Explainable Itemset Utility Maximization with Fuzzy Set

Guotao Xu[1], Jiahui Chen[1], Shicheng Wan[1], Cuiwei Peng[2], Yu Liu[1]*

[1] School of Computer Science and Technology, Guangdong University of Technology, China
[2] School of International Education, Guangdong University of Technology, China
guotaoxu00@gmail.com, csjhchen@gmail.com, scwan1998@gmail.com, vivianoop3@gmail.com, 1181576843ly@gmail.com

Abstract

Recently, fuzzy utility pattern mining has received much attention for its practicality and comprehensibility. It aims to discover high fuzzy utility itemsets (HFUIs) by considering not only utility but also linguistic factors. Among existing algorithms, experiments showed that fuzzy-list-based algorithms are effective and efficient. However, a significant disadvantage of fuzzy-list-based algorithms is that constructing and maintaining fuzzy-lists is time-consuming and memory-overhead. To address this issue, a novel algorithm named explainable **I**temset **U**tility **M**aximization with **F**uzzy **S**et (FS-IUM) is proposed in this paper. The traditional fuzzy-list structure is replaced by a better structure (i.e., fuzzy-list buffer), which speeds up the mining process and reduces memory consumption. Compared with fuzzy-list structure, the fuzzy-list buffer structure and its auxiliary structure help the algorithm locate the fuzzy-list quickly and thus reduce the runtime. Moreover, with an efficient fuzzy-list buffer construction method, the algorithm reduces the cost of candidate storage. Furthermore, with several efficient strategies, the proposed algorithm can prune numerous useless patterns in advance and thus considerably reduces the runtime usage. Finally, extensive experiments on various datasets were conducted to compare the performance of FS-IUM with some state-of-the-art algorithms. The experimental results reveal that the proposed fuzzy-list buffer-based algorithm highly outperforms the baselines in terms of runtime and memory consumption.

Keywords: Data mining, Fuzzy utility mining, Fuzzy-list buffer, Fuzzy set theory

1 Introduction

With the rapid development of data technology, how to mine valuable information from transaction databases has become a popular topic recently. Association rule mining (ARM) [1] is used to discover useful but hidden association rules, and the discovered rules are always frequent and high confidence. Association rules reflect the interdependence and relevance between distinct items and are easy to predict the occurrence of other related items. A classic sample is "Beer and Nappies" which was proposed by Walmart. Retailers noticed that married men often bought beer and nappies together after they were off duty. Thus, retailers tried to shelve beer and nappies together, and the sale volume of the product combination rose as expected. In the meanwhile, a subfield of ARM, frequent itemset mining (FIM) [2-3] has also been widely studied. FIM aims at mining frequent itemsets which are portions of association rule. However, ARM and FIM algorithms all only focus on the occurrence of itemsets but ignore other important factors (e.g., risk, unit profit, and weight) of different items. For instance, the sold volume of bread and milk is higher than that of steak and red wine, but the revenue of the second product combination is obviously larger than that of the first combination. In other words, ARM and FIM algorithms may output frequent but low profit results, which is unacceptable in some cases.

To address this issue, a new mining task called high utility itemset mining (HUIM) [4] was proposed and is based on utility from economics [5]. In this field, Shen *et al.* [6] firstly tried applying utility constraint into association rule mining. They stated that the utility consists of quantity and unit profit of item, which is neither monotone nor anti-monotone. Furthermore, due to the utility may be positive or negative, previous optimization methods in FIM cannot be directly adopted in HUIM. Then, Yao *et al.* [7] formalized the utility mining problem. However, their proposed approach was too naive to discover high utility itemsets (abbreviated as HUIs). In fact, the problem of these two mentioned works is ignoring the downward-closure property. These algorithms may face a "combinatorial explosion" of itemsets since the number of distinct items may be very large. In FIM domain, downward-closure property means the subsets of a frequent itemset must be always frequent. This can greatly help algorithms prune massive useless candidates during the mining process. Thus, transaction-weighted utilization (abbreviated *TWU*) model [8] was proposed. *TWU* of an itemset X is calculated by the summation utility of transactions which contain X. Since the occurrence times of supersets of X are always less than or equal to that of X in a database, *TWU* of these supersets must be lower than or equal to *TWU* of X too. Obviously, *TWU* is a powerful upper-bound and can be used to estimate the utility of X. If *TWU* of an itemset is lower than the user-specified minimal utility threshold (*minUtil*), we can safely prune the itemset and its supersets during the mining process. After that, there are a lot of information available in literature about HUIM algorithms (e.g., HUI-Miner [9], FHM [10], TopHUI [11], and ULB-Miner [12]).

Though HUIM algorithm assesses the importance of different items by numerical utility measure, the mined re-

*Corresponding Author: Shicheng Wan; E-mail: scwan1998@gmail.com
DOI: 10.53106/160792642023032402004

sults cannot provide more details of HUIs, such as purchase quantity of items. In some cases, users want to find "tall" trees in a forest or discover a group of "beautiful" girls. The adjectives "tall" and "beautiful" are both language terms that cannot be described directly in numerical values. Therefore, by combining with fuzzy set theory [13], Wang *et al.* [14] proposed a new framework called fuzzy utility mining (FUM) to discover high fuzzy utility itemsets (HFUIs) from quantitative transaction databases. Then, Lan *et al.* [15] adopted a user-defined membership function to assess items' fuzzy utility. The highlight of their work is implementing the downward-closure property in FUM, which is a general but effective fuzzy utility upper-bound (*FUUB*). Recently, Wan *et al.* [16] proposed a one phase FUM algorithm called FUIM. The remaining fuzzy utility is the first time adopted in FUM domain. The extensive experiments show FUIM performs better than previous algorithms. However, because of the fuzzy-list structure, FUIM costs too much runtime and memory in joining operation. We therefore infer that the performance of FUIM still can be improved. In this paper, we not only utilize some efficient pruning strategies, but also adopt buffer list structure to save runtime and memory. The novel approach is called explainable **I**temset **U**tility **M**aximization with **F**uzzy **S**et (abbreviated as FS-IUM). The following content details the major contributions of this paper.

1) A novel Fuzzy-List Buffer structure (named *FLBuf*) and a summary fuzzy-list (named *SL*) are proposed. *FLBuf* can be used repeatedly to store the key information of promising fuzzy itemsets. Accordingly, *SL* can quickly locate the start and end indexes of fuzzy items in *FLBuf*.

2) We use the Estimated Fuzzy utility Co-occurrence Structure (named EFuCS) to store the upper-bound of high-level itemsets. And the corresponding Estimated Fuzzy utility of itemset to Prune (named EF2P) strategy is used to prune the search space in advance.

3) We conduct extensive experiments on eight datasets (including real and synthetic) to demonstrate the effectiveness and efficiency of our novel algorithm. We also compare the performances of the HUIM and FUM algorithms. The experiments show that the FS-IUM outperforms other state-of-the-art algorithms.

The rest content of this paper is organized as follows. Section 2 briefly reviews the related work in HUIM and FUM domains. Thereafter, Section 3 introduces the preliminaries and defines the problem of high fuzzy utility itemset mining. The details of our proposed algorithm are presented in Section 4. In Section 5, the experimental results are discussed. In the end, we summarize this paper and plan the future work in Section 6.

2 Related Work

2.1 High Utility Itemset Mining

Interesting measures play a vital role in knowledge discovery [1-2, 17-18]. Due to relative importance of distinct items are not considered in frequent itemset mining, utility-based measures use the utilities of itemsets to reflect the user's goals. Since Shen *et al.* [6] first proposed utility constraint rule mining task, there are a multitude of investigators have hastened to improve the performance of high utility itemset mining (HUIM). In fact, all the proposed HUIM algorithms can be roughly divided into two classes. The first class is two phase model. Taking the most famous approach Two-Phase [8] as example, its mining mechanism is as follows: 1) In the Phase I, Two-Phase will scan the database and then generates a mass of candidates by using *TWU* upper-bound. We call Phase I as generating part; and 2) In the Phase II, it scans database multiple times and computes the real utility of candidates to discover HUIs. This phase is defined as checking part. The Two-Phase algorithm will repeat two phases until no more HUIs are generated. To reduce the number of candidates generation, studies [19-21] keep original database in a tree data structure. The experiments show the latter saves more time and memory usage than the former.

However, as previous content we mentioned, scanning database multiply is costly and unacceptable. Hence, the second class (one phase model) was proposed as a solution to the above problem. The key idea of one phase algorithms is utilizing efficient data structure to store major information of itemsets in memory. HUI-Miner [9] is the first and most famous list-based one phase algorithm. It only needs to scan the database twice times and then construct utility-lists for different itemsets. The utility-list collects all key information (i.e., transaction identification, utility value, and remaining utility) of an itemset. Then, the super-itemset can be obtained by joining two distinct utility-lists. Recently, Duong *et al.* [12] figured out that the major computational cost of list-based algorithm is maintaining massive utility-lists in memory. Therefore, they proposed a reusable utility-list buffer data structure. The novel structure only uses a portion of space to store basic information of promising items, and the rest is reused to keep information of itemsets. Compared with the state-of-the-art algorithms HUI-Miner [9] and FHM [10], the peak memory consumption of their novel algorithm was reduced nearly to six times, and the performance in terms of runtime cost also plays very well. All in all, there is a lot of information available in literature about HUIM. Further details can be found in studies [4, 22-24].

2.2 Fuzzy Utility Mining

As previously mentioned, high utility itemsets are usually lack of comprehensibility attribute. How to mine explainable results emerges as an important topic. Mining fuzzy itemsets is a solution to the above issue. Since Wang *et al.* [14] first proposed the fuzzy utility mining task, researchers have found a new way to get explainable patterns. Lan *et al.* continuously proposed two studies [15, 25]. And they designed a new fuzzy utility upper-bound (*FUUB*) to accelerate the mining process. In FUM algorithms, *FUUB* plays the same role as *TWU* works in HUIM. It can filter unpromising items after scans database once and is greatly reduces the number of candidates generation during mining process. In their proposed works, three parameters (user-defined membership function, a quantitative transaction database, and a user-specified minimal utility threshold) are taken as input, and then the algorithm yields high fuzzy utility itemsets (HFUIs) as output. The membership function is used to calculate different linguistic region values (e.g., *Low*, *Middle*, and *High*) of an item. Since a fuzzy itemset consists of distinct fuzzy

items, Lan *et al.* [15] utilized the minimum operator principle to compute the fuzzy utility of the itemset. That is, for example, if region values of two fuzzy items *egg.Low* and *milk.Middle* are 0.6 and 0.4 respectively, then the region value of fuzzy itemset {*egg.Low*, *milk.Middle*} is 0.4. In other words, the region value of *egg.Low* in the fuzzy itemset is set as 0.4 rather than 0.6. However, all above discussed approaches belong to two phase model. Recently, Wan *et al.* [16] proposed an efficient one phase algorithm named FUIM. They first proposed the remaining fuzzy utility notion, which is a tighter upper-bound than *FUUB*. The experiments reveal that FUIM performs better than previous algorithms.

At the same time, Huang *et al.* [26] noticed that traditional FUM algorithms do not consider the temporal factor, which plays an important role in many data analytic systems and applications. Hence, they proposed a new mining task called temporal fuzzy utility itemset mining (TFUIM) to solve this issue. Subsequently, there are other studies [27-28] continuously proposed to improve the performance of TFUIM algorithms. All in all, we suppose that FUIM adopts naive method to intersect different fuzzy-lists will cause poor efficiency. Thus, in this paper, we will propose a novel fuzzy utility itemset mining algorithm which performs better than FUIM in terms of both runtime and memory usage.

3 Preliminaries and Problem Statement

In this section, the basic definitions our novel algorithm adopted will be introduced and exemplified. Most notations are provided in studies [15-16], and we formulate the problem statement of fuzzy utility itemset mining finally.

3.1 Basic Preliminaries

In the novel algorithm, we assume $I = \{x_1, x_2, \ldots, x_n\}$ is a set of n different items. The itemset $X = \{x_1, x_2, \ldots, x_k\}$is a superset of items. We call X is an k-itemset if it consists of k distinct items where $|X| = k$. A quantitative transaction database $\mathcal{D} = \{T_1, T_2, \ldots, T_m\}$, and the transaction T_m is a subset of I. In particular, an item x_i in $\mathcal{D}$ has an external utility (e.g., unit profit) $p(x_i)$ and an internal utility (e.g., quantity) $q(x_i, T_i)$. We also take a sample transaction database (Table 1) as our running example in this paper, and the external utility of A, B, C, and D is \$5, \$3, \$2, and \$7 respectively. At the same time, the membership function is shown in Figure 1 and defines three regions *Low*, *Middle*, and *High*.

Definition 1: In the user-specified membership function, a fuzzy set f_{ij} of an item x_i in transaction T_j is defined as

$$f_{ij} = \left(\frac{f_{ij1}}{R_{i1}} + \frac{f_{ij2}}{R_{i2}} + \ldots + \frac{f_{ijl}}{R_{il}}\right), \quad (1)$$

where l is the number of regions given by the membership function. R_{il} is the l-th fuzzy region value of x_i, and $f_{ijl} \in [0,1]$. In addition, the fuzzy utility of the fuzzy item x_{il} in T_j is denoted as

$$fu_{ijl}(x_{il}, T_j) = f_{ijl} \times q(x_i, T_j) \times p(x_i), \quad (2)$$

and its total fuzzy futility in is defined as

$$fu_{il}(x_{il}) = \sum_{x_i \in T_j \wedge T_j \in \mathcal{D}} fu_{ijl}(x_i, T_j). \quad (3)$$

For example, consider the internal utility of an item D in transaction T_6 in Table 1 (that is $q(D, T_6) = 7$), $f_{D,6}$ is (0/*D.Low*, 0.8/*D.Middle*, 0.2/*D.High*) by the given membership function (Figure 1). Then, the fuzzy utility of fuzzy item *D.Middle* is 0.8×7×\$7, where is \$39.2.

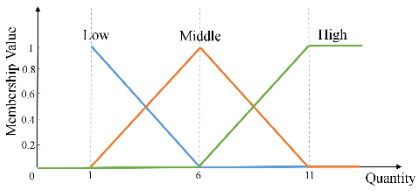

Figure 1. The membership function

Table 1. A simple quantitative transaction database

Tid	*A*	*B*	*C*	*D*
T_1	2	11	4	3
T_2	0	0	5	0
T_3	9	0	0	0
T_4	10	0	0	2
T_5	6	0	1	3
T_6	0	2	3	7
T_7	0	8	1	0
T_8	0	0	0	4
T_9	5	0	1	3
T_{10}	3	0	5	0

Definition 2: An itemset consists of several distinct items. Similarly, a fuzzy itemset is a superset of distinct fuzzy items. The fuzzy utility of a fuzzy itemset X in transaction T_j is defined as

$$fu_{jx}\left(X, T_j\right) = f_{jX} \times \sum_{x_{il} \in X \wedge X \subseteq T_j} fu_{ijl}(x_i, T_j), \quad (4)$$

and its corresponding fuzzy utility in $\mathcal{D}$ is denoted as

$$fu_X\left(X\right) = \sum_{X \subseteq T_j \wedge T_j \in \mathcal{D}} fu_{jx}(X, T_j), \quad (5)$$

Where f_{jx} is the minimal fuzzy region values of all fuzzy items in X. In addition, different fuzzy regions of an item cannot occur in a fuzzy itemset at the same time. In other words, an item cannot both *Low* and *High* in a fuzzy itemset.

For example, let fuzzy itemset X be {*B.Low*, *C.Middle*} in transaction T_6, the fuzzy region values of *B.Low* and *C.Middle* are 0.8 and 0.4, respectively. Thus, the fuzzy utility of {*B.Low*, *C.Middle*} in T_6 is 0.4 × ((2 × \$3) + (3 ×\$2)) = \$4.8, and the total fuzzy utility of X in $\mathcal{D}$ is $fu_{6,X}$ = \$4.8.

Definition 3: The fuzzy utility of a transaction T_j is denoted as

$$tfu\left(T_j\right) = \sum_{x_{il} \in T_j} fu_{ijl}(x_{il}, T_j). \quad (6)$$

Furthermore, the fuzzy utility of a quantitative transaction database $\mathcal{D}$ is defined as

$$fu_{\mathcal{D}} = \sum_{T_j \in \mathcal{D}} tfu(T_j). \quad (7)$$

Definition 4: Given a user-specified minimal utility threshold δ ($0 < \delta < 1$), if the total fuzzy utility of fuzzy itemset X is no less than *minUtil* ($fu_{\mathcal{D}} \times \delta$), then we suppose X is a high fuzzy utility itemset (simplified as HFUI). To simplify the expression, we will use "*minUtil*" to represent the user-specified minimal fuzzy utility in the rest paper rather than the formula.

For example, consider the example database (Table 1) and membership function (Figure 1), assume a fuzzy itemset X is *C.Low*. $fu_X(X) = fu_{1,X} + fu_{2,X} + fu_{5,X} + fu_{6,X} + fu_{7,X} + fu_{9,X} + fu_{10,X}$ = \$16.8. And the total fuzzy utility of its superset Y = {*C.Low*, *D.Middle*} is $fu_Y(Y) = fu_{1,Y} + fu_{5,Y} + fu_{6,Y} + fu_{9,Y}$ = \$63. If we assume δ = \$60, then Y is an HFUI, but X is not.

In the latest example, fuzzy itemset {*C.Low*, *D.Middle*} is an HFUI while its subset {*C.Low*} is not. This case indicates that fuzzy utility does not hold the downward-closure property. Therefore, to address this problem, we utilize the fuzzy utility upper bound (*FUUB*) [15-16]. The related notions and definitions are listed as follows.

Definition 5: The maximal fuzzy utility of an item x_i in transaction T_j is formulated as

$$mfu_{ij} = \max \{fu_{ij1}(x_i, T_j), fu_{ij2}(x_i, T_j), \ldots\}, \quad (8)$$

and the maximal transaction fuzzy utility of T_j is

$$mtfu_j = \sum_{x_i \in T_j} mfu_{ij}. \quad (9)$$

For example, consider Table 1 and Figure 1, the fuzzy utility of fuzzy items *D.Low* and *D.Middle* in transaction T_1 are \$12.6 and \$8.4. Thus, $mfu_{D,1}$ is \$12.6. In transaction T_4, the maximal fuzzy utility of items *A* and *D* are \$40 and \$11.2, respectively. Apparently, $mtfu_4 = mfu_{A,4} + mfu_{D,4}$ = \$51.2.

Definition 6: The fuzzy utility upper bound of a fuzzy itemset X is denoted as

$$FUUB_X = \sum_{X \subseteq T_j \wedge T_j \in \mathcal{D}} mtfu_j. \quad (10)$$

Given a user-specified minimum utility threshold δ, X is a high fuzzy utility upper bound itemset (*HFUUBI*) if and only if $FUUB_X$ is no less than *minUtil*.

Table 2. HFUIs w.r.t. *minUtil* = \$60

Fuzzy Itemset	*Utility*
{*A.Middle*}	\$86
{*A.High*}	\$67
{*D.Low*}	\$60.2
{*D.Middle*}	\$84
{*A.Middle*, *D.Low*}	\$77.2
{*C.Low*, *A.Middle*}	\$62.2
{*C.Low*, *D.Middle*}	\$63
{*C.Low*, *A.Middle*, *D.Low*}	\$68.4

Property 1. *Let fuzzy itemset X be a superset of another fuzzy itemset Y. Obviously,* $FUUB_X$ *is always no less than* $fu_X(X)$*, and* $FUUB_X \geq FUUB_Y$*. That is, if a fuzzy itemset is not an HFUUBI, it must be a low fuzzy utility itemset. The study* [15] *offers the proof details.*

3.2 Problem Statement

Given a user-defined membership function, a user-specified utility threshold, and a quantitative transaction database, the problem of high fuzzy utility itemset mining can be interpreted as discovering a complete set of high fuzzy itemsets from the quantitative transaction database.

For example, consider the sample database and membership function (Table 1 and Figure 1), if we set *minUtil* as \$60, a set of high fuzzy utility itemsets are listed in Table 2.

4 The Proposed Algorithm

As we have discussed in previous content, the major disadvantage of FUIM is its high cost because of its naive join operation. In this section, we propose a high fuzzy utility itemset mining algorithm (namely FS-IUM). In the following content, we also introduce several data structures and pruning strategies.

4.1 The Fuzzy-List Structure

Before introducing the fuzzy-list structure, we first talk the adopted ordering rule. Let $\prec$ be a global *FUUB*-ascending order on items from *I*. With $\prec$ order, a transaction can be sorted as a new revised transaction. Thus, in a revised transaction T_j, these fuzzy items after X are called remaining fuzzy items [16], and the summation of their maximum fuzzy utility is named remaining maximal fuzzy utility

$$rfu(X, T_j) = \sum_{x_i \in T_j \wedge X \prec x_i} mfu_{ij}, \quad (11)$$

which can easily estimate how much the fuzzy utility of X can be increased.

Definition 7: As shown in Figure 2, a fuzzy-list structure of a fuzzy itemset X is denoted as *ful*(X) which consists of several tuples. A tuple contains three elements: transaction identification (*Tid*), fuzzy utility ($fu(X, T_j)$) and remaining fuzzy utility ($rfu(X, T_j)$). In addition, the total fuzzy utility of X in $\mathcal{D}$ is the summation of all $fu(X, T_j)$ in *ful*(X), which is denoted as

$$sumFu(X) = \sum_{T_j \in ful(X)} fu(X, T_j), \quad (12)$$

and the summation of all remaining fuzzy utility of X is defined as

$$sumRfu(X) = \sum_{T_j \in ful(X)} rfu(X, T_j). \quad (13)$$

C.Low

tid	fu	rfu
1	3.2	20.6
2	2	0
5	2	42.6
6	3.6	39.2
7	2	0
9	2	32.6
10	2	9

A.Middle

tid	fu	rfu
1	2	12.6
3	18	0
4	10	11.2
5	30	12.6
9	20	12.6
10	6	0

D.Low

tid	fu	rfu
1	12.6	0
4	11.2	0
5	12.6	0
8	11.2	0
9	12.6	0

D.Middle

tid	fu	rfu
1	8.4	0
4	2.8	0
5	8.4	0
6	39.2	0
8	16.8	0
9	8.4	0

Figure 2. The fuzzy-lists of some fuzzy 1-itemsets

Property 2. *Let X be a fuzzy itemset. If sumFu*(*X*) *in ful*(*X*) *is higher than or equal to minUtil, we suppose X is an HFUI. Otherwise, it is a low fuzzy utility itemset* [16].

Property 3. *Let X be a fuzzy itemset. If the sum of sumFu and sumRfu of X in ful*(*X*) *is less than minUtil, we assume all the supersets of X are low fuzzy utility itemsets* [16].

Property 4. *Given two different fuzzy itemsets X and Y, if*

$$\sum_{T_j \in ful(X)} (fu(X,T_j) + rfu(X,T_j)) - \sum_{T_j \in ful(X) \wedge Y \subseteq T_j} (fu(X,T_j) + rfu(X,T_j))$$

is less than minUtil, then the fuzzy super-itemset XY and all its extensions cannot be HFUIs [16].

Due to the space limitation of this paper, the proof details of Properties 2, 3 and 4 are provided in Ref. [16].

4.2 The Fuzzy-List Buffer Structure

In the FUIM algorithm, joining fuzzy-lists is a significant step to obtain fuzzy utility information about fuzzy *k*-itemsets (where *k* is higher than 1). The key part of joining operation is locating identical *Tids* in fuzzy-lists. FUIM adopts a binary search method on fuzzy-lists to check whether there are elements with the same *Tid*. The time complexity is $\mathcal{O}(mlogn)$, where *m* and *n* are the size of two joined fuzzy-lists. However, joining operation is costly in terms of runtime. Especially, the algorithm has to maintain massive fuzzy-lists until all of them have already been compared. Therefore, the memory usage of FUIM is also intolerable. Then, we propose a fuzzy-list buffer data structure to solve this issue.

Definition 8: The fuzzy-list buffer is like a memory pipeline, it is defined as

$$FLBuf = \{e | e \in ful(X)\}, \quad \textbf{(14)}$$

where *X* is a fuzzy itemset, and *e* is a tuple of fuzzy-list of *X*. We call the tuple of buffer structure as data segment [12].

Definition 9: To quickly find identical *Tids* in *FLBuf*, we also adopt index segment structure [12]. Given a fuzzy itemset *X*, the index segment of *X* is a summary fuzzy-list (*SL*(*X*)) which records the index information of *X* in *FLBuf*. An index segment of *X* is a tuple (*X*, *startPos*, *endPos*, *sumFu*, *sumRfu*). In addition, all the *SL*(*X*) can also be stored in a memory pipeline as the same as *FLBuf*.

The *FLBuf* and *SL* structures are shown in Figure 3. Take fuzzy item *D.Middle* as an example, the summary information of *D.Middle* consists of the following information: the item is *D.Middle*, its start position index and end position index in the lists are 6 and 12, the summation of its fuzzy utility is $84, and the summation of its remaining fuzzy utility is $0. It is also efficient to obtain a fuzzy-list stored in the *FLBuf* using the *SL* structure. For instance, after accessing the summary fuzzy-list of fuzzy item *D.Middle*, its fuzzy-list can be read directly in the *FLBuf* structure from the *SL*({*D.Middle*}).*startPos* to *SL*({*D.Middle*}).*endPos* (in blue segment in Figure 3 (up)).

Tids =	1	3	4	5	9	10	1	4	5	6	8	9
fus =	2	18	10	30	20	6	8.4	2.8	8.4	39.2	16.8	8.4
rfus =	12.6	0	11.2	12.6	12.6	0	0	0	0	0	0	0

Itemsets =	{AM}	{DM}
StartPoss =	0	6
EndPoss =	6	12
SumFus =	86	84
SumRfus =	49	0

Figure 3. The *FLBuf* (up) and *SL* (down) structures of two fuzzy 1-itemsets

4.3 The Estimated Fuzzy Utility Co-occurrence Structure

Definition 10: Let I^* be a set of high fuzzy utility upper bound 1-itemset, and the elements of I^* are sorted in $\prec$ order. The new structure consists of several triples of the form ($x \in I^*$, $y \in I^*$, $z \in R$). A triple (*x*, *y*, *z*) indicates that $FUUB_{xy} = z$. The *EFuCS* can be implemented by hashmap where $z \geq minUtil$ and $z \neq \$0$.

Property 5. *Let Px and Py are two different fuzzy itemsets. In EFuCS structure, if* $z = FUUB_{P_{xy}}$ *is less than minUtil, the new fuzzy itemset Pxy and all its supersets are low fuzzy utility itemsets. The proof details directly follow from* ***Property 1****.*

4.4 Efficient Pruning Strategies

Due to the utilized data structures are more complex than simple fuzzy-list, adopting efficient pruning strategies to avoid degrading the mining performance is urgency. In our novel algorithm, we utilize the following pruning strategies.

Strategy 1. Consider the **Property 1**, if *FUUB* of a fuzzy itemset is less than *minUtil*, we can safely prune the fuzzy itemset and its supersets during the following process.

We notice that high fuzzy upper bound 1-itemsets ($FUUB_1$) are computed after the first-time database scans. Consider the **Strategy 1**, the algorithm only needs to construct *FLBuf* of $FUUBI_1$, and the search space during mining becomes smaller than before. Then, the following pruning strategies can be optimized with the same manner.

Strategy 2. Consider the **Property 5**, the estimated fuzzy utility of itemset to prune strategy (EF2P) is declared as follows. If the *FUUB* value of two different fuzzy itemsets is no less than *minUtil*, then we assume the combination of two fuzzy itemsets is a potential HFUI. Otherwise, we can prune the search space to speed up the mining process.

The *EFuCS* structure can reduce the useless joining operation times of fuzzy itemsets when certain requirements are satisfied. The *EFuCS* structure and its corresponding EF2P strategy therefore can decrease a number of candidates.

Strategy 3. According to the **Property 3**, if the summation of *sumFu* and *sumRfu* values of a fuzzy itemset is less than *minUtil*, then we prune this fuzzy itemset as well as its supersets to reduce the search space.

Strategy 4. Based on the **Property 4**, if the inequality is true, it is no need to generate the high-level fuzzy itemsets by joining two related fuzzy itemsets.

Inspired by the study [29], the early abandoning strategy

(EA) is applied to exit the fuzzy-list construction process in advance when specific conditions are satisfied. The EA strategy is introduced in the following.

Strategy 6. Let *EAMeasure* be the sum of *sumFu* and *sumRfu* of two distinct fuzzy itemsets. When constructing a fuzzy-list, the approach subtracts the sum of *fu* and *rfu* from *EAMeasure* for those transactions whose *Tids* are not equal. If the *EAMeasure* value is less than *minUtil*, the construction process can quit early, and the rest of transactions will be abandoned.

4.5 The FLBuf Reusing Memory Construct Method

Now we give the introduction of the *FLBuf* structure. In FUIM, after constructing a fuzzy-list of a certain fuzzy itemset, the fuzzy-list may no longer be used. This causes a waste of the memory allocated for storing the fuzzy-list of the fuzzy itemset. A novel method for constructing the fuzzy-list buffer is shown in Algorithm 1. The main operations for memory re-utilization are shown as follows. If a fuzzy itemset is no longer used to explore the search space, then the allocated memory that is used to store fuzzy-list will be reused for the next promising fuzzy itemset. In addition, new memory is allocated only when the *FLBuf* is at capacity (lines 17-26). Furthermore, based on **Strategy 6**, the EA strategy is applied to the construction of *FLBuf* using the variable *EAMeasure*. The detailed implementation is shown in Algorithm 1.

4.6 Proposed FS-IUM Algorithm

The Algorithm 2 is the search procedure. It takes six input parameters: 1) a prefix fuzzy itemset *P*, 2) a set of extension fuzzy items of *P ExtensionsOfP*, 3) a user-specified minimal utility *minUtil*, 4) the *EFuCS* structure, 5) the *FLBuf* structure, and 6) the summary fuzzy-list *SL*. For each fuzzy itemset $X \in ExtensionsOfP$, *X* is firstly checked to learn whether it is an HFUI (lines 2-4). Then, according to **Strategy 3**, if the sum of *sumFu* and *sumRfu* of *X* is no less than *minUtil*, the fuzzy super-itemsets of *X* are regarded as potential HFUIs and should be further examined (line 5). Then, in line 7, the procedure checks each fuzzy itemset $Y \in ExtensionsOfP$ ($X \prec Y$) because all 1-itemsets in *FLBuf* are sorted with *FUUB*-ascending order. The algorithm checks whether there exists two fuzzy itemsets in *EFuCS* that their *FUUB* value is no less than *minUtil* (line 8). If there exists one, the procedure will call the *FLBuf*-Construct method (cf. Algorithm 1) to construct the fuzzy-list segment of *Pxy* by joining the fuzzy-list segments of *P*, *X* and *Y* (line 9). According to **Strategy 4**, if the *sumFu* value of *XY* in *SL* is higher than 0, then the procedure will combine *X* with *Y* to create a high-level fuzzy itemset *Pxy* and add it to the extensions of *X* (lines 10-13). Finally, the procedure recursively calls itself until there are no HFUIs generated (line 18).

Based on previous introduction, the main pseudocode is shown in Algorithm 3. The input parameters are 1) a quantitative transaction database $\mathcal{D}$, 2) a pre-defined membership function *R*, and 3) a user-specified fuzzy utility *minUtil*. The outcome is a set of HFUIs. The main algorithm carries out the following steps. At the first time, it scans $\mathcal{D}$ to compute the *FUUB* value of all items x_i in $T_j \in \mathcal{D}$ by the membership function *R* in line 1. Next, based on the *FUUB* values, it builds the promising set I^*, which consists of all fuzzy items meet condition (i.e., $FUUB \geq minUtil$) (line 2). Then, the algorithm sorts all $x_i \in I^*$ by the *FUUB*-ascending order (line 3). The main algorithm scans the database $\mathcal{D}$ for the second time to construct the *FLBuf*, *SL*, and *EFuCS* structures in line 4. At last, when the mining process terminates, the algorithm calls a recursive method (cf. Algorithm 2) to find all HFUIs and output the results (lines 5 and 6).

Algorithm 1. *FLBuf*-Construct procedure

Input:
FLBuf, SL: buffer structure;
P, *Px*, *Py*: fuzzy itemsets;
minUtil: user-given minimal utility threshold.
Output:
Status of updating *FLBuf*, *SL* with fuzzy itemset *Pxy*;

```
1: let PPnt, PxPnt, PyPnt be three pointers that initially
   point to FLBuf at positions SL(P).StartPos, SL(Px).Start
   Pos, SL(Py).StartPos, respectively;
2: let EAMeasure=SL(Px).sumFu + SL(Px).sumRfu +
   SL(Py).sumFu + SL(Py).sumRfu;
3: let insertPos = SL.Last.EndPos;
4: while PxPnt ≠ SL(Px).EndPos and PyPnt ≠
   SL(Py).EndPos do
5:     if Tids[PxPnt] < Tids[PyPnt] then
6:         PxPnt++;
7:         subtract EAMeasure by (fus[PxPnt] +
           rfus[PxPnt]);
8:     else if Tids[PxPnt] > Tids[PyPnt] then
9:         PyPnt++;
10:        subtract EAMeasure by (fus[PyPnt] + rfus[PyPnt]);
11:    else
12:        if SL(P) ≠ null then
13:            while PPnt ≠ SL(P).EndPos and
                   Tids[PPnt] ≠ Tids[PxPnt] do
14:                PPnt++;
15:            end
16:        end
17:        if insertPos ≥ |FLBuf| then
18:            FLBuf.Tids[Tids.count+] = Tids[PxPnt];
19:            FLBuf.fus[fus.count+] = fus[PxPnt]
+ fus[PyPnt] – fus[PPnt];
20:            FLBuf.rfus[rfus.count+] = rfus[PyPnt];
21:        else
22:            insertPos++; /* reuse memory */
23:            FLBuf.Tids[insertPos] = Tids[PxPnt];
24:            FLBuf.fus[insertPos] = fus[PxPnt] +
                   fus[PyPnt] – fus[PPnt];
25:            FLBuf.rfus[insertPos] = rfus[PyPnt];
26:        end
27:        PxPnt++;
28:        PyPnt++;
29:    end
30: end
31: if EAMeasure < minUtil then
32:    return false
33: end
34: update SL(Pxy);
35: return true
```

Algorithm 2. The search procedure

Input:
P: a prefix fuzzy itemset;
ExtensionsOfP: a set of extension fuzzy items of *P*;
minUtil: a user-specified minimal utility threshold;
EFuCS: the estimated fuzzy utility co-occurrence structure;
FLBuf: the fuzzy-list buffer structure;
SL: the summary fuzzy-list.

```
1: for X ∈ ExtensionsOfP do
2:    if SL(X).sumFu ≥ minUtil then
3:       HFUIs ← X;
4:    end
5:    if SL(X).sumFu + SL(X).sumRfu ≥ minUtil then
6:       initialize ExtensionsOfX as null;
7:       for Y after X in ExtensionsOfP do
8:          if ∃ (X, Y, z) ∈ EFuCS and z ≥ minUtil then
9:             if FLBuf-Construct(FLBuf, SL, P, X, Y,
                  minUtil) == true then
10:               Pxy ← X ⋃ Y;
11:               if SL(Pxy).sumFu > 0 then
12:                  ExtensionsOfX ← ExtensionsOfX ⋃ Pxy;
13:               end
14:            end
15:         end
16:      end
17:      Px ← P ⋃ X;
18:      call Search(Px, ExtensionsOfX, minUtil,
            EFuCS, FLBuf, SL);
19:   end
20: end
```

Algorithm 3. The FS-IUM algorithm

Input:
$\mathcal{D}$: a quantitative transaction database;
R: a pre-defined membership function;
minUtil: a user-specified minimal utility threshold.
Output:
HFUIs: A set of high fuzzy utility itemsets;

```
1: scan D to compute the FUUB of all items x_i by R;
2: let I* be the set of fuzzy items x, where FUUB_x ≥ minUtil;
3: sort all x_i ∈ I* by FUUB-ascending order;
4: scan D again to build FLBuf, SL, and EFuCS;
5: call Search (ϕ, I*, minUtil, EFuCS, FLBuf, SL);
6: return HFUIs
```

5 Performance Evaluation

In this section, we describe various experiments to assess the performance of the proposed FS-IUM algorithm. We select several compared algorithms in the following. We had compared the performance of FS-IUM with FUIM [16] and two well-known HUIM algorithms (FHM_{eucst} and ULB-Miner [12]) in terms of runtime and memory usage. We modify *EUCS* structure of FHM [10] as similar to *EFuCS* in our algorithm. To further assess the efficiency of the *EFuCS* structure, a various version named FLB-Miner is utilized, which yields HFUIs without utilizing the *EFuCS* structure and EF2P strategy. Each experiment was tested at least three times. In addition, if the tested algorithm runs out of 600,000 seconds, we suppose the algorithm cannot get the right results. Hence, we use "-" to represent it in tables.

5.1 Experimental Setup

The experimental algorithms were implemented with Java language, and the experiments were conducted on the Windows 10 (64-bit) Professional Edition system with an Intel Core i7 processor of 2.5 GHz and 16 GB of RAM.

In our experiments, we used eight different real and synthetic datasets. All datasets can be freely downloaded from the SPMF library (http://www.philippe-fournier-viger.com/spmf/index.php). The external utility of each item was created randomly from 1 to 10,000 by using a log-normal distribution approach. And the internal utility of each item in each transaction was randomly created within a range (1-6) for the Foodmart dataset, and within a range (1-5) for the other datasets. Table 3 shows more details of all datasets, including the type of all datasets, the number of transactions and distinct items, as well as the maximum and average length of all transactions. We assumed all items have the same membership function with three fuzzy regions (*Low*, *Middle*, and *High*). The membership function is shown in Figure 4.

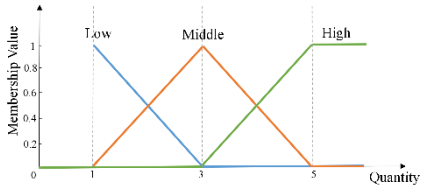

Figure 4. Membership function in the experiments

Table 3. Characteristics of experimental datasets

Dataset	#Trans	#Items Signature (MS)	AvgLen	MaxLen	#Type
Retail	88,163	16,471	10.3	76	Sparse
Foodmart	4,141	1,559	4.4	11	Sparse
BMSPOS	515,367	1,658	6.5	164	Sparse
T10I4D100K	100,000	1,001	10.1	29	Sparse
Chess	3,196	75	37	37	Dense
Accident	340,183	468	33.8	49	Dense
Mushroom	8,124	119	23	23	Dense
T40I10D100K	100,00	1,001	39.6	77	Dense

5.2 Runtime Cost Analysis

To better understand the runtime of the tested algorithms, we compared their runtime in all datasets under various thresholds (δ). As shown in Figure 5, our proposed FS-IUM algorithm performs significantly better than other algorithms. In particular, the advantage of our novel algorithm is more obvious while the threshold decreases. In the meanwhile, the HUIM algorithms, FHM_{eucst} and ULB-Miner, often run out of time. For example, on the Retail dataset, FS-IUM only takes half of running time of FUIM to discover a complete set of HFUIs, but all HUIM algorithms run out of time. Moreover, it can be observed that FLB-Miner also spends less runtime compared to FUIM in general. Thanks to the fuzzy-list buffer

structure and its efficient auxiliary technique, as well as the pruning strategies we adopted like the EF2P strategy, FS-IUM can strikingly reduce the execution time. Furthermore, we notice that ULB-Miner performs better than FS-IUM in some cases. The main reason we suppose is that ULB-Miner generates fewer candidates than FS-IUM does in these datasets. That is why the rest HUIM algorithms yield numerous candidates and oftentimes run out of time. In a word, the proposed FS-IUM algorithm can efficiently mine HFUIs and outperforms the compared algorithms in all datasets.

5.3 Memory Usage Analysis

We compare the maximum memory usage of the tested algorithms on all datasets (the thresholds δ are the same as Figure 5), and the details are shown in Table 4. It can be observed that the proposed novel algorithm outperforms the other algorithms in most cases. The tested HUIM algorithms are still unable to complete the mining process on some datasets, so we do not list their results on these datasets. For example, on the Mushroom dataset, the peak memory usages of FUIM, FLB-Miner, and FS-IUM are 325.8 MB, 395.6 MB, and 102.5 MB, respectively. This indicates that FS-IUM uses about three times less memory than that of FUIM and FLB-Miner. Since the fuzzy-list buffer structure and the *FLBuf* construct method are capable of reusing memory, the memory consumption of maintaining information in *FLBuf* which is no longer used will be recalled and reused for other high-level fuzzy itemsets. Hence, as mentioned above, even though FS-IUM may create more candidates than that of FUIM, FS-IUM can still reduce memory usage in most cases. And the FS-IUM algorithm decreases memory consumption dramatically. In a word, massive memory is saved and reduced in FS-IUM, and thus FS-IUM costs less memory than other compared algorithms.

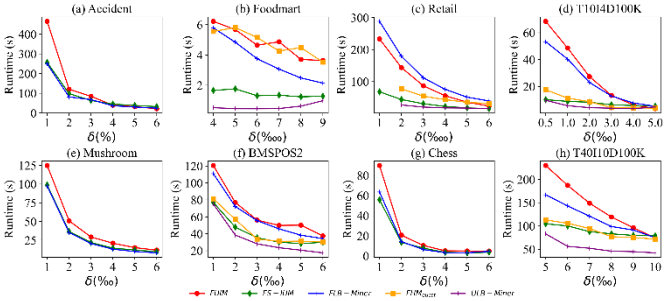

Figure 5. Runtime comparison with various δ

Table 4. Compare the peak memory (MB)

Dataset	FHM_{eucst}	ULB-Miner	FUIM	FLB-Miner	FS-IUM
Retail	-	-	652.3	1411.3	520.3
Foodmart	160.5	268.6	110.7	244.8	61.7
BMSPOS	1107.1	1148.5	1079.6	1100.1	979.7
T10I4D100K	787.1	583.5	446.7	832.8	514.6
Chess	-	-	298.1	339.9	106.2
Accident	-	-	1205.6	498	491.7
Mushroom	-	-	325.8	395.6	102.5
T40I10D100K	1195.6	1252.8	1211.7	1155.9	1190

5.4 Scalability Analysis

In the last subsection, we conduct experiments to compare the scalability of FUIM, FLB-Miner and the proposed FS-IUM on a real-life dataset (i.e., Accident). In each test, the dataset size ($|\mathcal{D}|$) is 40k, 55k, 70k, 85k, and 100k transactions when δ is fixed to 1% ("k" is equal to 1,000). The experimental results in terms of runtime and memory usages are shown in Figure 6. It can be observed that, along with the dataset size increases, both the runtime and memory consumption of FS-IUM increase more smoothly than those of other algorithms. The proposed algorithm has almost linear scalability when the dataset size increases in terms of runtime usage. In summary, FS-IUM outperforms other compared algorithms on the scalability test.

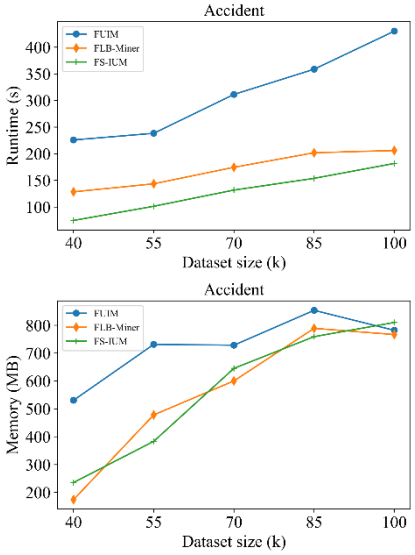

Figure 6. The scalability test under varied $|\mathcal{D}|$

6 Conclusions

Existing fuzzy-list-based algorithms for mining high fuzzy utility itemsets are efficient and easy to implement. However, they require a large amount of memory for maintaining fuzzy-lists. To solve this problem, in this paper, we proposed a novel memory reuse structure called fuzzy-list buffer. This paper presents an efficient algorithm called FS-IUM for high fuzzy utility itemset mining. The new algorithm combines the fuzzy-list buffer structure with efficient techniques for building fuzzy-list segments to decrease the runtime and memory consumption. The *FLBuf* structure and its ancillary structure (*SL*) help to locate and retrieve the fuzzy-list quickly. According to the efficient *EFuCS* structure and EF2P strategy, FS-IUM can significantly reduce runtime consumption compared to the state-of-the-art algorithms. In addition, FS-IUM decreases memory usage by the fuzzy-list buffer construction method for reusing memory. Finally, we have conducted an extensive experiment on eight datasets to compare the performances of FHM_{eucst}, ULB-Miner, FUIM, FLB-Miner, and FS-IUM. Our experimental results demonstrate that the proposed FS-IUM algorithm improves the efficiency of mining.

In future work, we plan to further improve the effective-

ness of FS-IUM, such as applying more efficient pruning strategies and investigating other optimization techniques. Furthermore, since data stream mining has become a popular research topic in many domains recently, how to apply the fuzzy-list buffer structure to tackle the data stream mining issue will be an interesting and challenging task.

Acknowledgement

This work was partially supported by the National Natural Science Foundation of China (Grant Nos. 61902079 and 62002136), Guangzhou Basic and Applied Basic Research Foundation (Grant Nos. 202102020928 and 202102020277).

References

[1] R. Agrawal, R. Srikant, Fast algorithms for mining association rules in Large Databases, *Proceedings of the 20th ACM International Conference on Very Large Data Bases*, Santiago de Chile, Chile, 1994, pp. 487-499.

[2] J. Han. J. Pei, Y. Yin, Mining frequent patterns without candidate generation, *ACM SIGMOD Record*, Vol. 29, No. 2, pp. 1-12, June, 2000.

[3] J. Pei, J. Han, H. Lu, S. Nishio, S. Tang, D. Yang, H-Mine: Fast and space-preserving frequent pattern mining in large databases, *IIE Transactions*, Vol. 39, No. 6, pp. 593-605, March, 2007.

[4] W. Gan, J. C. W. Lin, P. Fournier-Viger, H. C. Chao, V. S. Tseng, P. S. Yu, A survey of utility-oriented pattern mining, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 33, No.4, pp. 1306-1327, April, 2021.

[5] T. Hutchison, A. Marshall, C. Guillebaud, Reviews: Principles of economics, *The Economic History Review*, Vol. 15, No. 3, pp. 558-560, April, 1963.

[6] Y. Shen, Z. Zhang, Q. Yang, Objective-oriented utility-based association mining, *Proceedings of the 2nd IEEE International Conference on Data Mining*, Maebashi, Japan, 2002, pp. 426-433.

[7] H. Yao, H. J. Hamilton, C. J. Butz, A foundational approach to mining itemset utilities from databases, *Proceedings of the 4th SIAM International Conference on Data Mining*, Lake Buena Vista, Florida, USA, 2004, pp. 482-486.

[8] Y. Liu, W. K. Liao, A. Choudhary, A two-phase algorithm for fast discovery of high utility itemsets, *Proceedings of the 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Hanoi, Vietnam, 2005, pp. 689-695.

[9] M. Liu, J. Qu, Mining high utility itemsets without candidate generation, *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, Maui, Hawaii, USA, 2012, pp. 55-64.

[10] P. Fournier-Viger, C. W. Wu, S. Zida, V. S. Tseng, FHM: Faster high-utility itemset mining using estimated utility co-occurrence pruning, *Proceedings of the 21st International Symposium on Methodologies for Intelligent Systems*, Roskilde, Denmark, 2014, pp. 83-92.

[11] W. Gan, S. Wan, J. Chen, C. M. Chen, L. Qiu, TopHUI: Top-k high-utility itemset mining with negative utility, *Proceedings of the 8th IEEE International Conference on Big Data*, Atlanta, GA, USA, 2020, pp. 5350-5359.

[12] Q. H. Duong, P. Fournier-Viger, H. Ramampiaro, K. Nørvåg, T. L. Dam, Efficient high utility itemset mining using buffered utility-lists, *Applied Intelligence*, Vol. 48, No. 7, pp. 1859-1877, July, 2018.

[13] L. A. Zadeh, Fuzzy Sets, *Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems*, World Scientific Publishing Company, 1996, pp. 394-432.

[14] C. M. Wang, S. H. Chen, Y. F. Huang, A fuzzy approach for mining high utility quantitative itemsets, *Proceedings of the 18th IEEE International Conference on Fuzzy Systems*, Jeju, Korea (South), 2009, pp. 1909-1913.

[15] G. C. Lan, T. P. Hong, Y. H. Lin, S. L. Wang, Fuzzy utility mining with upper-bound measure, *Applied Soft Computing*, Vol. 30, pp. 767-777, May, 2015.

[16] S. Wan, W. Gan, X. Guo, J. Chen, U. Yun, FUIM: Fuzzy utility itemset mining, October, 2021. https://arxiv.org/abs/2111.00307

[17] C. C. Aggarwal, M. A. Bhuiyan, M. A. Hasan, Frequent pattern mining algorithms: A survey, in: C. Aggarwal, J. Han (Eds.), *Frequent Pattern Mining, Springer, Cham*, 2014, pp. 19-64.

[18] M. J. Zaki, Scalable algorithms for association mining, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 12, No. 3, pp. 372-390, June, 2000.

[19] V. S. Tseng, C. W. Wu, B. E. Shie, P. S. Yu, UP-Growth: an efficient algorithm for high utility itemset mining, *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington, DC, USA, 2010, pp. 253-262.

[20] C. F. Ahmed, S. K. Tanbeer, B. S. Jeong, Y. K. Lee, Efficient tree structures for high utility pattern mining in incremental databases, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 21, No. 12, pp. 1708-1721, December, 2009.

[21] V. S. Tseng, B. E. Shie, C. W. Wu, P. S. Yu, Efficient algorithms for mining high utility itemsets from transactional databases, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 25, No. 8, pp. 1772-1786, August, 2013.

[22] J. C. W. Lin, T. Li, P. Fournier-Viger, J. Zhang, X. Guo, Mining of high average-utility patterns with item-level thresholds, *Journal of Internet Technology*, Vol. 20, No. 1, pp. 187-194, January, 2019.

[23] S. Wan, J. Chen, Z. Qi, W. Gan, L. Tang, Fast RFM model for customer segmentation, *Companion Proceedings of the 34th Web Conference*, Lyon, France, 2022, pp. 965-972.

[24] S. Wan, J. Chen, P. Zhang, W. Gan, T. Gu, Discovering top-k profitable patterns for smart manufacturing, *Companion Proceedings of the 34th Web Conference*, Lyon, France, 2022, pp. 956-964.

[25] G. C. Lan, T. P. Hong, Y. H. Lin, S. L. Wang, Fast discovery of high fuzzy utility itemsets, *Proceedings of the 27th IEEE International Conference on Systems, Man, and Cybernetics*, San Diego, CA, USA, 2014, pp.

2764-2767.
[26] W. M. Huang, T. P. Hong, G. C. Lan, M. C. Chiang, J. C. W. Lin, Temporal-based fuzzy utility mining, *IEEE Access*, Vol. 5, pp. 26639-26652, November, 2017.
[27] T. P. Hong, C. Y. Lin, W. M. Huang, K. S. M. Li, L. S. L. Wang, J. C. W. Lin, Using tree structure to mine high temporal fuzzy utility itemsets, *IEEE Access*, Vol. 8, pp. 153692-153706, August, 2020.
[28] T. P. Hong, C. Y. Lin, W. M. Huang, S. M. Li, S. L. Wang, J. C. W. Lin, A one-phase tree-structure method to mine high temporal fuzzy utility itemsets, *Applied Sciences*, Vol. 12, No. 6, Article No. 2821, March, 2022.
[29] Q. H. Duong, B. Liao, P. Fournier-Viger, T. L. Dam, An efficient algorithm for mining the top-k high utility itemsets, using novel threshold raising and pruning strategies, *Knowledge-Based Systems*, Vol. 104, pp. 106-122, July, 2016.

Biographies

Guotao Xu is an undergraduate student at Guangdong University of Technology, Guangzhou, China, majoring in information security. His research interests include data mining and privacy preservation.

Jiahui Chen (Member, IEEE) received the MS and PhD degrees from South China University of Technology, China. He joined National University of Singapore as a research scientist for one year. He is currently an associate professor in Guangdong University of Technology. His research interests include post-quantum cryptography and information security.

Shicheng Wan received the B.S. degree in Gnanan Normal University, Ganzhou, China in 2020. He is currently a postgraduate with the Department of Computer Sciences, Guangdong Technology University, Guangdong, China. His research interests include data mining, utility mining, and big data.

Cuiwei Peng, is an undergraduate student at Computer Science and Technology from Guangdong University of Technology, Guangzhou, China. Her research interests include big data and data mining.

Yu Liu is an undergraduate student at Guangdong University of Technology, Guangzhou, China, majoring in information security. His research interests include data mining, blockchain technology, and blockchain consensus mechanisms.