as

$$tfuubr_X = \sum_{X \in Trans_y \cap Trans_y \in LTP_X} mtfu_{yX} \Big/ \sum_{Trans_y \in LTP_{all}} tfu_y.$$

Definition 11: If the $tfuubr_X$ value is no less than minimum temporal fuzzy utility threshold λ, fuzzy itemset X is a high temporal fuzzy utility upper bound itemsets (*HTFUUBIs*).

According to the above definitions, the possible candidate itemsets (*HTFUUBIs*) can be found by using the upper-bound model [15] and then determined whether these candidate itemsets are desired. The following terms with satisfying high temporal fuzzy utility itemsets (*HTFUIs*) are defined as below.

Definition 12: The temporal fuzzy utility ratio of the fuzzy itemset X is defined as $tfur_X$. That is:

$$tfur_X = \sum_{X \in Trans_y \cap Trans_y \in LTP_X} fu_{yX} \Big/ \sum_{Trans_y \in LTP_X} tfu_y.$$

Definition 13: If $tfur_X$ is no less than minimum temporal fuzzy utility threshold λ, fuzzy itemset X is an *HTFUI*.

Temporal fuzzy utility mining includes quantity, profit, transformed linguistic terms, and temporal behavior to mine high fuzzy utility itemsets with the temporal property. These itemsets derived from [15] include more useful and meaningful knowledge than those derived from fuzzy utility mining [20] despite the former has more the number of itemsets than those of the latter. Because of this, late-on-shelf or time-limited items may be found by using [15].

However, the monotonic property, which is the leading spirit for association rule, is not held in temporal fuzzy utility mining to prune unnecessary itemsets efficiently. Lan *et al.* designed a practical model with fuzzy utility upper-bounds to prune unpromising candidates early and also proposed a fuzzy-utility algorithm and adopted the minimum operator for the intersection. Afterward, Huang *et al.* proposed an extended upper-bound model and function to find more relevant patterns with considering temporal property as an essential factor. Those algorithms [15], [20] for mining high fuzzy utility itemsets were based on the Apriori-based approach and found knowledge patterns by scanning multiple databases. However, the Apriori-based approach generated explosive candidate itemsets and resulted in the execution time highly.

IV. THE PROPOSED ALGORITHM

A Fast High Temporal Fuzzy Utility Pattern tree (*FHTFUP*) algorithm, in this paper, is proposed to find *HTFUIs* based on the two-phase upper bound model [15] and FP-tree [10]. Using this model is mainly to find all possible temporal fuzzy utility upper bound itemsets based on downward closure property. The concept of FP-tree is thus used to decrease explosive candidates compared with the generate-and-test method. Based on the above definitions, there are mainly two phases in the proposed algorithm: (1) Finding the possible candidate itemsets by using similar FP-Growth and (2) Scanning the temporal fuzzy utility database to decide whether the candidate itemsets are *HTFUIs*.

The Proposed FHTFUP Algorithm for Mining Temporal Fuzzy Utility Itemsets:

INPUT:
(1) *TQD*, a temporal quantitative database with n quantitative transactions,
(2) m items in *TQD*,
(3) membership functions for m items,
(4) p time periods, and
(5) a pre-defined minimum temporal fuzzy utility threshold λ.

OUTPUT: All *HTFUIs* itemsets satisfying λ.

Phase I (Find Candidate Itemsets):

STEP 1. Transform the occurring time of each transaction in *TQD* into a unique time period.

STEP 2. Calculate the STP_{im} value of each item i_m in *TQD*.

STEP 3. For each item i_m in $Trans_y$, turn its quantitative value v_{ym} into a fuzzy set f_{ym} denoted as $\left(\frac{\mu_{ym1}}{R_{m1}} + \frac{\mu_{ym2}}{R_{m2}} + \ldots + \frac{\mu_{ymh}}{R_{mh}}\right)$ based on the membership grade for the quantity of each item i_m.

STEP 4. For each transaction $Trans_{jy}$ which exists in each period p_j of *TQD*, carry out the following substeps to process:
(a) Calculate the fu_{jymh} of the h-th fuzzy item of item i_m in $Trans_{jy}$.
(b) Find the mfu_{jym} of i_m in $Trans_{jy}$.
(c) Determine the tfu_{jy} and the $mtfu_{jy}$ of each $Trans_{jy}$.

STEP 5. Build the $HTFUUBI_1$ table as empty, where each tuple has a fuzzy 1-itemset, the total *mtfu* value, and the occurrence frequency value.

STEP 6. Calculate the *tfuubr* value of each fuzzy 1-itemset. If its *tfuubr* is larger than or equal to λ, it is $HTFUUBI_1$.

STEP 7. Insert the fuzzy 1-itemset into the $HTFUUBI_1$ table, including their total *mtfu* value and the occurrence frequency.

STEP 8. For *TQD*, delete the fuzzy 1-itemsets not existed in the $HTFUUBI_1$ table.

STEP 9. According to the descending order of their frequency in the $HTFUUBI_1$ table, sort the fuzzy 1-itemsets in the $HTFUUBI_1$ table as the header table of the *FHTFUP* tree.

STEP 10. For each transaction $Trans_y$ in *TQD*, insert each transaction into the *FHTFUP* tree. If an item in a transaction does not exist at the corresponding branch of the *FHTFUP* tree, insert the item to the end of the branch, and add the *mtfu* value of $Trans_y$. If yes, just add the *mtfu* value of $Trans_y$ to the *mtfu* value of the corresponding node.

STEP 11. After STEP 10, the final *FHTFUP* tree is constructed. Along with the *FHTFUP* tree, the candidate high temporal fuzzy utility itemsets *HTFUIs* can then be found in a way similar to FP-Growth mining, but much more complex. After finding the candidate *HTFUIs*, and Phase II needs to be executed.

Phase II (Find HTFUIs):

STEP 12. Obtain the fuzzy utility fu_X of each candidate *HTFUI* by scanning *TQD*.

STEP 13. Carry out the following substeps for each candidate *HTFUI*:
(a) Get LTP_X of itemset X and figure the sum of *tfu* in LTP_X.
(b) If the fu_X value divided by the sum of *tfu* is larger than the threshold λ, X is *HTFUI*; otherwise, remove it.

STEP 14. Output the *HTFUIs* to users.

TABLE 3. Start periods of items in the example.

Period	*STP*
A	P_1
B	P_1
C	P_1
D	P_2
E	P_2

TABLE 4. Converted linguistic representations of transactions in Table 1.

TID	*Linguistic representation*
$Trans_1$	$(\frac{0}{A.Low}, \frac{1}{A.Middle}, \frac{0}{A.High})(\frac{0.67}{C.Low}, \frac{0}{C.Middle}, \frac{0}{C.High})$
$Trans_2$	$(\frac{0.67}{A.Low}, \frac{0.33}{A.Middle}, \frac{0}{A.High})(\frac{0.67}{B.Low}, \frac{0.33}{B.Middle}, \frac{0}{B.High})$
$Trans_3$	$(\frac{0.33}{C.Low}, \frac{0}{C.Middle}, \frac{0}{C.High})(\frac{0.33}{A.Low}, \frac{0}{A.Middle}, \frac{0}{A.High})$
$Trans_4$	$(\frac{1}{A.Low}, \frac{0}{A.Middle}, \frac{0}{A.High})(\frac{1}{B.Low}, \frac{0}{B.Middle}, \frac{0}{B.High})$ $(\frac{0.67}{C.Low}, \frac{0.33}{C.Middle}, \frac{0}{C.High})(\frac{0.67}{D.Low}, \frac{0.33}{D.Middle}, \frac{0}{D.High})$ $(\frac{0.67}{E.Low}, \frac{0}{E.Middle}, \frac{0}{E.High})$
$Trans_5$	$(\frac{1}{A.Low}, \frac{0}{A.Middle}, \frac{0}{A.High})(\frac{0}{B.Low}, \frac{1}{B.Middle}, \frac{0}{B.High})$ $(\frac{0.67}{E.Low}, \frac{0}{E.Middle}, \frac{0}{E.High})$
$Trans_6$	$(\frac{1}{A.Low}, \frac{0}{A.Middle}, \frac{0}{A.High})(\frac{0.67}{D.Low}, \frac{0.33}{D.Middle}, \frac{0}{D.High})$ $(\frac{0.67}{E.Low}, \frac{0}{E.Middle}, \frac{0}{E.High})$

V. AN EXAMPLE FOR THE PROPOSED ALGORITHM

The following is a running example of the proposed algorithm. Table 1 shows a database (*TQD*) containing temporal and quantitative properties. In utility mining, the item profits must be known in advance; they are given in Table 2. Besides, the fuzzy concept is also concerned with our algorithm. Thus we assume the membership functions of the above five items shown in Figure 1. A user-defined mining threshold is considered as 40% for this example.

Steps 1 to 3: The start time periods of items are found in Table 1: the results are shown in Table 3. For example, $Trans_1$ contains *A* and *C* at quantities of 6 and 2, respectively. Given Figure 1, the quantity values are transformed into fuzzy sets (0, 1, 0) and (0.67, 0, 0). The results are shown in Table 4; other transactions are processed likewise.

Step 4: The fuzzy utility values of the items in each transaction in Table 4 are calculated. For instance: item *C* in $Trans_1$. The fuzzy utility values of three fuzzy items of the item *C* are 0.67*2*4 = 5.36, 0*2*4 = 0 and 0*2*4 = 0, respectively. The same process is applied to the other items. The results are shown in Table 5.

According to definitions 3, 7 and 8, the *mfu* values of items *A* and *C* in $Trans_1$ are 12 and 5.36, respectively. The *mtfu* of $Trans_1$ is 12 + 5.36 = 17.36. The *tfu* of $Trans_1$ is 12 + 5.36 = 17.36. The other transactions are processed in the same way. The *mtfu* and *tfu* are represented in the last columns of Table 5.

Step 5: Initialize the $HTFUUBI_1$ table as empty.

Steps 6 to 7: Find each fuzzy 1-itemset in Table 5 and calculate the corresponding *tfuub*. If its *tfuubr* is not smaller than the given threshold, put it into the $HTFUUBI_1$ table. For instance, the fuzzy 1-itemset {*A.Low*} occurs in $Trans_2$, $Trans_3$, $Trans_4$, $Trans_5$, and $Trans_6$, and the *mtfu* values in these transactions are 21.44, 1.98, 45.42, 47.36, and 23.36, respectively. And then, STP_{all} is P_2, and the pre-defined threshold is 40%. The *tfu* values in the range from STP_{all} to LTP_{all} are 1.98, 53.34, 47.36, and 26, respectively. Consequently, the *tfuub* of 1-itemset {*A.Low*} is 1.98 + 53.34 + 47.36 + 26 (= 128.68). The *tfuubr* value of {*A.Low*} is (21.44+1.98+45.42+47.36+23.36)/128.68 = 108.4% which is larger than predefined threshold. Therefore, itemset {*A.Low*} is inserted into the $HTFUUBI_1$ table. The other fuzzy 1-itemsets are processed likewise. The results are shown in Table 6.

Step 8: Remove those fuzzy itemsets not included in the $HTFUUBI_1$ table from Table 4. For instance, {*A.Middle*} is not in the $HTFUUBI_1$ table. Consequently, {*A.Middle*} and its fuzzy utility value in each transaction must be removed. The results are shown in Table 7.

Step 9: The fuzzy 1-itemsets in Table 6 are then sorted according to the descending order of the frequency of each fuzzy 1-itemset. For instance, the frequency of {*A.Low*} is 5 because {*A.Low*} exists in $Trans_2$, $Trans_3$, $Trans_4$, $Trans_5$, and $Trans_6$ in Table 7. The sorted results are shown in Table 8 as the header table.

Steps 10 to 11: The difference between our proposed algorithm and the FP-Tree [10] is to use the values of

TABLE 5. All fuzzy utility values for all transactions.

TID	*A*			*B*			*C*		
	L	*M*	*H*	*L*	*M*	*H*	*L*	*M*	*H*
$Trans_1$	0	12	0	0	0	0	5.36	0	0
$Trans_2$	5.36	2.64	0	16.08	7.92	0	0	0	0
$Trans_3$	0.66	0	0	0	0	0	1.32	0	0
$Trans_4$	6	0	0	18	0	0	10.72	5.28	0
$Trans_5$	6	0	0	0	36	0	0	0	0
$Trans_6$	6	0	0	0	0	0	0	0	0

D			*E*			*tfu*	*mtfu*
L	*M*	*H*	*L*	*M*	*H*		
0	0	0	0	0	0	17.36	17.36
0	0	0	0	0	0	32	21.44
0	0	0	0	0	0	1.98	1.98
5.36	2.64	0	5.36	0	0	53.34	45.42
0	0	0	5.36	0	0	47.36	47.36
5.36	2.64	0	12	0	0	26	23.36

TABLE 6. The $HTFUUBI_1$ table.

fuzzy itemset	*mtfu*	*frequency*
A.Low	139.56	5
B.Low	66.86	2
B.Middle	68.8	2
C.Low	64.76	3
D.Low	68.78	2
D.Middle	68.78	2
E.Low	116.14	3

TABLE 7. Fuzzy utility values in transactions after removing unsuitable fuzzy items.

TID	A	B		C	D		E	tfu	mtfu
	L	L	M	L	L	M	L		
$Trans_1$	0	0	0	5.36	0	0	0	17.36	17.36
$Trans_2$	5.36	16.08	7.92	0	0	0	0	32	21.44
$Trans_3$	0.66	0	0	1.32	0	0	0	1.98	1.98
$Trans_4$	6	18	0	10.72	5.36	2.64	5.36	53.34	45.42
$Trans_5$	6	0	36	0	0	0	5.36	47.36	47.36
$Trans_6$	6	0	0	0	5.36	2.64	12	26	23.36

TABLE 8. Header table in *FHTFUP* tree.

fuzzy itemset	mtfu	frequency
A.Low	139.56	5
C.Low	64.76	3
E.Low	116.14	3
B.Low	66.86	2
B.Middle	68.8	2
D.Low	68.78	2
D.Middle	68.78	2

itemset	mtfu	frequency
A.L	139.56	5
C.L	64.76	3
E.L	116.14	3
B.L	66.86	2
B.M	68.8	2
D.L	68.78	2
D.M	68.78	2

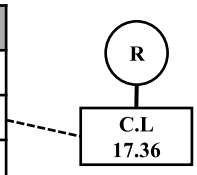

FIGURE 2. $Trans_1$ is inserted into the tree.

the nodes of the *FHTFUP* tree with the *mtfu* to replace the item frequencies in the FP-Tree. All transactions in Table 7 are inserted into the *FHTFUP* tree tuple by tuple. The construction process is stated below.

The first transaction is then inserted into the *FHTFUP* tree as the first branch. Take $Trans_1$ as an example. This transaction $Trans_1$ just contains a fuzzy item $\{C.Low\}$, which appears in $Trans_1$, and the *mtfu* value of $Trans_1$ is 17.36. The fuzzy item $\{C.Low\}$ is inserted into the *FHTFUP* tree as the child of the root, and the node is assigned the corresponding *mtfu* value. The results after inserting the first transaction are shown in Figure 2.

The second transaction is next processed. $Trans_2$ contains fuzzy items $\{A.Low\}$, $\{B.Low\}$ and $\{B.Middle\}$ and the *mtfu* value of $Trans_2$ is 21.44. $\{A.Low\}$ is then inserted into the *FHTFUP* tree as the second branch because the node of the fuzzy item $\{A.Low\}$ does not share the same prefix with the first transaction in the *FHTFUP* tree. The node $\{A.Low\}$ is created to connect with the root and attached the *mtfu* value of $Trans_2$, which is 21.44. The next node, which is a fuzzy

itemset	mtfu	frequency
A.L	139.56	5
C.L	64.76	3
E.L	116.14	3
B.L	66.86	2
B.M	68.8	2
D.L	68.78	2
D.M	68.78	2

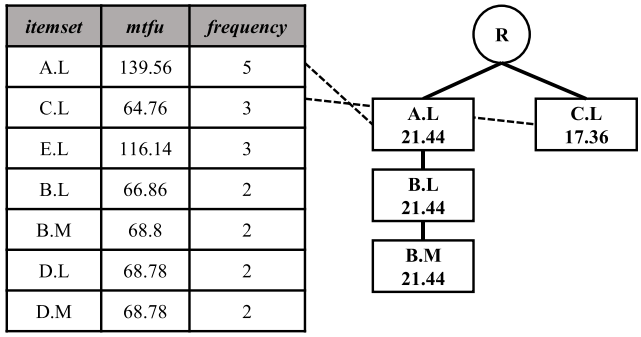

FIGURE 3. $Trans_2$ is inserted into the tree.

itemset	mtfu	frequency
A.L	139.56	5
C.L	64.76	3
E.L	116.14	3
B.L	66.86	2
B.M	68.8	2
D.L	68.78	2
D.M	68.78	2

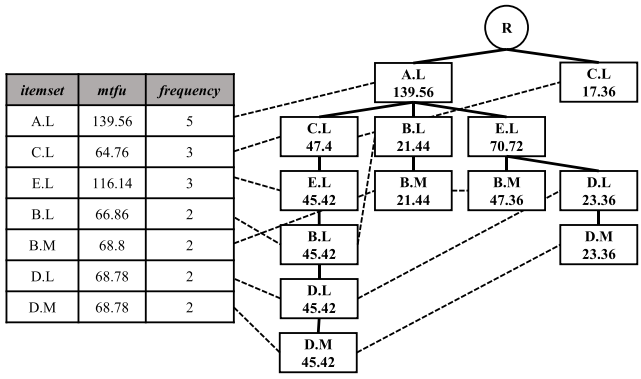

FIGURE 4. The final results for the tree.

item $\{B.Low\}$ is then inserted as the child of the first node $\{A.Low\}$. The same procedure is processed for $\{B.Middle\}$. Each node in the branch is attached to the *mtfu* value of $Trans_2$. The results after inserting the second transaction are shown in Figure 3.

The remaining transactions in Table 7 are processed in the same way. The final result of the *FHTFUP* tree is shown in Figure 4.

After the *FHTFUP* tree is constructed, the candidate *HTFUIs* with one or more fuzzy items can then be found in a similar way to the FP-Growth [10]. The fuzzy 1-itemsets of header table in Table 8 are then processed bottom-up and one by one. The corresponding conditional temporal fuzzy pattern tree is thus build from the prefix paths of the item in the *FHTFUP* tree. In this case, itemset $\{D.Middle\}$ is first processed. For node $\{D.Middle\}$, two paths in Figure 4 can be derived: $[\{A.Low\}, \{C.Low\}, \{E.Low\}, \{B.Low\}, \{D.Low\}, \{D.Middle\}]$, $[\{A.Low\}, \{E.Low\}, \{D.Low\}, \{D.Middle\}]$.

After completing the conditional temporal fuzzy utility pattern tree for fuzzy 1-itemset $\{D.Middle\}$, the fuzzy itemsets with $\{D.Middle\}$ can then be generated by the recursive method of the FP-Growth. And generated candidates are check whether their *mtfu* values are satisfying the pre-defined threshold in temporal fuzzy utility mining, the fuzzy 1-itemset $\{D.Low\}$ in Figure 5 needs to be removed because of $\{D.Low\}$ and $\{D.Middle\}$ that have identical item D. After deleting $\{D.Low\}$, the conditional temporal fuzzy utility pattern tree for fuzzy 1-itemset $\{D.Middle\}$ is shown in Figure 6. Besides, it can be observed that STP_{all}

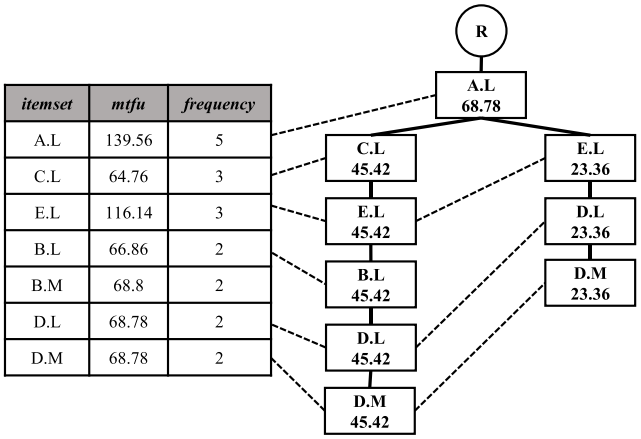

FIGURE 5. The conditional temporal fuzzy utility pattern tree for {D.Middle}.

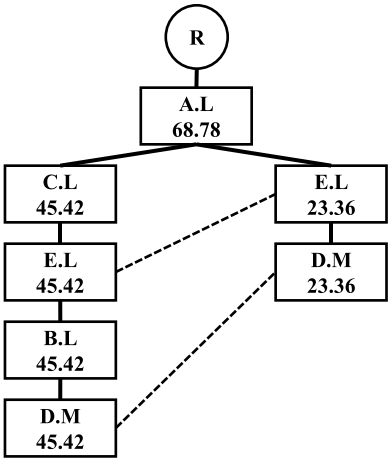

FIGURE 6. The conditional temporal fuzzy utility pattern tree with {D.Middle} after deleting {D.Low}.

is P_2, and the pre-defined threshold is 40%. The *tfu* values in the range from STP_{all} to LTP_{all} are 1.98, 53.34, 47.36, and 26. Therefore, the threshold value is calculated as (1.98+53.34+ 47.36 + 26)*40% = 51.472.

Next, all fuzzy 1-itemsets in Figure 6 need to check whether their values are satisfying threshold value. In this case, the two paths in Figure 6 have the node {*D.Middle*}. Their value for {*D.Middle*} in the two branches is 45.42 + 23.36(= 68.78), which is larger than the threshold value (51.472). Therefore, keep this node {*D.Middle*} in this tree. For another example, the left path in Figure 6 has the node {*B.Low*}. Their value for {*B.Low*} in the left branch is 45.42, which is smaller than the threshold value (51.472), so it is omitted in this tree. Similarly, remaining fuzzy 1-itemsets in Figure 5 need to be check in the same way, the final result for conditional temporal fuzzy utility pattern tree for fuzzy 1-itemset {*D.Middle*} is shown in Figure 7.

Next, the conditional temporal fuzzy utility pattern tree for fuzzy 1-itemset {*D.Middle*} can derive the candidate *HTFUIs*, which are {*E.Low*, *D.Middle*}, {*A.Low*, *D.Middle*}, {*A.Low*, *E.Low*, *D.Middle*}, and {*D.Middle*}. The conditional

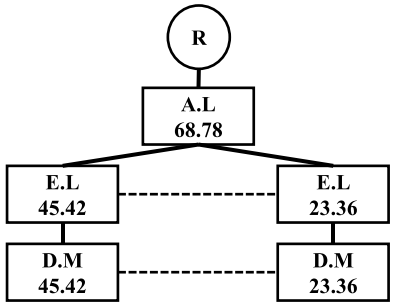

FIGURE 7. The conditional temporal fuzzy utility pattern tree with {D.Middle} after removing unsuitable fuzzy itemsets.

temporal fuzzy utility pattern tree for other fuzzy 1-itemsets is processed in the same way.

Step 12: After completing the above step, all candidate *HTFUIs* can be found, and the rescan database is executed to get the *tfu* values of the candidate *HTFUIs*.

Step 13: Using {*E.Low*, *D.Middle*} as an example. It can be observed that the *STP* value of the two items, *E* and *D*, is P_2. The $LTP_{\{E.Low,D.Middle\}}$ are P_2 to P_4. Therefore, the *tfu* values of each transaction in $LTP_{\{E.Low,D.Middle\}}$ are 1.98 + 53.34 + 47.36 + 26 = 128.68. According to the ninth definition, the $fu_{\{E.Low,D.Middle\}}$ value can be gotten. First, the fuzzy values of *E.Low* and *D.Middle* are {0.67, 0.33} in $Trans_4$ and {0.67, 0.33} in $Trans_6$. And then the fuzzy values of {*E.Low*, *D.Middle*} in $Trans_4$ and $Trans_6$ are 0.33 and 0.33, respectively. The fuzzy utility value of {*E.Low*, *D.Middle*} is calculated as 0.33*[(4*2) + (2*4)] + 0.33*[(4*2) + (3*4)] = 11.88. As such, $tfu_{\{E.Low,D.Middle\}}$ is 11.88 / 128.68, which is smaller than the threshold. It is not *HTFUI*. The same process can handle other itemsets.

VI. PERFORMANCE EVALUATION

Extensive experiments, in this section, were used to evaluate the proposed *FHTFUP* approach on some synthetic datasets. Because the temporal fuzzy utility data mining problem is interesting, it considers temporal property, quantity information for each item, its profit, and transformed linguistic terms. The existing *TP-TFU* method [15] is used as a benchmark algorithm for comparison with the proposed *FHTFUP* algorithm.

A. EXPERIMENTAL SETUP AND DATASET DESCRIPTION

Some synthetic datasets produced from [17] and two real datasets [9], [27] were applied to evaluate the performance of the proposed *FHTFUP* and previous *TP-TFU* method on a computer. We implemented two algorithms in Java code and executed on a desktop computer with Dual-Core Processor 3.3GHZ, 16GB RAM, and Windows 7. In those datasets, the quantitative attributes were generated the quantitative values randomly in the [1, 10] interval. The profit values for items were set from 1 to 1000 at random. Besides, the membership functions of all items were used 3-fuzzy terms in Figure 1. Time periods are assigned 2. The parameters of the synthetic databases are described in Table 9.

TABLE 9. Synthetic dataset parameters.

T	The average length of items in a transaction
I	The average length of maximal potentially frequent itemsets
N	The total number of different items in a dataset
D	The number of transactions in a dataset

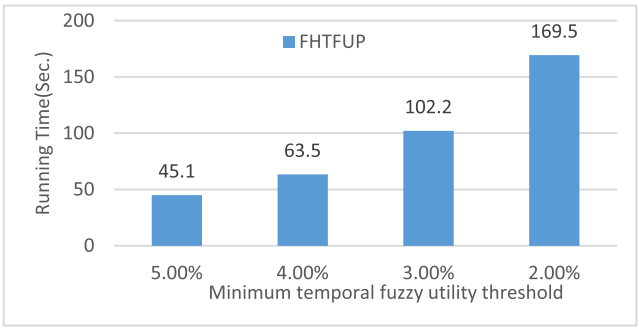

FIGURE 8. Running time of the proposed method on T7I4N4KD150K.

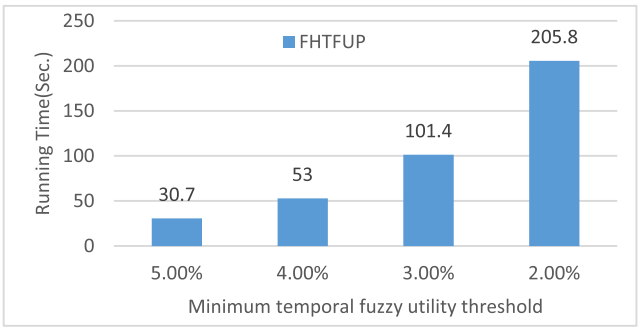

FIGURE 9. Running time of the proposed method on T7I4N4KD250K.

B. THE PERFORMANCE OF EXECUTION TIME

This subsection compared the computation time of the proposed *FHTFUP* approach on different test datasets with varying D, T, N parameters was evaluated.

For example, the computation time of the proposed *FHTFUP* approach is shown in Figures 8 to 10. The size of the synthetic datasets is changed, and the pre-defined minimum temporal fuzzy utility threshold is varied from 2% to 5% with a 1% increment on different test datasets. Other parameters, T and N, are set to 7 and 4K, respectively.

The execution times along with distinct thresholds, are shown in Figures 8 to 10. As you can see, the proposed *FHTFUP* algorithm is slower, especially when the size of the test datasets as the D parameter increased. On the same dataset, the proposed approach was also slower when the threshold decreased. It is because when the size of the synthetic datasets is set larger, the proposed *FHTFUP* method has to keep many fuzzy 1-itemsets of different transactions in the *FHTFUP* tree, which is more time-consuming.

For example, Figures 11 to 13 show the execution time of the proposed *FHTFUP* method. The total items N and the number of transaction D in test datasets are fixed, the average

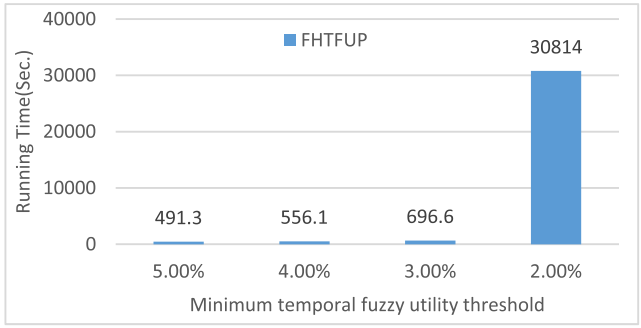

FIGURE 10. Running time of the proposed method on T7I4N4KD300K.

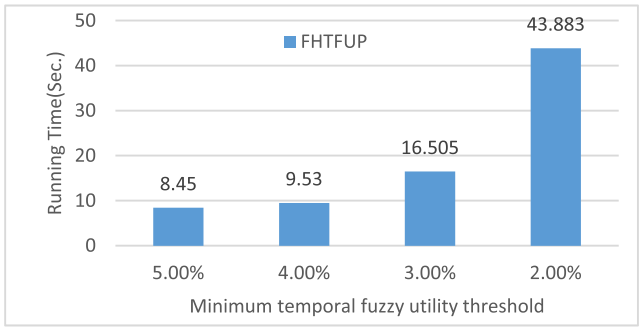

FIGURE 11. Running time of the proposed method on T8I4N4KD200K.

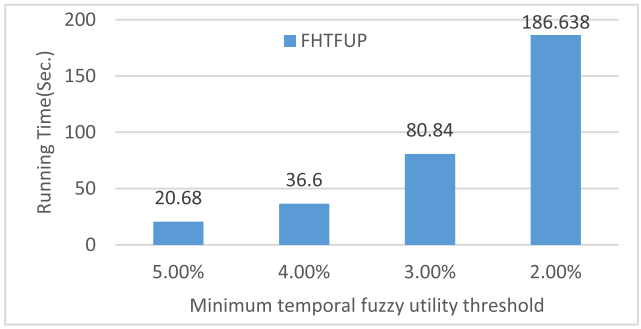

FIGURE 12. Running time of the proposed method on T9I4N4KD200K.

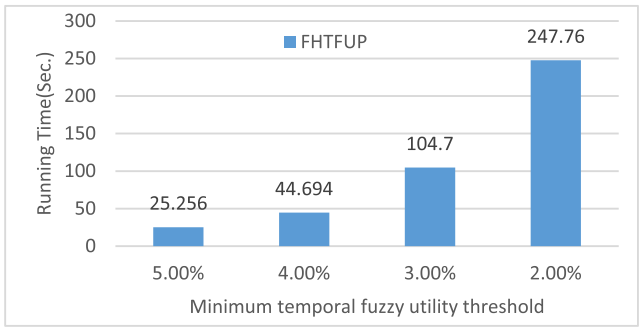

FIGURE 13. Running time of the proposed method on T10I4N4KD200K.

items in a transaction T are changed, and the pre-defined minimum temporal fuzzy utility threshold is also varied from 2% to 5% with a 1% increment on different test datasets.

The running times along with the different T are shown in Figures 11 to 13. We observed that the proposed *FHTFUP*

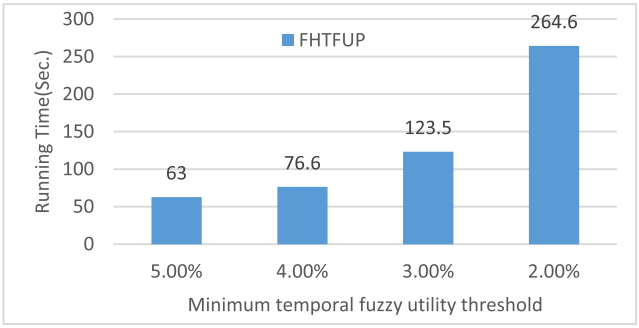

FIGURE 14. Running time of the proposed method on T7I4N5KD100K.

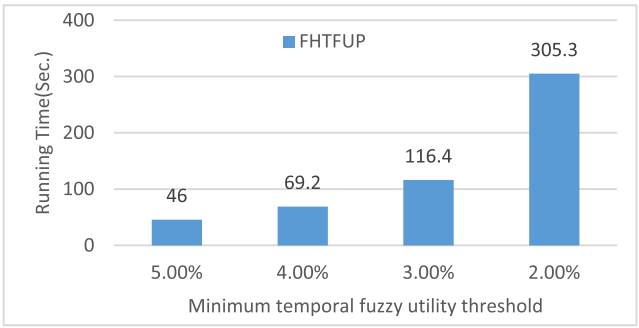

FIGURE 15. Running time of the proposed method on T7I4N6KD100K.

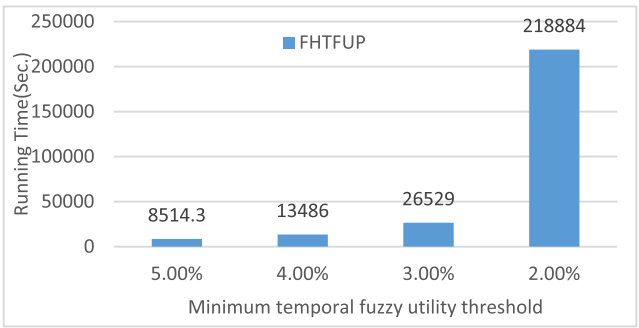

FIGURE 16. Running time of the proposed method on T7I4N7KD100K.

algorithm is slower when the T parameter is increasing. This is because each transaction in a dataset included more distinct items. For each one on three different datasets, the proposed approach was slower when the threshold decreased. In other words, when the threshold decreased, a large amount of candidate *k-HTFUIs* derived from the proposed tree was generated.

For example, the execution time of the proposed *FHTFUP* method on three different datasets are revealed in Figures 14 to 16. The number of items in a dataset as the N parameter is changed except the other parameters (T and D) are fixed, and the pre-defined minimum temporal fuzzy utility threshold is set from 2% to 5%.

The running times along with different items in a dataset are shown in Figure 14 to 16. It can be seen that the proposed *FHTFUP* method is slower when the number of items in a dataset, considered as the N parameter raised. This is because more different fuzzy regions derived from distinct

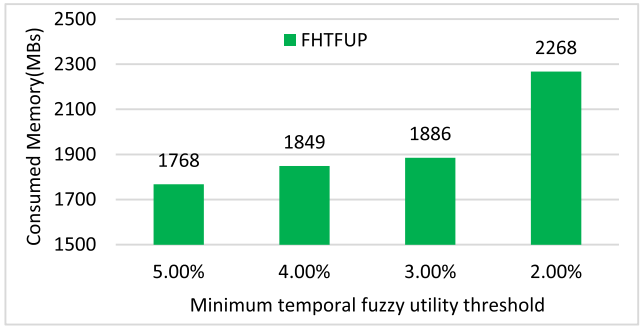

FIGURE 17. Consumed memory of the proposed method on T7I4N4KD150K.

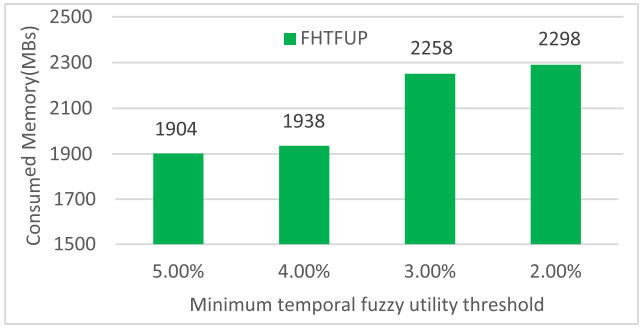

FIGURE 18. Consumed memory of the proposed method on T7I4N4KD250K.

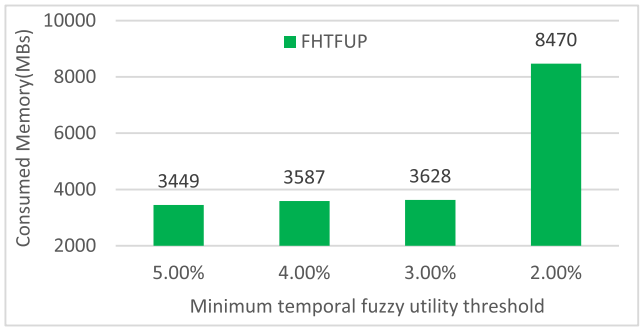

FIGURE 19. Consumed memory of the proposed method on T7I4N4KD300K.

items according to membership functions were generated. Also, you can see that the computation time of the proposed method was slower in a dataset when the threshold decreased. This is because when the N parameter raised, many generated fuzzy 1-itemsets with satisfying the threshold were used to build the proposed tree, and then many candidate *HTFUIs* are produced.

C. THE PERFORMANCE OF CONSUMED MEMORY

With varying D, T, N parameters, this subsection discusses the peak consumed memory of the proposed *FHTFUP* approach on varying datasets as an essential measure.

For example, Figures 17 to 19 show the consumed memory of the proposed *FHTFUP* approach. Except for T and N parameters, the size of the synthetic datasets is unfixed from 150K to 300K, and the pre-defined minimum temporal fuzzy

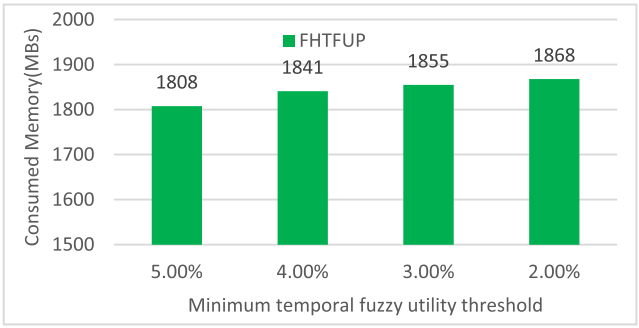

FIGURE 20. Consumed memory of the proposed method on T8I4N4KD200K.

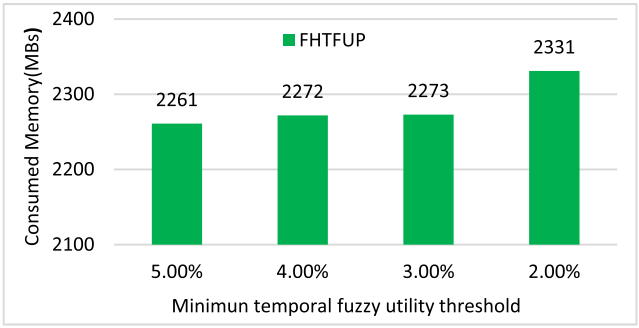

FIGURE 21. Consumed memory of the proposed method on T9I4N4KD200K.

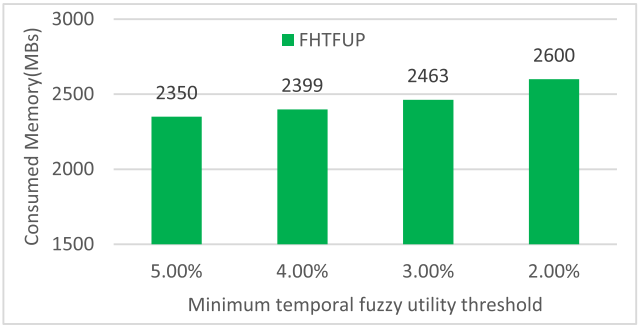

FIGURE 22. Consumed memory of the proposed method on T10I4N4KD200K.

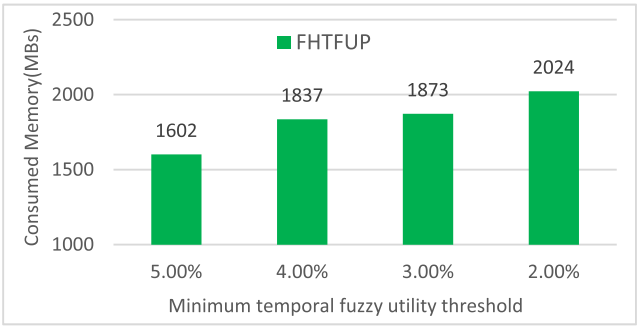

FIGURE 23. Consumed memory of the proposed method on T7I4N5KD100K.

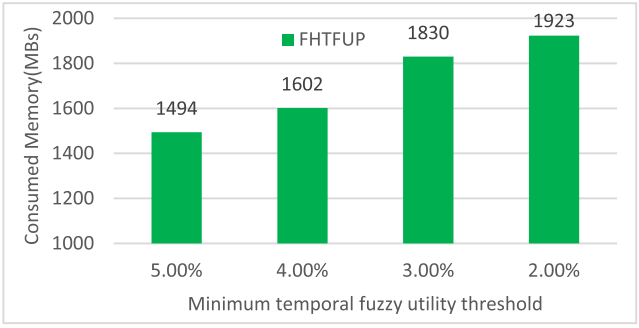

FIGURE 24. Consumed memory of the proposed method on T7I4N6KD100K.

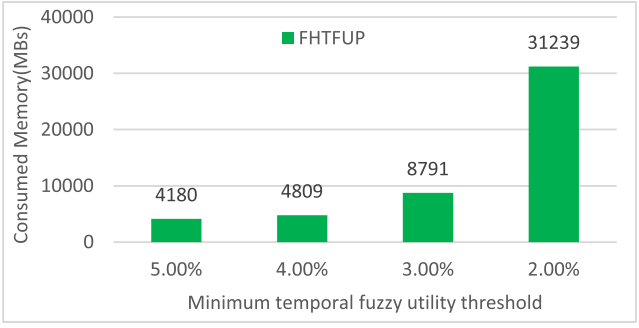

FIGURE 25. Consumed memory of the proposed method on T7I4N7KD100K.

utility threshold is varied from 2% to 5% with a 1% increment on different synthetic datasets.

The consumed memory along with the different number of transactions was shown in Figure 17 to 19. Along with the D value increased, the proposed method needed memory space more. Besides, as you can see that when the threshold is raised, fewer *HTFUIs* are produced by the proposed *FHTFUP* algorithm, and the consumed memory was also decreased. In summary, those results showed that used memory space increased in the same threshold of three different datasets when the D parameter raised. The reason is that numerous fuzzy 1-itemsets in the different transactions were then used to build the proposed tree, and thus algorithm needs more memory.

For example, the memory consumption of the proposed *FHTFUP* method on three different T of datasets was shown in Figures 20 to 22. The total items and the number of transactions in synthetic datasets are fixed, the average items in a transaction are changed, and the pre-defined minimum temporal fuzzy utility threshold is set from 2% to 5%.

The consumed memory along with the different T parameters was shown in Figure 20 to 22. As you can see, the proposed *FHTFUP* method on three different datasets consumed memory when $T = 8$, $T = 9$, and $T = 10$. It is clear that the *FHTFUP* method with larger T needed more memory space than that with smaller T. When the T parameter grew, it meant the different items in a transaction increased. Hence, many fuzzy 1-items were kept in the proposed tree, and the memory space also raised. On one of three different datasets, the proposed algorithm consumed more memory when the threshold declined because of more fuzzy itemsets with satisfying the threshold.

For example, Figures 23 to 25 show the consumed memory of the proposed *FHTFUP* method. The number of items in a

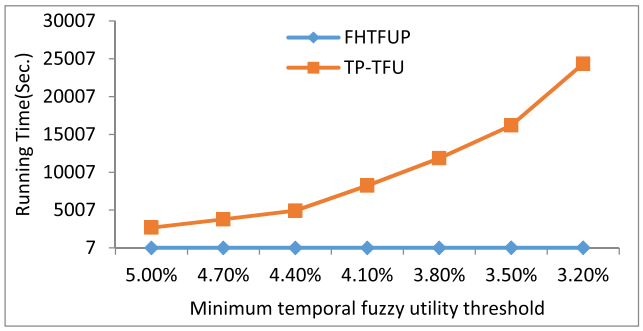

FIGURE 26. Running time for two methods on T6I4N4KD200K.

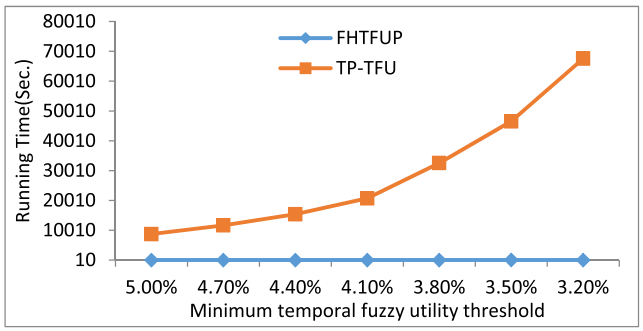

FIGURE 27. Running time for two methods on T7I4N4KD200K.

dataset is changed except the other parameters are fixed and the pre-defined minimum temporal fuzzy utility threshold is set from 2% to 5%.

Used memory space along with the different number of items in a dataset was shown in Figures 23 to 25. We observed that the proposed *FHTFUP* algorithm needed more memory space when the *N* parameter raised. This main reason is that when the number of items in a dataset was raised, the produced fuzzy 1-itemsets with their *tfuubr* values satisfying the threshold value became more and more. The consumed memory in each dataset was larger when the threshold decreased.

D. COMPARISON WITH TP-TFU

To gain insights into the advancement of the proposed *FHTFUP* method, this subsection compared the running time and memory consumption by the two algorithms. Two test datasets of T6I4N4KD200K and T7I4N4KD200K were then performed. Here the minimum temporal fuzzy utility threshold was set from 5% to 3.2%. The experimental results are displayed in Figures 26 to 29.

The results in Figures 26 and 27 showed that the running time of the *FHTFUP* approach was faster than that of the *TP-TFU* approach when the threshold varied from 5% to 3.2%. In Figures 28 and 29, the memory space of the *FHTFUP* approach is less than that of the *TP-TFU* approach when the threshold varied from 5% to 3.2%.

E. THE PERFORMANCE EVALUATION ON REAL DATASETS

In this subsection, two real datasets, Foodmart [27] and Mushroom [9], were used to evaluate the execution time and memory consumption by the two methods.

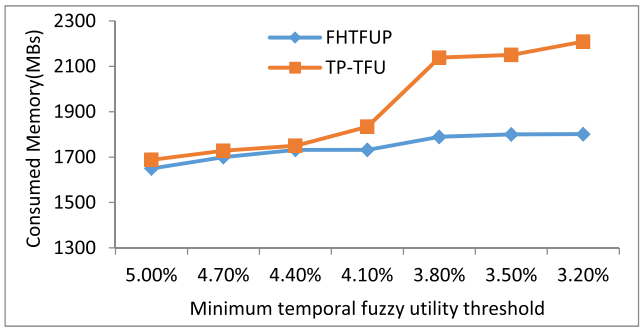

FIGURE 28. Used memory for two methods on T6I4N4KD200K.

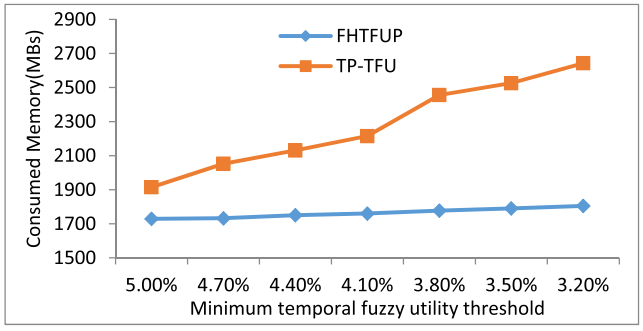

FIGURE 29. Used memory for two methods on T7I4N4KD200K.

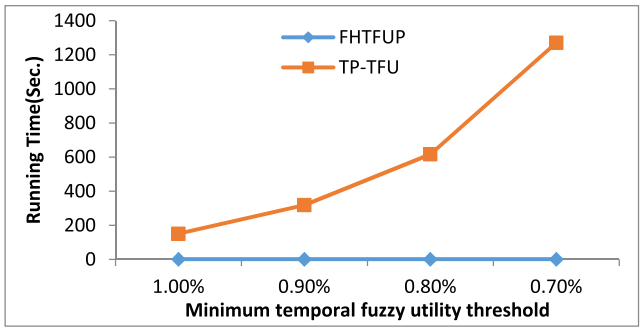

FIGURE 30. Running time for two methods on Foodmart.

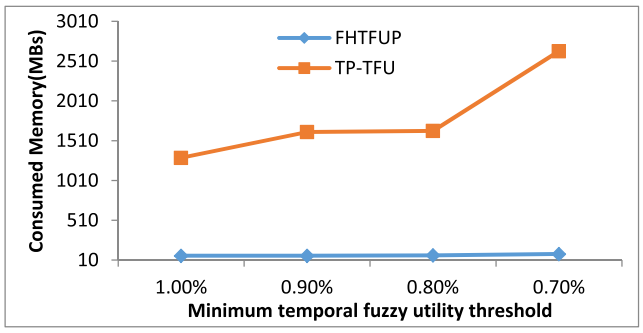

FIGURE 31. Used memory for two methods on Foodmart.

In the Foodmart dataset, the minimum temporal fuzzy utility threshold was set at 1% to 0.7%. The experimental results on the running time and used memory are shown in Figures 30 and 31.

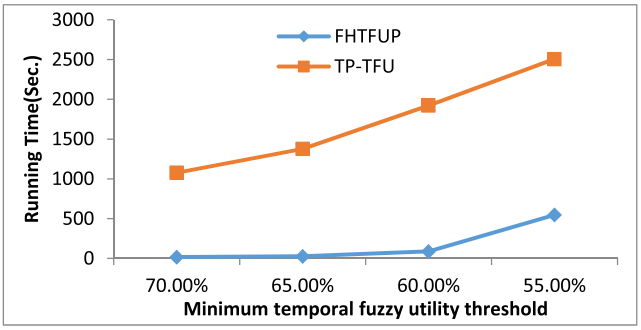

FIGURE 32. Running time for two methods on Mushroom.

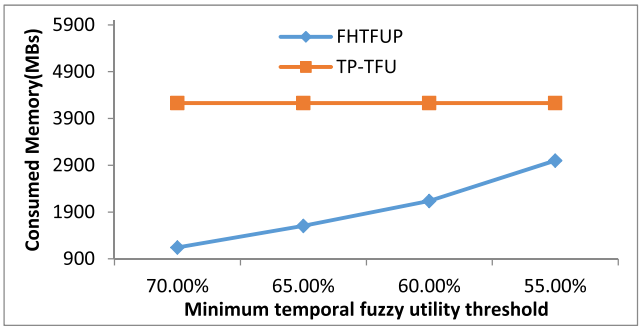

FIGURE 33. Used memory for two methods on Mushroom.

In the Mushroom dataset, the minimum temporal fuzzy utility threshold was set at 70% to 55%. The experimental results on the running time and used memory are shown in Figures 32 and 33.

From Figures 30 and 32, it can be observed that the *FHTFUP* approach ran faster than the *TP-TFU* approach when the threshold decreased.

The results in Figure 31 showed the memory consumption of the *FHTFUP* approach was less than that of the *TP-TFU* approach when the threshold varied from 1% to 0.7%. In Figure 33, the memory usage of the *TP-TFU* method was always the same because it had used the maximum memory (a quarter of the actual physical memory) in Java Virtual Machine. Therefore, we observed that the *TP-TFU* algorithm needed more memory space than the proposed method.

VII. CONCLUSION

We proposed, in this paper, the *FHTFUP* algorithm based on the two-phase upper bound model [15] and FP-tree [10] to find *HTFUIs*. The model can find all possible itemsets (*HTFUUBIs*) with keeping the downward-closure property. Using FP-tree is to decrease explosive candidates compared with the generate-and-test method [15]. On the experimental evaluation, corresponding varied parameters on the synthesized and two real datasets were tested for the two algorithms. The results show that the execution efficiency of the proposed approach is less than that of the previous method [15] on different datasets.

In the future, we will focus on this study to search for how to reduce running time and consumed memory. Larger databases are used to evaluate the proposed method with other metrics, such as the execution time and memory used. Besides, handling the maintenance problems in this topic will also be considered. For example, transactions in a dataset are deleted due to the overdue, corrected transactions when there are errors, or inserted new transactions if needed.

REFERENCES

[1] R. Agrawal and R. Srikant, "Fast algorithm for mining association rules," in *Proc. 20th Int. Conf. Very Large Data Bases*, vol. 1215, 1994, pp. 487–499.
[2] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," *ACM SIGMOD Rec.*, vol. 22, no. 2, pp. 207–216, Jun. 1993.
[3] R. Agrawal, T. Imieliński, and A. Swami, "Database mining: A performance perspective," *IEEE Trans. Knowl. Data Eng.*, vol. 5, no. 6, pp. 914–925, Dec. 1993.
[4] R. Chan, Q. Yang, and Y.-D. Shen, "Mining high utility itemsets," in *Proc. 3rd IEEE Int. Conf. Data Mining*, 2003, pp. 19–26.
[5] C.-Y. Chang, M.-S. Chen, and C.-H. Lee, "Mining general temporal association rules for items with different exhibition periods," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2002, pp. 59–66.
[6] C.-H. Chen, G.-C. Lan, T.-P. Hong, and S.-B. Lin, "Mining fuzzy temporal association rules by item lifespans," *Appl. Soft Comput.*, vol. 41, pp. 265–274, Apr. 2016.
[7] D. Chen, S. L. Sain, and K. Guo, "Data mining for the online retail industry: A case study of RFM model-based customer segmentation using data mining," *J. Database Marketing Customer Strategy Manage.*, vol. 19, no. 3, pp. 197–208, Sep. 2012.
[8] D. Enke and S. Thawornwong, "The use of data mining and neural networks for forecasting stock market returns," *Expert Syst. Appl.*, vol. 29, no. 4, pp. 927–940, Nov. 2005.
[9] *Frequent Itemsets Mining Dataset Repository*. Accessed: Feb. 28, 2019. [Online]. Available: http://fimi.cs.helsinki.fi/data/
[10] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, vol. 29, 2000, pp. 1–12.
[11] L. Hirschman, J. C. Park, J. Tsujii, L. Wong, and C. H. Wu, "Accomplishments and challenges in literature data mining for biology," *Bioinformatics*, vol. 18, no. 12, pp. 1553–1561, Dec. 2002.
[12] T. P. Hong, C. S. Kuo, and S. C. Chi, "Mining fuzzy sequential patterns from quantitative data," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 1999, pp. 962–966.
[13] T.-P. Hong, C.-S. Kuo, and S.-C. Chi, "Trade-off between computation time and number of rules for fuzzy mining from quantitative data," *Int. J. Uncertainty, Fuzziness Knowl.-Based Syst.*, vol. 9, no. 5, pp. 587–604, Oct. 2001.
[14] T. P. Hong, C. Y. Lin, W. M. Huang, S. M. Li, S. L. Wang, and J. C. W. Lin, "Mining temporal fuzzy utility itemsets by tree structure," in *Proc. IEEE Int. Conf. Big Data*, Dec. 2019, pp. 2659–2663.
[15] W.-M. Huang, T.-P. Hong, G.-C. Lan, M.-C. Chiang, and J. C.-W. Lin, "Temporal-based fuzzy utility mining," *IEEE Access*, vol. 5, pp. 26639–26652, 2017.
[16] W. Huo, X. Feng, and Z. Zhang, "An efficient approach for incremental mining fuzzy frequent itemsets with FP-tree," *Int. J. Uncertainty, Fuzziness Knowl.-Based Syst.*, vol. 24, no. 3, pp. 367–386, Jun. 2016.
[17] IBM Quest Data Mining Projection. (1996). *Quest Synthetic Data Generation Code*. [Online]. Available: http://www.almaden.ibm.com/cs/quest/syndata.html
[18] G.-C. Lan, C.-H. Chen, T.-P. Hong, and S.-B. Lin, "A fuzzy approach for mining general temporal association rules in a publication database," in *Proc. 11th Int. Conf. Hybrid Intell. Syst. (HIS)*, Dec. 2011, pp. 611–615.
[19] G.-C. Lan, T.-P. Hong, and V. S. Tseng, "Discovery of high utility itemsets from on-shelf time periods of products," *Expert Syst. Appl.*, vol. 38, no. 5, pp. 5851–5857, May 2011.
[20] G.-C. Lan, T.-P. Hong, Y.-H. Lin, and S.-L. Wang, "Fuzzy utility mining with upper-bound measure," *Appl. Soft Comput.*, vol. 30, pp. 767–777, May 2015.
[21] C.-H. Lee, C.-R. Lin, and M.-S. Chen, "On mining general temporal association rules in a publication database," in *Proc. IEEE Int. Conf. Data Mining*, Nov. 2001, pp. 337–344.

[22] C.-W. Lin, T.-P. Hong, and W.-H. Lu, "Linguistic data mining with fuzzy FP-trees," *Expert Syst. Appl.*, vol. 37, no. 6, pp. 4560–4567, Jun. 2010.
[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] Y. Liu, W. Liao, and A. Choudhary, "A fast high utility itemsets mining algorithm," in *Proc. 1st Int. Workshop Utility-Based Data Mining*, 2005, pp. 90–99.
[25] Y. Liu, W. Liao, and A. Choudhary, "A two-phase algorithm for fast discovery of high utility itemsets," in *Proc. Pacific–Asia Conf. Knowl. Discovery Data Mining*, vol. 3518, 2005, pp. 689–695.
[26] J. Luo and S. M. Bridges, "Mining fuzzy association rules and fuzzy frequency episodes for intrusion detection," *Int. J. Intell. Syst.*, vol. 15, no. 8, pp. 687–703, 2000.
[27] Microsoft. *Example Database Foodmart of Microsoft Analysis Services*. Accessed: Feb. 28, 2019. [Online]. Available: http://msdn.microsoft.com/en-us/library/aa217032(SQL.80).aspx
[28] 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, doi: 10.1109/ACCESS.2020.2979289.
[29] Y. Qiu, Y. J. Lan, and Q. S. Xie, "An improved algorithm of mining from FP-tree," in *Proc. Int. Conf. Mach. Learn. Cybern.*, vol. 3, Aug. 2004, pp. 1665–1670.
[30] B. E. Shie, V. S. Tseng, and P. S. Yu, "Online mining of temporal maximal utility itemsets from data streams," in *Proc. ACM Symp. Appl. Comput.*, 2010, pp. 1622–1626.
[31] S. G. Totad, R. B. Geeta, and P. V. G. D. P. Reddy, "Batch incremental processing for FP-tree construction using FP-growth algorithm," *Knowl. Inf. Syst.*, vol. 33, no. 2, pp. 475–490, Nov. 2012.
[32] B. Vo, L. T. T. Nguyen, T. D. D. Nguyen, P. Fournier-Viger, and U. Yun, "A multi-core approach to efficiently mining high-utility itemsets in dynamic profit databases," *IEEE Access*, vol. 8, pp. 85890–85899, 2020, doi: 10.1109/ACCESS.2020.2992729.
[33] X. Wang and L. Wang, "Incremental mining of high utility co-locations from spatial database," in *Proc. IEEE Int. Conf. Big Data Smart Comput.*, Feb. 2017, pp. 215–222.
[34] C.-H. Weng, "Identifying association rules of specific later-marketed products," *Appl. Soft Comput.*, vol. 38, pp. 518–529, Jan. 2016.
[35] C.-H. Weng and Y.-L. Chen, "Mining fuzzy association rules from uncertain data," *Knowl. Inf. Syst.*, vol. 23, no. 2, pp. 129–152, May 2010.
[36] H. Yao, H. J. Hamilton, and C. J. Butz, "A foundational approach to mining itemset utilities from databases," in *Proc. 3rd SIAM Int. Conf. Data Mining*, Apr. 2004, pp. 482–486.
[37] J. S. Yeh, Y. C. Li, and C. C. Chang, "Two-phase algorithm for a novel utility-frequent mining model," in *Proc. Pacific–Asia Conf. Knowledge Discovery Data Mining*, vol. 4819, 2007, pp. 433–444.
[38] L. A. Zadeh, "Fuzzy sets," *Inf. Control*, vol. 8, no. 3, pp. 338–353, 1965.
[39] Z. Zhang, J. Huang, J. Hao, J. Gong, and H. Chen, "Extracting relations of crime rates through fuzzy association rules mining," *Appl. Intell.*, vol. 50, no. 2, pp. 448–467, 2020.

TZUNG-PEI HONG (Senior Member, IEEE) 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 at the Department of Computer Science, Chung-Hua Polytechnic Institute, from 1992 to 1994, and at the Department of Information Management, I-Shou University, from 1994 to 2001. He was in charge of the whole computerization and library planning for 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, Taiwan, 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 and the Director of the AI Research Center, National University of Kaohsiung. He is also a Joint Professor at 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. His current research interests include knowledge engineering, data mining, soft computing, management information systems, and www applications. He is also a Board Member of more than 40 journals and a Program Committee Member of more than 900 conferences. He received the first National Flexible Wage Award from the Ministry of Education in Taiwan.

CHENG-YU LIN received the B.S. degree in computer science from Nation Pingtung University, in 2017, and the M.S. degree in computer science and information engineering from National Sun Yat-sen University, in 2019. He is currently at Portwell as a Software Engineer. His research interests include data mining and fuzzy theory when he was studying in graduate school and data hiding and fragile water marker when he was studying in university.

WEI-MING HUANG received the B.S. degree from the Department of Computer Science and Information Engineering, Da-Yeh University, Yuanlin, Taiwan, in 2001, the M.S. degree from the Department of Computer and Communication Engineering, National Kaohsiung First University of Science and Technology, Kaohsiung, Taiwan, in 2003, and the Ph.D. degree from the Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, in 2019. He has been serving at the Department of Electrical and Control, China Steel, Taiwan, since 2005. His research interest includes data mining.

KATHERINE SHU-MIN LI (Senior Member, IEEE) received the B.S. degree in computer science from Rutgers University, New Brunswick, NJ, USA, and the M.S. degree in computer science and the Ph.D. degree in electrical engineering from National Chiao-Tung University, Taiwan, in 2001 and 2006, respectively.

She is currently a Full Professor with the Department of Computer Science and Engineering, National Sun Yat-Sen University, Kaohsiung, Taiwan. Her current research interests include wafermap pattern recognition by machine learning techniques, design for yield (DfY), foundry automation, interposer test, 2.5D/3D/SiP IC test, microfluidic chip synthesis and test/diagnosis/fault tolerance, hardware security and trojan, design for security (DfS), oscillation ring test schemes, and 5G/RF test. She is a member of the IEEE Education Society, the IEEE Circuits and Systems Society, the Association for Computing Machinery (ACM), and the ACM Special Interest Group on Design Automation and has been a member of the IEEE Women in Engineering (WIE), since October 2013.

LEON SHYUE-LIANG WANG (Member, IEEE) received the Ph.D. degree from The State University of New York at Stony Brook, in 1984. From 1984 to 1994, he was with the University of New Haven and the New York Institute of Technology as an Assistant/Associate Professor. From 1994 to 2002, he was with I-Shou University, Taiwan, where he served as the Director of the Computing Center, the Chairman of the Information Management Department, and the Director of Library. From 2003 to 2007, he rejoined NYIT. From 2009 to 2016, he was a Professor, the Chairman, the Dean of the College of Management, and the Vice President at the National University of Kaohsiung, Taiwan. Since August 2016, he has been the President at the National University of Kaohsiung, where he is currently a Distinguished Professor. He has published over 230 articles in the areas of data mining, privacy preservation, and soft computing. He is a Fellow of the Institute of Engineering and Technology, U.K. He received the 2016 Outstanding Leadership Award from the Chinese American Academic and Professional Society, New York, USA, and a member of the Phi Tau Phi Scholar Honor Society. He was a recipient of the 2011–2014 and 2014–2017 National Flexible Wage Awards from the Ministry of Education in Taiwan. He served as a PC Member and the Session Chair for over 160 international conferences. He is the President of the Taiwanese Association of Social Networks and the Editor-in-Chief of the *International Journal of Information Privacy, Security and Integrity*.

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, in 2010. 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 has published more than 300 research articles in refereed journals, such as the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, the IEEE TRANSACTIONS ON CYBERNETICS, the *ACM Transactions on Knowledge Discovery from Data*, and the *ACM Transactions on Data Science*, and international conferences, such as the IEEE ICDE, the IEEE ICDM, PKDD, and PAKDD. His research interests include data mining, soft computing, artificial intelligence and machine learning, and privacy preserving and security technologies. He is also the Project Co-Leader of well-known SPMF and also the Funder and Project Leader of PPSF Library. He is the Fellow of the IET (FIET) and a Senior Member of the ACM. He is the Editor-in-Chief of the *International Journal of Data Science and Pattern Recognition*, an Associate Editor of IEEE ACCESS and the *Journal of Internet Technology*, and an Editor of *Intelligent Data Analysis* and the *International Journal of Interactive Multimedia and Artificial Intelligence*.