applied sciences

MDPI

Article

A One-Phase Tree-Structure Method to Mine High Temporal Fuzzy Utility Itemsets

Tzung-Pei Hong [1,2,*], Cheng-Yu Lin [2], Wei-Ming Huang [3], Shu-Min Li [2], Shyue-Liang Wang [1] and Jerry Chun-Wei Lin [4]

1 Department of Computer Science and Information Engineering, National University of Kaohsiung, Kaohsiung 811, Taiwan; slwang@nuk.edu.tw
2 Department of Computer Science and Engineering, National Sun Yat-Sen University, Kaohsiung 804, Taiwan; mos.lin@portwell.com.tw (C.-Y.L.); smli@cse.nsysu.edu.tw (S.-M.L.)
3 Department of Electronic and Control, China Steel Corporation, Kaohsiung 812, Taiwan; granthill168@gmail.com
4 Department of Computer Science, Electrical Engineering and Mathematical Sciences, Western Norway University of Applied Science, 5020 Bergen, Norway; jerrylin@ieee.org
* Correspondence: tphong@nuk.edu.tw

Abstract: Compared to fuzzy utility itemset mining (FUIM), temporal fuzzy utility itemset mining (TFUIM) has been proposed and paid attention to in recent years. It considers the characteristics of transaction time, sold quantities of items, unit profit, and transformed semantic terms as essential factors. In the past, a tree-structure method with two phases was previously presented to solve this problem. However, it spent much time because of the number of candidates generated. This paper thus proposes a one-phase tree-structure method to find the high temporal fuzzy utility itemsets in a temporal database. The tree was designed to maintain candidate 1-itemsets with their upper bound values meeting the defined threshold constraint. Besides, each node in this tree keeps the required data of a 1-itemset for mining. We also designed an algorithm to construct the tree and gave an example to illustrate the mining process in detail. Computational experiments were conducted to demonstrate the one-phase tree-structure method is better than the previous one regarding the execution time on three real datasets.

Keywords: fuzzy set; data mining; temporal fuzzy utility mining; temporal database; tree structure

Citation: Hong, T.-P.; Lin, C.-Y.; Huang, W.-M.; Li, S.-M.; Wang, S.-L.; Lin, J.C.-W. A One-Phase Tree-Structure Method to Mine High Temporal Fuzzy Utility Itemsets. *Appl. Sci.* **2022**, *12*, 2821. https://doi.org/10.3390/app12062821

Academic Editor: Juan J. Rodríguez

Received: 5 December 2021
Accepted: 7 March 2022
Published: 9 March 2022

1. Introduction

Frequent itemset mining (FIM) is one of the most popular data-mining problems. It uses a pre-defined threshold to determine the important itemsets in a dataset [1–3]. With the mining procedure, useful and implicit information in a dataset is discovered to help managers make good decisions efficiently. It can also be further utilized to find other kinds of knowledge, such as association rules [4,5] and sequential patterns [6].

In business, frequent itemsets derived by FIM usually have low profits because cheap items are usually sold the most. However, the favorable itemsets may be the ones with high profits, but they often have small counts. For example, in a supermarket, television has a much higher gain than a piece of bread, even when the frequency of the former is much less than that of the latter. Therefore, the combinations of items that can obtain good total profits are more important than those of only high frequencies. Thus, in 2004, the utility itemset mining (UIM) problem was proposed to concern both the amounts of items sold and the unit profits of items while determining desired itemsets [7]. The problem uses a measure for itemsets, and those satisfying the utility threshold constraint are regarded as high utility itemsets. It, however, does not hold the downward-closure property and thus needs more processing time than association-rule mining.

In 2005, Liu et al. proposed a method with two phases to solve the UIM problem [8,9]. They presented an upper-bound measure to keep the anti-monotonic property in the searching process, thus reducing the search space and execution time. In the first phase, a database needs to be processed, and the upper-bound value of each itemset is calculated to decide whether it is a candidate. Thus, unpromising itemsets can be trimmed early. In the second phase, the candidate itemsets kept in phase 1 are checked for whether they are really high utility itemsets by the second scan of the database.

In addition to item amounts and profits, the transaction time is another critical factor to business strategy. Goods in a supermarket are registered with their exhibition time periods. In other words, different goods sold may be on the shelf at different time periods. If all items in a supermarket are considered to be of the same length as that in the whole database, the mined result may cause some biases. The temporal relationships among items sold are complicated and not strictly explained in UIM. Some studies claimed mining approaches to determine the orderly association among items with the temporal relationship. For example, the PPM (progressive partition miner) method suggested by Lee et al. is to find temporal patterns with the exhibition time of items [10]. The SPF (segmented progressive filter) method suggested by Chang et al. is to mine corresponding temporal rules. They considered that items had certain lifespans in which they were available for being purchased [11]. The SLMCM approach suggested by Weng viewed different products with their marked times [12]. He used the marked times for all items and presented a measuring method to mine relevant patterns for avoiding generating useless itemsets.

For helping humans to understand the knowledge easily, the fuzzy set theory [13] was applied to FUIM to interpret quantitative values as linguistic terms. Chen et al. suggested a method to derive fuzzy association rules with the temporal property [14]. They used the membership-function tuning mechanism to transform item quantities into fuzzy sets. Wu et al. introduced the LEFT2 algorithm to find fuzzy frequent itemsets [15]. In recent years, some FUIM approaches [16,17] were proposed to combine transformed fuzzy terms with UIM. They considered the fuzzy-set theory, quantitative information, and utility factor to find the itemsets with high fuzzy utility values. For example, Huang et al. introduced the TP-TFU method to solve it [16]. They also suggested a two-phase approach. In the first phase, based on their designed upper-bound model, candidate itemsets are found in level-wise processing. In the second phase, the method scans the database to ensure whether the candidates are truly high.

In order to solve the TFUIM (temporal fuzzy utility itemset mining) problem, a two-phase algorithm based on a tree structure was previously proposed to improve the performance efficiency in mining [18]. Its main goal is to reduce the number of candidates compared with that in [16]. The first phase is to hold candidate itemsets with the upper bounds satisfying the threshold in the tree. With traversing the tree, candidate itemsets can be found. They then need to be decided for being high itemsets in the second phase. However, generating the possible candidates in the first phase in [18] is also time-consuming. This study thus proposes a one-phase algorithm based on a tree structure with the array list to overcome the execution shortcoming without scanning the database in the second phase. The proposed tree structure holds all the required information. It needs additional memory to store mining information but can run faster than the previous one. The proposed mining approach based on the designed tree structure can find desired high itemsets without scanning the database again.

The rest of the paper is organized as follows. Section 2 presents relevant works related to the proposed approach; Section 3 explains the TFUIM problem and some terms; Section 4 describes the proposed approach and uses an example to illustrate the approach; Section 5 shows some experimental comparison on three real-life datasets; Lastly, a conclusion is provided in Section 6.

2. Related Works

FIM from a database is fundamental but important in knowledge discovery. Many approaches are proposed for FIM, and among them, the Apriori algorithm is a classical level-by-level processing method [1–3]. In Apriori, multiple database scans are used to find frequent itemsets and association rules; therefore, it spends a large amount of execution time in mining frequent itemsets. In order to overcome the problem, Han et al. presented the FP-tree (frequent pattern tree) strategy to store mining information in each node [19]. This strategy finds frequent itemsets by recursively building conditional FP-trees and traversing them. It only needs two database scans to find desired itemsets, much less than that required in Apriori.

Unlike FIM, the UIM problem [7] was proposed to avoid considering only frequencies in FIM. UIM is more complicated than FIM. It takes the following properties for each item: the amounts appearing in transactions and the unit profit. It defines the utility measure from the two property values and uses it to judge whether an item or itemset is useful based on a given utility threshold constraint. A high utility itemset thus means such a useful itemset. Several studies were developed to solve the UIM problem. For example, Dawar et al. applied a data structure to store mining information and then designed a tree-based method to reduce any candidates generated in mining [20]. Nawaz et al. used the concept of genetic algorithms to determine most of the high utility itemsets by using crossover operations [21] within a limited time. Besides, Singh et al. used transaction merging and dataset projection to find top-k utility itemsets [22].

However, UIM in the mining procedure does not have the anti-monotonic property. In other words, the utility value of an itemset is not certainly less than those of its subsets. Therefore, the search space in UIM is large, which results in expensive searching costs. In order to reduce the search space in UIM, the transaction weighted utilization (TWU) was proposed [9] as an upper-bound measure, and it was also adopted in some other methods [23,24]. The itemsets using the measure of TWU possess the anti-monotonic property, so Liu et al. designed a two-phase algorithm based on the measure [9]. In the first phase, this algorithm obtains the candidates with TWU values not less than the utility threshold. The algorithm then scans the database again in the second phase to find the actual utility values of the candidates and judge whether they satisfy the threshold constraint. However, this method consumes much execution time due to the level-wise processing. Some approaches combining the TWU and the FP-tree concepts were then proposed to improve the execution efficiency of UIM [25,26].

Fuzzy sets [13] were used to handle linguistic and uncertain situations in many domains, such as technology, economy, and business. They are easily comprehended and consistent with human perception and sense. Fuzzy sets can be thought of as an extension of crisp sets. When using fuzzy sets for FIM, item quantities in a quantitative dataset are first transformed into linguistic terms with membership values, and then the linguistic terms are processed by modifying the traditional mining procedures. Some quantitative FIM approaches were proposed. For example, Srikant and Agrawal introduced a method to derive quantitative association rules by dividing the quantitative values of items into some intervals [27]. Chen et al. considered fuzzy multi-level association-rule mining [28] and used the cumulative probability distribution to generate the number of intervals and build membership functions of all the items. Besides, practical applications about transformed linguistic terms from quantitative information were proposed. For example, Dhanaseelan and Jeyasutha improved and expanded the fuzzy mining approach to analyze and extract critical factors affecting breast cancer from the quantitative database for health information [29]. To carry out association broadcasting, realize visualization, and determine valuable information for a human resource management system, Wang and Wang used some techniques about fuzzy data mining to implement data partition [30]. They used an incremental mining approach to handle newly inserted data and a clustering approach to cluster larger amounts of data, respectively. Furthermore, Yavari et al. applied the patient profile (including age, gender, and medical condition) to generate fuzzy partitions and then

presented a fuzzy pattern mining approach with three phases to find the combinations between risk factors and diseases [31].

The mining results derived from the traditional UIM [8,9] lack quantitative information because they only find high utility itemsets without containing item amount information. The quantitative relationships of the items are not mined and displayed. As mentioned above, fuzzy FIM could find the semantic meaning for item quantity relationships. Thus, Lan et al. solved the fuzzy utility mining (FUM) problem to fit real applications [17]. They considered the item quantities, item unit profits, and semantic meaning to mine high fuzzy utility itemsets. The same as the linguistic conversion step in fuzzy FIM, their approach first converts the quantitative values into fuzzy terms and then mines high fuzzy utility linguistic patterns. Since UIM does not keep the downward-closure property, so does FUM. Therefore, they designed an upper-bound measure to hold the property in FUM. Hong et al. then adopted tree structures to mine fuzzy utility itemsets [32]. After that, Wan et al. also used a fuzzy list structure embedded in trees to improve the efficiency of Lan et al.'s approach [33]. Since the mining process in FUM consumed too much runtime in the mining process, Yang et al. proposed an evolutionary algorithm to reduce the computational cost [34]. Besides, to consider the on-shelf time of an item, Huang et al. presented a research issue to consider the time factor for an item in FUM [16]. They introduced the designed upper-bounds for reducing the search space in mining and presented the TP-TFU (two-phased temporal fuzzy utility mining) method to find high fuzzy temporal-based itemsets. Hong et al. then designed a two-phase tree-structure approach to improve the efficiency of TP-TFU [18]. In this paper, a one-phase approach will be proposed.

3. Problem Definition

Assume TQD denote a temporal quantitative transaction database. It may have: $TQD = \{T_1, T_2, \ldots, T_z\}$, where z is the number of transactions. Let I be the set of items in TQD. Thus, $I = \{i_1, i_2, i_3, \ldots, i_m\}$, where m is the number of items. Each transaction T_y in TQD has its identifier denoted TID. Each item i_m in a transaction T_y includes its quantity sold, denoted v_{ym}, and unit profit called the external utility and denoted $s(i_m)$. There is a set of time periods in TQD, denoted $P = \{p_1, p_2, p_3, \ldots, p_r\}$, where r is the number of time periods. Table 1 shows a database with five different items and three time periods. The unit profit values of the items are listed in Table 2.

Table 1. An example of temporal quantitative transaction database.

Period	*TID*	*Quantitative Items*	*Transformed Fuzzy Terms*
P_1	T_1	{a:6}, {c:2}	$(1/a.M)$, $(0.67/c.L)$
	T_2	{a:4}, {b:4}	$(0.67/b.L + 0.33/b.M)$
P_2	T_3	{a:1}, {c:1}	$(0.33/c.L)$, $(0.33/a.L)$
	T_4	{a:3}, {b:3}, {c:4}, {d:4}, {e:2}	$(1/a.L)$, $(1/b.L + 0.67/e.L)$
P_3	T_5	{a:3}, {b:6}, {e:2}	$(1/a.L)$, $(1/b.M.)$, $(0.67/e.L.)$
	T_6	{a:3}, {d:4}, {e:2}	$(1/a.L + 0.67/e.L)$

Table 2. Unit Profit values of Items.

Item	*Profit*
a	2
b	6
c	4
d	2
e	4

Given the temporal fuzzy utility threshold λ and the membership functions of each item, such as those in Figure 1, the problem is to find all the high temporal fuzzy utility

itemsets from the temporal database. The following definitions [16] are first given to explain the mining problem.

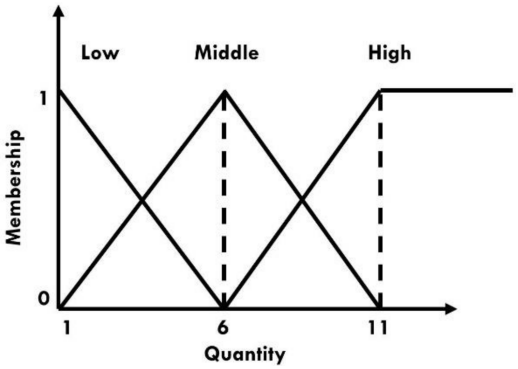

Figure 1. Membership functions of 3-linguistic terms.

Definition 1. *Let the h-th linguistic term of the item i_m be denoted R_{mh}. The fuzzy utility fu_{ymh} of the linguistic term R_{mh} in a transaction T_y is defined as:*

$$fu_{ymh} = *v_{ym} * s(i_m) \tag{1}$$

where v_{ym} is the quantitative value of the item i_m in the transaction T_y, u_{ymh} is the fuzzy value of v_{ym} in R_{mh}, and $s(i_m)$ is the unit profit value of i_m.

Assume Figure 1 shows the three membership functions for item a. Its quantity in T_2 is 4, which is transformed into $(0.67/a.L + 0.33/a.M)$. The fuzzy utility of $R_{a.L}$ is thus $0.67 \times 4 \times 2$, which is 5.36.

Definition 2. *The transactional fuzzy utility tfu_y of a transaction T_y is the summation of the fuzzy utility values of all linguistic terms in T_y.*

Definition 3. *A fuzzy itemset includes one or more linguistic terms with no terms derived from the same item.*

Take transaction T_6 as an example. $(a.L, d.L)$ is a valid fuzzy itemset with two fuzzy items, but $(d.L, d.M)$ is not because it is derived from the same item d.

Definition 4. *Let X be a fuzzy itemset. The fuzzy utility of X in T_y, denoted fu_{yX}, is:*

$$= * \sum_{R_{mh} \subseteq X} v_{ym} * s(i_m) \tag{2}$$

where the μ_{yX} is the minimum scalar cardinality in R_{mh} in X.

Take the fuzzy itemset $(a.L, d.L)$ in T_6 as an example. The fuzzy value of $(a.L, d.L)$ in T_6 is 0.67 (=min(1, 0.67)). Thus, its $fu_{6,\,(a.L,\,d.L)}$ value is calculated as $0.67 \times [(3 \times 2) + (4 \times 2)] = 9.38$.

Definition 5. *The start transaction period (STP) of an item is its first appearing time period in TQD.*

Definition 6. *The lasting transaction periods of an itemset X, denoted as LTP_X, is the set of the periods from the latest STP among the items in X to the end of TQD.*

Definition 7. *The maximal fuzzy utility mfu_{ym} of an item i_m in the transaction T_y is the maximum fuzzy utility value of all its linguistic terms.*

Definition 8. *The maximal transactional fuzzy utility $mtfu_y$ of a transaction T_y is the summation of the maximal fuzzy utility values of all the items in T_y.*

Definition 9. *The start transaction period STP_{all} of all the items in TQD is the latest time period of all the items.*

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

$$tfuubr_X = \sum_{X \in T_y \cap T_y \in LTP_X} mtfu_{yX} / \sum_{T_y \in LTP_{all}} tfu_y. \quad (3)$$

Definition 11. *A fuzzy itemset X is a high temporal fuzzy utility upper-bound itemset (HTFUUBI) iff $tfuubr_X \geq \lambda$.*

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

$$tfur_X = \sum_{X \in Trans_y \cap Trans_y \in LTP_X} fu_{yX} / \sum_{Trans_y \in LTP_X} tfu_y \quad (4)$$

Definition 13. *A fuzzy itemset X is called a high temporal fuzzy utility itemset (HTFUI) iff $tfur_X \geq \lambda$.*

4. The Proposed Algorithm

This section presents the proposed array-embedded tree algorithm for *TFUIM*, called the *ATTFUM* algorithm. In the past, a two-phase algorithm [18] was introduced to solve the temporal fuzzy utility mining problem. This study uses a one-phase method to find *HTFUIs* according to the upper bound model [16] and FP-tree [19]. First, the data-processing step needs to be processed. The item quantities in all the transactions are transformed into fuzzy sets based on the membership functions. If the *tfuubr* value of a linguistic term is not less than the specified threshold, it is treated as an *HTFUUBI*. Then, the linguistic terms which are not *HTFUUBIs* are deleted from the transformed database. The remaining terms are then sorted in descending counts. The sorted terms in each transaction are used to build the tree. Finally, a mining procedure similar to the FP-Growth in FIM is designed to find out potential candidates and determine whether they are *HTFUIs*. The details for the steps of the proposed method with examples are described in the following four subsections.

4.1. Transformed into a Fuzzy Database

In order to execute the *ATTFUM* method for mining *HTFUIs*, the pre-processing step is first executed. First, each transaction *TID* in a database is converted according to its sale time to a specific time period. The start time period of each item in the given database is then recorded. Next, the quantities of all the items in a database are fuzzified into linguistic terms based on the defined membership functions. In the transformation process, the fuzzy value of each transformed term can be obtained as well. For the *TQD* in Table 1, the start time periods (*STPs*) of all the items are shown in Table 3.

Table 3. The STP of all items for the running example.

Item	***a***	***b***	***c***	***d***	***e***
STP	P_1	P_1	P_1	P_2	P_2

Assume the five items in *TQD* use the same membership functions in Figure 1. The final fuzzified results are listed in the last column of Table 1. Then the upper-bound measure is utilized to generate the downward-closure property. The *mtfu* value of each transaction needs to be calculated. Thus, to calculate the upper-bound value of a transaction, the *fu* value of each linguistic term in the transformed database (the last column of Table 1) needs to be obtained. For example, for the linguistic term (*a.L*) with the value 0.66 in T_2, its $fu_{2,a.L}$ is 5.36 (=0.67 × 4 × 2). The *fu* values of the other linguistic terms in T_2 are $fu_{2,a.M}$ (2.64), $fu_{2,b.L}$ (16.08), and $fu_{2,a.M}$ (7.92), respectively. Thus, the *mtfu* and *tfu* values of T_2 are

21.44 (=5.36 + 16.08) and 32 (=5.36 + 2.64 + 16.08 + 7.92). The *mtfu* and *tfu* values for all the transactions are listed in Table 4.

Table 4. The *mtfu* and *tfu* values for each transaction.

	TID					
	T_1	T_2	T_3	T_4	T_5	T_6
tfu	17.36	32	1.98	53.34	47.36	26
mtfu	17.36	21.44	1.98	45.42	47.36	23.36

Next, the *tfuubr* value of each linguistic term is calculated. Take the linguistic term (*a.L*) in the last column of Table 1 as an example. It can be observed that the term (*a.L*) happens in T_2, T_3, T_4, T_5, and T_6, and their *mtfu* values are shown in Table 4. Besides, the STP_{all} derived from Table 3 is P_2. Assume the defined threshold is 40%. According to definition 10, the *tfuubr* value of the linguistic term (*a.L*) is first handled. The numerator is 139.56 (=21.44 + 1.98 + 45.42 + 47.36 + 23.36). Then, the denominator is calculated as 128.68 (=1.98 + 53.34 + 47.36 + 26), which depicts the *tfu* value from STP_{all} (P_2) to LTP_{all} (P_3). The *tfuubr* value of the linguistic term (*a.L*) is then 108.4% (=139.56/128.68), which is not less than the defined threshold. Therefore, the linguistic term (*a.L*) is an *HTFUUBI*. The same process is performed for the other terms, and the sorted results based on their occurrence frequency are displayed in Table 5.

Table 5. The sorted linguistic terms satisfy the threshold constraint.

Linguistic Term	***a.L***	***c.L***	***e.L***	***b.L***	***b.M***	***d.L***	***d.M***
Total mtfu	139.56	64.76	116.14	66.86	68.8	68.78	68.78
Occurrence	5	3	3	2	2	2	2

The linguistic terms that are not *HTFUUBIs* are then deleted from the transformed database. The remaining linguistic terms in a transaction are sorted by descending counts and formed as a revised database. The resultant revised database is shown in Table 6.

Table 6. The sorted revised database.

Period	**TID**	**Transformed Fuzzy Linguistic Terms**
P_1	T_1	$(0.67/c.L)$
	T_2	$(0.67/a.L)$, $(0.67/b.L)$, $(0.33/b.M)$
P_2	T_3	$(0.33/a.L)$, $(0.33/c.L)$
	T_4	$(1/a.L)$, $(0.67/c.L)$, $(0.67/e.L)$, $(1/b.L)$, $(0.67/d.L)$, $(0.33/d.M)$
P_3	T_5	$(1/a.L)$, $(0.67/e.L)$, $(1/b.M)$
	T_6	$(1/a.L)$, $(0.67/e.L)$, $(0.67/d.L)$, $(0.33/d.M)$

The rearranged terms in each transaction are then used to be inserted into the tree sequentially. The process is described below.

4.2. Tree Structure Construction

The proposed algorithm begins building a tree structure after forming the fuzzified database. Each transaction is sequentially processed. The linguistic terms in each transaction are inserted to create nodes in the tree. Each node is attached with the *total_mtfu* field and the array-list data structure for storing mining information. The *total_mtfu* field stores the accumulated *mtfu* value. If the inserted linguistic term in a transaction belongs to this node, set $total\_mtfu = total\_mtfu + mtfu_y$. The array-list data structure includes three subfields: *TID*, fuzzy value, and utility value.

For example, the linguistic term *c.L* of transaction T_1 in Table 6 results in the first branch of the tree. The initial *total_mtfu* value of the node is the *mtfu* value of T_1, which

is 17.36 in Table 4. Then, its array-list structure contains the transaction identifier T_1, the fuzzy value of the term *c.L* (=0.67) in Table 6, and its utility value (=2 × 4 = 8) derived from Tables 1 and 2. The first inserted node is shown in Figure 2.

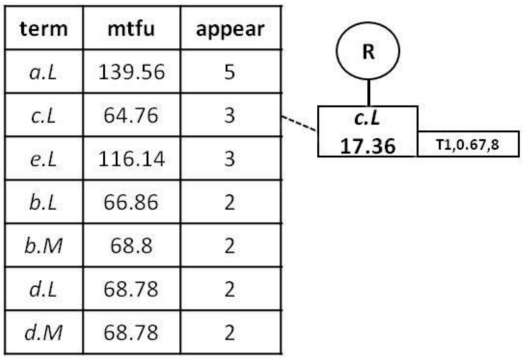

Figure 2. Inserting T_1 into the tree.

Next, the transaction T_2 is handled. It consists of three linguistic terms (*a.L*), (*b.L*), and (*b.M*). The *mtfu* value of T_2 is 21.44 in Table 4. The first linguistic term (*a.L*) is added to the tree. The node of the term (*a.L*) cannot share the same prefix (*c.L*) of the tree with the transaction T_1. Therefore, the second branch is grown to connect the node (*a.L*) and the root. The term (*b.L*) is then added to the tree as the child of the (*a.L*) node. The term (*b.M*) is handled in the same way. Thus, in this branch, there are three nodes inserted, all with the *mtfu* (21.44) value of T_2 attached. After T_2 is processed, the tree is shown in Figure 3. The complete tree for all the transactions is depicted in Figure 4.

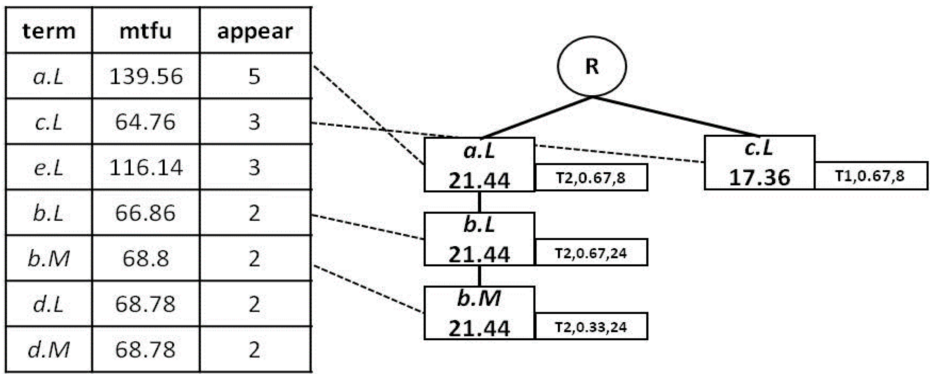

Figure 3. Inserting T_2 into the tree.

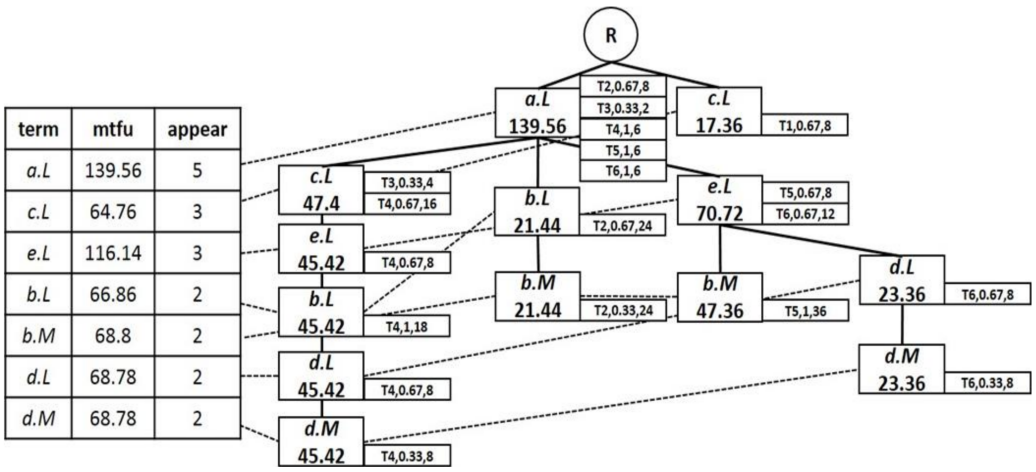

Figure 4. The complete tree.

4.3. The Tree-Constructing Algorithm

The details of the tree-construction method are given below.

The proposed Algorithm 1:

Algorithm 1: tree construction.

INPUT: *TQD*, a temporal quantitative database;
m, number of items in *TQD*;
μ, membership functions;
r, number of time periods;
λ, minimum temporal fuzzy utility threshold.

OUTPUT: a constructed tree.

STEP 1: According to the r time periods for *TQD*, find the corresponding time period of each transaction.

STEP 2: For each item i_m in *TQD*, find its STP_{im} and then derive STP_{all}.

STEP 3: For each transaction T_y in *TQD*, transform the quantity sold v_{ym} of each item i_m into linguistic items R_{m1} to R_{mh} in a fuzzy set f_{ym}, with u_{ym1} to u_{ymh} being their scalar cardinality values in f_{ym} based on the membership functions μ . Let *RTQD* denote the revised *TQD*.

STEP 4: For each transaction T_y in a period p_j in *RTQD*, conduct the following substeps:

Step 4.1: Count the fu_{ymh} value of the h-th linguistic term R_{mh} in T_y.

Step 4.2: Find the mfu_{ym} value of i_m in T_y.

Step 4.3: Calculate the tfu_y and $mtfu_y$ values of each T_y.

STEP 5: Calculate LTP_{all}.

STEP 6: Build the upper-bound table as empty, with each tuple having the three fields: linguistic term, total *mtfu*, and count.

STEP 7: For each linguistic term R_{mh} in *RTQD*, calculate its *tfuubr* value. If the value is not less than λ, set it as an $HTFUUBI_1$.

STEP 8: All the $HTFUUBI_1$'s in *RTQD* are inserted into the upper-bound table, with their linguistic term, total *mtfu* value, and count.

STEP 9: For each linguistic term R_{mh} in *RTQD*, if it has not existed in the upper-bound table, remove it.

STEP 10: Resort the linguistic terms in the upper-bound table based on their descending counts. The upper-bound table acts as the Header_Table.

STEP 11: Insert each transaction T_y in *RTQD* into the tree. If a linguistic term x in T_y has not been at the corresponding branch in the tree, add the term x to the end of the branch as the node x, and put the *mtfu* value of T_y and the array-list structure data (including *TID* of T_y, the fuzzy value of x and utility value of x) into the node. If a linguistic term x in T_y has been at the corresponding branch in the tree, just add the *mtfu* value of T_y to the *total_mtfu* field of the node x and put the related data of the term x in the transaction T_y into the end of the array-list structure. Until no transaction needs to be processed, the tree is constructed.

4.4. Mining HTFUIs from the Tree

When the tree is built, the candidate *HTFUI* itemsets can then be mined using a procedure similar to the FP-Growth approach [19]. The procedure is, however, more complicated than the FP-Growth due to the complex data kept in the nodes. It will check whether a candidate has its *tfur* value satisfying the defined threshold.

For a developed tree such as that in Figure 4, the mining algorithm finds *HTFUIs* as follows. The Header_Table includes the sorted linguistic terms according to the order of descending counts. A conditional pattern tree is grown for each linguistic term in the Header_Table. These terms are processed bottom-up and one by one. The *mtfu* values of the generated candidate itemsets, including the processed linguistic term, are then recursively calculated. If their *mtfu* values are not less than the defined threshold, they are removed in the conditional pattern tree. In Section 4.3, each branch in the tree is built using the *mtfu* values of the linguistic terms in a transaction. Thus, the candidate fuzzy k-itemsets obtained

by the *total_mtfu* field in the node can be found easily directly from the tree. If the *tfur* values of the candidate fuzzy *k*-itemsets are larger than or equal to the threshold by using the array-list data in the node, the *k*-itemsets are *HTFUIs*. Below is the mining algorithm.

The mining Algorithm 2:

Algorithm 2: mining HTFUIs from the tree.

INPUT: r, the constructed tree;
λ, minimum temporal fuzzy utility threshold.
OUTPUT: All HTFUI itemsets.
For each linguistic term x bottom-up in the Header_Table, conduct the following steps:

Step 1: From the constructed tree, form a conditional pattern tree of the node x by the link of x in the Header_Table.
Step 2: Check each prefix node in the conditional pattern tree from the node x for whether the prefix node and x are originated from the same item. If yes, remove the prefix node.
Step 3: Check the *total_mtfu* value for each prefix node in the conditional pattern tree from x for whether the value is not less than $\sum_{T_y \in LTP_{all}} tfu_y * \lambda$. If no, remove the prefix node.
Step 4: Find candidate *HTFUIs* from the remaining conditional pattern tree from x.
Step 5: Calculate the $tfur_z$ value from the node of each candidate z with the array-list structure using the intersection operation as the fuzzy operator. If $tfur_z$ is not less than λ, z is a *HTFUI*.

For example, take the linguistic term (*d.M*) in the Header_Table in Figure 4 as an example. Its conditional pattern tree is shown in Figure 5a. There are two branches derived from the linguistic term (*d.M*), as shown in Figure 5b.

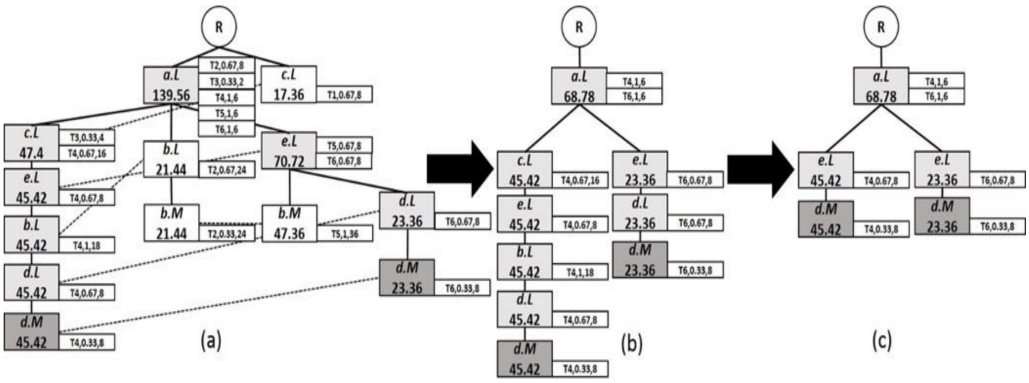

Figure 5. The conditional pattern tree for the linguistic term (*d.M*). (**a**) The whole frequent pattern tree; (**b**) The two branches for (*d.M*); (**c**) The final conditional pattern tree for (*d.M*).

Along with building the conditional pattern tree for the linguistic term (*d.M*), the combinations of the possible fuzzy candidate itemsets with (*d.M*) are then derived using the mining algorithm recursively. All the possible fuzzy itemsets are processed for checking whether the *mtfu* values of the candidates are not less than the threshold in the temporal fuzzy utility mining. The linguistic terms (*d.L*) and (*d.M*) are originated from the same item *d*, so the linguistic terms (*d.L*) in two branches of the conditional pattern tree are removed in Figure 5b. All the linguistic terms in Figure 5b are then processed for checking whether their *mtfu* values meet the threshold value. The total *mtfu* value of each fuzzy itemset, including (*d.M*), is then counted from individual branches. In this case, the fuzzy itemset (*e.L*, *d.M*) exist in two branches of Figure 5b. Thus, their *mtfu* value is 68.78 (=45.42 + 23.36). It is larger than the threshold value (=(1.98 + 53.34 + 47.36 + 26) × 40% = 51.472). As another example, the *mtfu* values of the linguistic terms (*c.L*) and (*b.L*), which are both 45.42 in the left branch, are less than the threshold value, so they are omitted in the two branches. The final result for the conditional pattern tree with the linguistic term (*d.M*) is shown in Figure 5c.

All the fuzzy itemsets derived from the conditional pattern tree can be obtained without rescanning the database. In this example, all the possible itemsets with *d.M* are [(*d.M*, *e.L*): 68.78], [(*d.M*, *a.L*): 68.78] and [(*d.M*, *e.L*, *a.L*): 68.78] derived from Figure 5c. Next, all the fuzzy itemsets are processed to check whether their *tfur* values satisfy the threshold. If yes, they are *HTFUIs*. Take [(*d.M*, *e.L*)] as an example. According to definition 12, the *tfur* value of the fuzzy itemsets [(*d.M*, *e.L*)] needs to be calculated. It can be known that the $LTP_{[(d.M,\ e.L)]}$ range is from P_2 to P_4. Therefore, the *tfu* values of all the transactions in $LTP_{[(d.M,\ e.L)]}$ are summed, and its value is 128.68 (=1.98 + 53.34 + 47.36 + 26), which is the denominator. Next, the fuzzy utility value of [(*d.M*, *e.L*)] is calculated. In Figure 5c, the array-list structure information of the nodes (*d.M*) and (*e.L*), including identifier transaction, fuzzy value, and utility value, can be obtained. Thus, the fuzzy values of (*d.M*) and (*e.L*) are both 0.33 and 0.67 from their array-list structure in T_4 and T_6. By using the intersection operation in the fuzzy sets, the fuzzy values of the fuzzy itemset [(*d.M*, *e.L*)] in T_4 and T_6 are 0.33 and 0.33, respectively. The fuzzy utility value of [(*d.M*, *e.L*)] is calculated as $0.33 \times (8 + 8) + 0.33 \times (8 + 8) = 11.88$, and its *tfur* value is 9.23% (=11.88/128.68), which is less than the defined threshold (40%). It is not an *HTFUI*. The remaining itemsets are handled in the same way.

5. Performance Evaluation

Three real datasets, mushroom [35], retail [35], and foodmart [36], were used to compare the proposed *ATTFUM* and the previous *FHTFUP* [18] algorithms. These two methods were run on a desktop computer with a Dual-Core Processor of 3.3 GHz and 16 GB of RAM. The programming language is JAVA. The quantitative value for each item in mushroom, retail, and foodmart datasets was assigned from 1 to 10 randomly. The profits for items were randomly set within 1 to 1000. In the experiments, the three datasets were transformed into three linguistic terms according to the pre-defined membership functions in Figure 1. The number of time periods is assigned 2.

The running time and the memory consumption for the two methods in different thresholds were evaluated. The results are shown in Figures 6 and 7.

Figure 6 shows the comparison results on the three different datasets under varying thresholds. Obviously, the proposed *ATTFUM* method has better performance in terms of computation time when compared to the *FHTFUP* method. The two algorithms derive the same *HTFUIs*, but *ATTFUM* needs less running time compared to the other one. Along with the threshold set at a lower value, *ATTFUM* has better performance than *FHTFUP*. This is because the *HTFUIs* derived by the *ATTFUM* method is processed by each node with the array-list structure information, but those derived from the *FHTFUP* method are mined by performing database rescans. For the three different datasets, when the threshold decreased, the proposed approach ran slower.

Figure 7 shows the memory consumption for the two approaches on three different datasets under varying thresholds. Along with the threshold decreasing in the three different datasets, the memory space used for *ATTFUM* is more than that for *FHTFUP*. This main reason is that each node in the proposed algorithm needs to keep the array-list data for mining information of the same linguistic term in different transactions. Besides, it can be observed that the proposed method required more memory consumption when the threshold decreased.

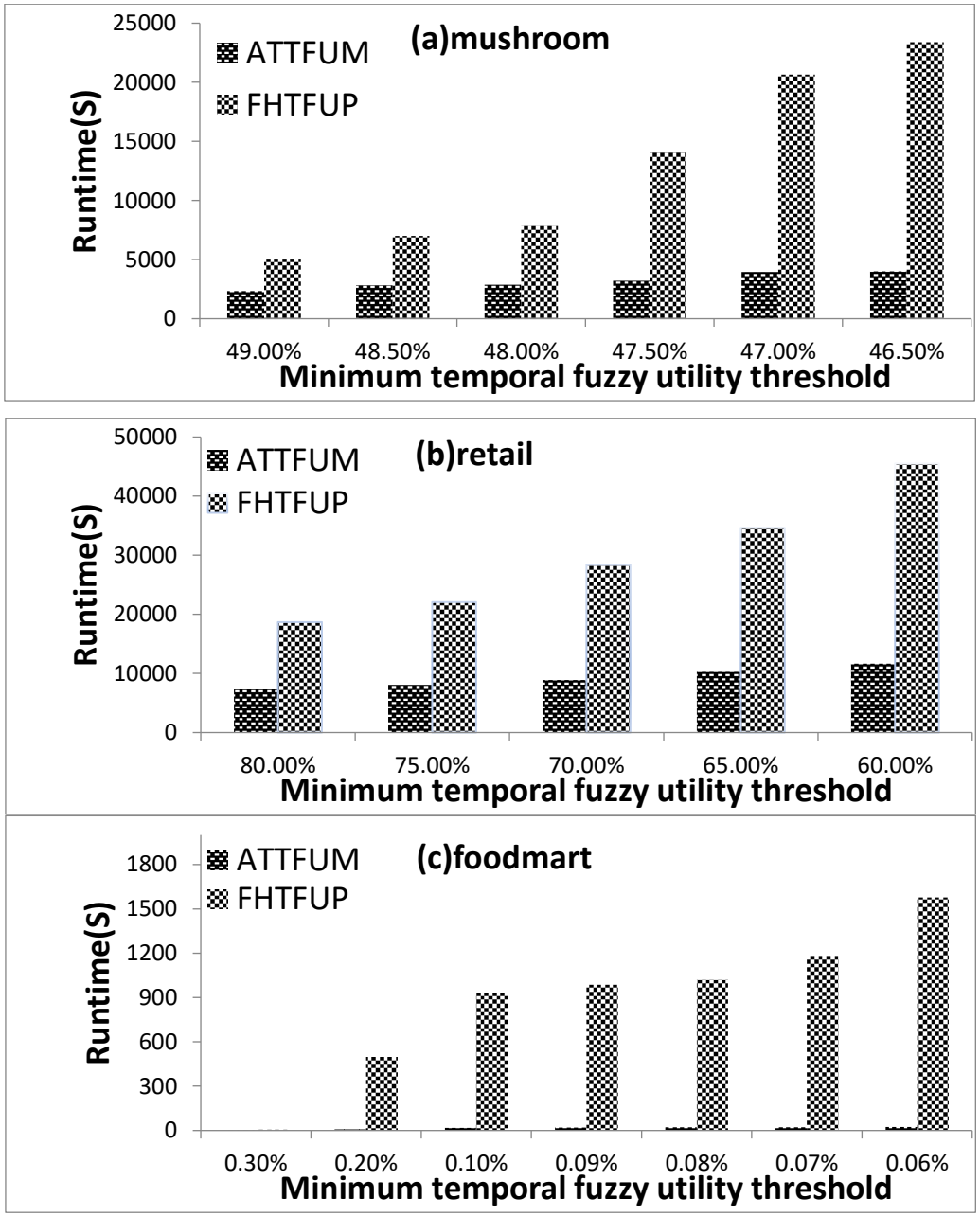

Figure 6. The comparison of running time for the two methods with different thresholds.

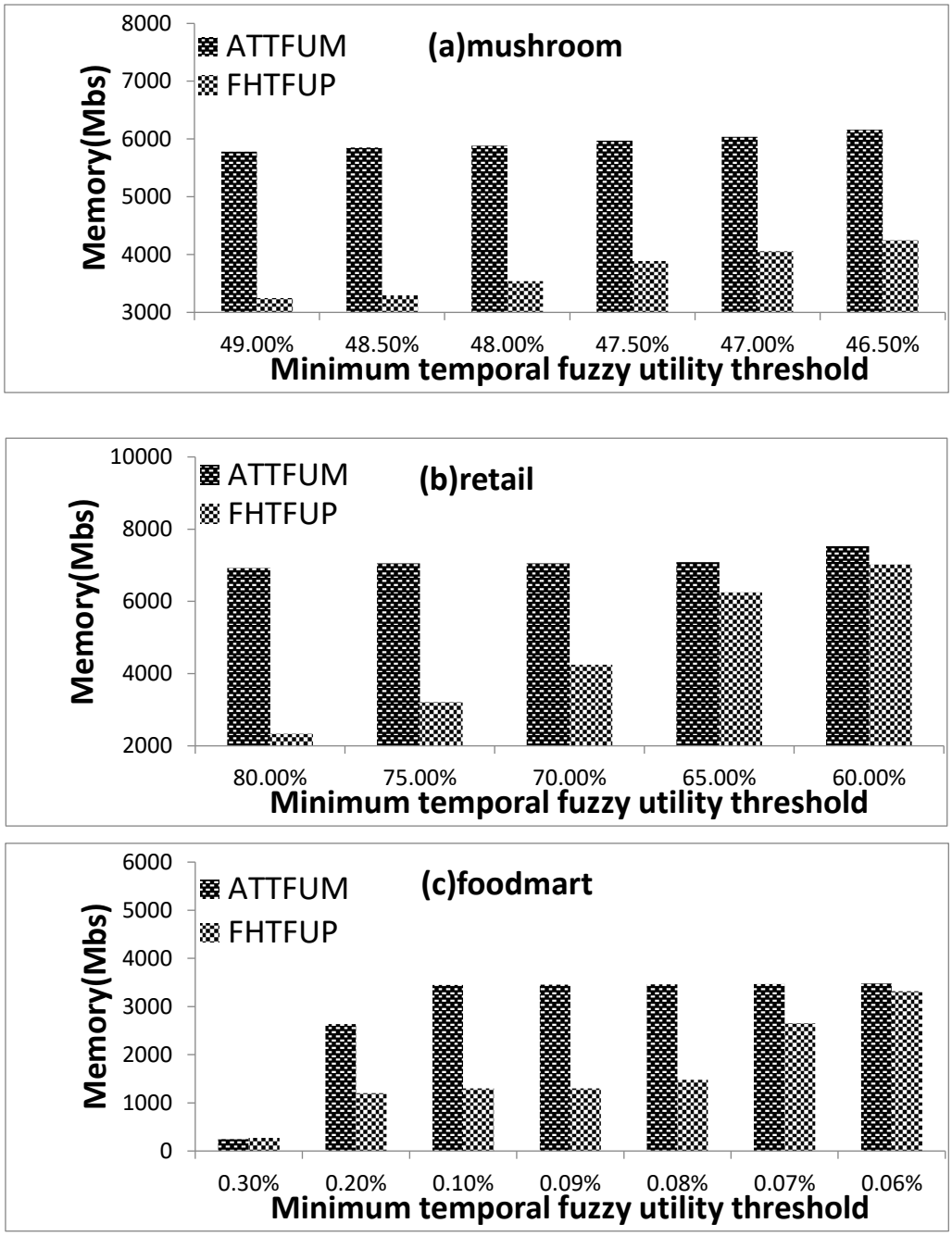

Figure 7. The comparisons of memory consumption for two methods in distinct thresholds.

In summary, the proposed *ATTFUM* method with each node, including the array-list information, requires less running time when compared to the two-phase *FHTFUP* method. Although the proposed method uses the array list to store mining information in each node, it does not need to rescan the database to determine *HTFUIs* when compared with the *FHTFUP* method. Thus, the computation time is significantly reduced.

6. Conclusions

In this paper, to solve the problem of mining *HTFUIs* in determining temporal knowledge, the *ATTFUM* algorithm is proposed. The proposed method is used to handle linguistic terms in each transaction for a temporal database using the concepts of the FP-tree and the upper-bound model for downward closure property. Each node in the proposed tree has the upper-bound value and the array-list information, keeping the information in

different transactions. Therefore, the upper-bound values are used to derive the potential candidates from the conditional pattern trees in the mining process. Along with each node in a conditional pattern tree, the *HTFUIs* can be completely obtained using the array-list information in nodes.

Compared with the non-temporal fuzzy utility mining, the proposed approach can provide a more flexible consideration to real applications. When the number of time periods is one, the temporal fuzzy utility mining degenerates to the fuzzy utility mining. Thus, the former can be thought of as a generalization of the latter. Besides, compared to the previous temporal fuzzy utility mining approach, the experimental results on the three real databases showed that the proposed ATTFUM algorithm is superior to the FHTFUP algorithm in temporal fuzzy utility mining problems in terms of the execution time. Furthermore, by adopting the array-list structure to store mining information in each node, the proposed *ATTFUM* can efficiently mine *HTFUIs*, although it uses more memory usage than *FHTFUP*. The proposed *ATTFUM* can reduce the execution time due to the avoidance of the database rescan in the second phase when compared with the previous *FHTFUP* approach.

For further research issues, different tree structures and other data structures could be designed to improve the performance of the *ATTFUM* algorithm. Besides, the scenario of inserting new transactions into a database may be considered, and different maintenance problems of the TFUIM may be handled. How to process different exhibition time periods of items in fuzzy mining is also an interesting challenge. Some variants, such as mining with multiple minimum thresholds, can also be designed based on the proposed architecture.

Author Contributions: Conceptualization, T.-P.H. and C.-Y.L.; methodology, T.-P.H. and C.-Y.L.; software, C.-Y.L. and W.-M.H.; validation, T.-P.H. and S.-L.W.; formal analysis, T.-P.H. and J.C.-W.L.; data curation, W.-M.H. and S.-L.W.; writing—original draft preparation, T.-P.H., C.-Y.L. and W.-M.H.; writing—review and editing, S.-M.L. and J.C.-W.L.; supervision, S.-M.L.; funding acquisition, T.-P.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Ministry of Science and Technology of the Republic of China under Grant MOST 109-2221-E-390-013-MY2.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data is available at http://fimi.cs.helsinki.fi/data/ (accessed on 28 February 2022).

Conflicts of Interest: This is a modified and expanded version of the paper "One-phase temporal fuzzy utility mining [37]," presented at the 2020 IEEE International Conference Fuzzy Systems (FUZZ-IEEE), 2020, Glasgow, UK. We also claim that this present study has not been published or accepted for publication in other journals.

References

1. Agrawal, R.; Srikant, R. Fast algorithm for mining association rules. In Proceedings of the Twentieth International Conference on Very Large Data Bases, Miami Beach, FL, USA, 12–15 September 1994; Volume 1215, pp. 487–499.
2. Agrawal, R.; Imieliński, T.; Swami, A. Mining association rules between sets of items in large databases. In Proceedings of the ACM SIGMOD International Conference on Management of Data, New York, NY, USA, 1 June 1993; Volume 22, pp. 207–216. [CrossRef]
3. Agrawal, R.; Imielinski, T.; Swami, A. Database mining: A performance perspective. *IEEE Trans. Knowl. Data Eng.* **1993**, *5*, 914–925. [CrossRef]
4. Guney, S.; Peker, S.; Turhan, C. A Combined Approach for Customer Profiling in Video on Demand Services Using Clustering and Association Rule Mining. *IEEE Access* **2020**, *8*, 84326–84335. [CrossRef]
5. Wang, C.-H. Association rule mining and cognitive pairwise rating based portfolio analysis for product family design. *J. Intell. Manuf.* **2017**, *30*, 1911–1922. [CrossRef]
6. Ding, Z.; Liao, X.; Su, F.; Fu, D. Mining Coastal Land Use Sequential Pattern and Its Land Use Associations Based on Association Rule Mining. *Remote Sens.* **2017**, *9*, 116. [CrossRef]
7. Yao, H.; Hamilton, H.J.; Butz, C.J. A Foundational Approach to Mining Itemset Utilities from Databases. In Proceedings of the 2004 SIAM International Conference on Data Mining, Lake Buena Vista, FL, USA, 22–24 April 2004; pp. 482–486.

8. Liu, Y.; Liao, W.-K.; Choudhary, A. A fast high utility itemsets mining algorithm. In Proceedings of the 1st International WORKSHOP on Utility-Based Data Mining, Chicago, IL, USA, 21 August 2005; pp. 90–99.
9. Liu, Y.; Liao, W.-K.; Choudhary, A. A Two-Phase Algorithm for Fast Discovery of High Utility Itemsets. In Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, Hanoi, Vietnam, 18–20 May 2005; Volume 3518, pp. 689–695.
10. Lee, C.-H.; Lin, C.-R.; Chen, M.-S. On mining general temporal association rules in a publication database. In Proceedings of the 2001 IEEE International Conference on Data Mining, Maebashi City, Japan, 29 November–2 December 2001; pp. 337–344.
11. Chang, C.-Y.; Chen, M.-S.; Lee, C.-H. Mining general temporal association rules for items with different exhibition periods. In Proceedings of the 2002 IEEE International Conference on Data Mining, Maebashi City, Japan, 9–12 December 2002; pp. 59–66.
12. Weng, C.-H. Identifying association rules of specific later-marketed products. *Appl. Soft Comput.* **2016**, *38*, 518–529. [CrossRef]
13. Zadeh, L.A. Fuzzy Sets. In *Information and Control*; Elsevier: Amsterdam, The Netherlands, 1965; Volume 8, pp. 338–353.
14. Chen, C.-H.; Chou, H.; Hong, T.-P.; Nojima, Y. Cluster-Based Membership Function Acquisition Approaches for Mining Fuzzy Temporal Association Rules. *IEEE Access* **2020**, *8*, 123996–124006. [CrossRef]
15. Wu, T.-Y.; Lin, J.C.-W.; Yun, U.; Chen, C.-H.; Srivastava, G.; Lv, X. An efficient algorithm for fuzzy frequent itemset mining. *J. Intell. Fuzzy Syst.* **2020**, *38*, 5787–5797. [CrossRef]
16. Huang, W.-M.; Hong, T.-P.; Lan, G.-C.; Chiang, M.-C.; Lin, J.C.-W. Temporal-Based Fuzzy Utility Mining. *IEEE Access* **2017**, *5*, 26639–26652. [CrossRef]
17. Lan, G.-C.; Hong, T.-P.; Lin, Y.-H.; Wang, S.-L. Fuzzy utility mining with upper-bound measure. *Appl. Soft Comput.* **2015**, *30*, 767–777. [CrossRef]
18. Hong, T.-P.; Lin, C.-Y.; Huang, W.-M.; Li, K.S.-M.; Wang, L.S.-L.; Lin, J.C.-W. Using Tree Structure to Mine High Temporal Fuzzy Utility Itemsets. *IEEE Access* **2020**, *8*, 153692–153706. [CrossRef]
19. Han, J.; Pei, J.; Yin, Y. Mining frequent patterns without candidate generation. In Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data—SIGMOD '00, Dallas, TX, USA, 15–18 May 2000; Volume 29, pp. 1–12.
20. Dawar, S.; Goyal, V.; Bera, D. A one-phase tree-based algorithm for mining high-utility itemsets from a transaction database. *arXiv* **2019**, arXiv:1911.07151.
21. Nawaz, M.S.; Fournier-Viger, P.; Song, W.; Lin, J.C.-W.; Noack, B. Investigating Crossover Operators in Genetic Algorithms for High-Utility Itemset Mining. In Proceedings of the 13th Asian Conference on Intelligent Information and Database Systems, Phuket, Thailand, 7–10 April 2021; Springer Science and Business Media LLC: Berlin/Heidelberg, Germany, 2021; pp. 16–28.
22. Singh, K.; Singh, S.S.; Kumar, A.; Biswas, B. TKEH: An efficient algorithm for mining top-k high utility itemsets. *Appl. Intell.* **2018**, *49*, 1078–1097. [CrossRef]
23. Liu, J.; Wang, K.; Fung, B.C.M. Mining High Utility Patterns in One Phase without Generating Candidates. *IEEE Trans. Knowl. Data Eng.* **2016**, *28*, 1245–1257. [CrossRef]
24. Shie, B.-E.; Tseng, V.S.; Yu, P.S. Online mining of temporal maximal utility itemsets from data streams. In Proceedings of the 2010 ACM Symposium on Applied Computing, Sierre Switzerland, 22–26 March 2010; pp. 1622–1626. [CrossRef]
25. Yildirim, I.; Celik, M. An Efficient Tree-Based Algorithm for Mining High Average-Utility Itemset. *IEEE Access* **2019**, *7*, 144245–144263. [CrossRef]
26. Tseng, V.S.; Wu, C.W.; Shie, B.E.; Yu, P.S. UP-growth: An efficient algorithm for high utility itemset mining. In Proceedings of the 16th ACM SIGKDD International Conference Knowledge Discovery and Data Mining, Washington, DC, USA, 24–28 July 2010; pp. 253–262.
27. Srikant, R.; Agrawal, R. Mining quantitative association rules in large relational tables. In Proceedings of the SIGMOD International Conference on Management of Data, Montreal, QC, Canada, 4–6 June 1996; pp. 1–12.
28. Chen, J.-S.; Chen, F.-G.; Wang, J.-Y. Enhance the Multi-level Fuzzy Association Rules Based on Cumulative Probability Distribution Approach. In Proceedings of the 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, Kyoto, Japan, 8–10 August 2012; pp. 89–94.
29. Ramesh, D.F.; Jeyasutha, M. A Novel Fuzzy Frequent Itemsets Mining Approach for the Detection of Breast Cancer. *Int. J. Inf. Retr. Res.* **2021**, *11*, 36–53. [CrossRef]
30. Wu, Y.; Wang, Z.; Wang, S. Human Resource Allocation Based on Fuzzy Data Mining Algorithm. *Complexity* **2021**, *2021*, 1–11. [CrossRef]
31. Yavari, A.; Rajabzadeh, A.; Abdali-Mohammadi, F. Profile-based assessment of diseases affective factors using fuzzy association rule mining approach: A case study in heart diseases. *J. Biomed. Inform.* **2021**, *116*, 103695. [CrossRef] [PubMed]
32. Hong, T.P.; Lin, C.Y.; Huang, W.M.; Li, S.M.; Lin, J.C.W. Applying tree structures to mine fuzzy high utility itemsets. In Proceedings of the 28th National Conference on Fuzzy Systems and Its Applications, Hsinchu, Taiwan, 4–7 November 2020; pp. 176–181.
33. Wan, S.; Gan, W.; Guo, X.; Chen, J.; Yun, U. FUIM: Fuzzy utility itemset mining. *arXiv* **2021**, arXiv:2111.00307.
34. Yang, F.; Mu, N.; Liao, X.; Lei, X. EA-HUFIM: Optimization for Fuzzy-Based High-Utility Itemsets Mining. *Int. J. Fuzzy Syst.* **2021**, *23*, 1652–1668. [CrossRef]
35. Frequent Itemsets Mining Dataset Repository. Available online: http://fimi.cs.helsinki.fi/data/ (accessed on 15 August 2018).
36. Microsoft, Example Database Foodmart of Microsoft Analysis Services. Available online: http://msdn.microsoft.com/en-us/library/aa217032(SQL.80).aspx (accessed on 15 August 2018).
37. Hong, T.-P.; Lin, C.-Y.; Huang, W.-M.; Li, S.-M.; Wang, S.-L.; Lin, J.C.-W. One-Phase Temporal Fuzzy Utility Mining. In Proceedings of the 2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Glasgow, UK, 19–24 July 2020; pp. 1–5.