A Single-Stage Tree-Structure-Based Approach to Determine Fuzzy Average-Utility Itemsets

Tzung-Pei Hong[1,2](✉), Meng-Ping Ku[2], Hsiu-Wei Chiu[1], Wei-Ming Huang[3], Shu-Min Li[2], and Jerry Chun-Wei Lin[4]

[1] National University of Kaohsiung, Kaohsiung 811, Taiwan
tphong@nuk.edu.tw, fuvu7620250@gmail.com
[2] National Sun Yat-Sen University, Kaohsiung 804, Taiwan
peter0617ku@gamil.com, smli@cse.nsysu.edu.tw
[3] China Steel, Inc., Kaohsiung 806, Taiwan
granthill168@gmail.com
[4] Western Norway University of Applied Science, 5063 Bergen, Norway
jerrylin@ieee.org

Abstract. Fuzzy utility mining (FUM) techniques are used to mine high fuzzy utility itemsets for market analysis. They consider items that include purchase quantities, unit profits, and linguistic terms representing quantity information. Although FUM facilitates market analysis, it has the measurement problem in which the fuzzy utility value for an itemset may be higher than that for its subset. In the past, a tree-based mining method was proposed to find fuzzy average-utility itemsets using a two-stage strategy tree-based method with an average-utility measure. It was, however, computationally expensive because two-stage processing was needed. To handle this, we propose a single-stage tree-structure-based method that uses an external list for each node in the tree to find fuzzy average-utility itemsets efficiently. Experimental results show that the proposed method outperforms the former approach in terms of execution time.

Keywords: FP-growth · Fuzzy theory · Fuzzy average-utility mining · Tree structure

1 Introduction

Pattern mining is an active subfield of data mining used to find interesting knowledge patterns in a large database, where mined rules are used for decision support. The Apriori algorithm [1, 2] considers item frequencies in a binary database, but the number of items sold or their importance is ignored. Utility mining (UM) was thus proposed [3], where items in a database include the purchase quantities and relative importance indicating unit profits or weights. Its goal is to find high-utility itemsets, which indicate potential importance; however, the downward closure (DC) property does not hold in UM. Two-stage mining is used to improve mining efficiency [4]. In the UM mining process, larger itemsets in a transaction tend to have a greater utility value than that of their sub-itemsets.

Hence using the same threshold to evaluate itemsets, regardless of their length, is an unfair strategy. The average-utility mining algorithm is used to normalize itemset lengths [6]. In contrast to UM, FUM jointly considers the characteristics of UM and fuzzy reasoning to identify high fuzzy utility itemsets and handle quantity information better via its transformed linguistic terms [5]. However, as with UM, the DC property does not hold for FUM either. Thus, two-stage [9] and two-stage tree-structure [8] based approaches were proposed to find desired itemsets using an average-utility measurement. However, their two-stage nature makes these methods computationally expensive. To efficiently extract fuzzy average-utility itemsets, a single-stage tree structure method based on FP-tree [7] is proposed, in which each node in the tree has an external array list to store mined information. Experiments demonstrate improved execution times with respect to [8]. However, adding list information for each node also increases memory consumption compared to [8].

2 Related Work

The frequent itemset mining, named Apriori [1, 2], is used to find knowledge patterns in which their frequency counting is executed by scanning multiple databases. To account for the performance, the FP-Growth [7] is then proposed by applying tree structure to store mined information, reducing the database scans. The Apriori is useful, but it does not take into account item quantities or unit profits for items. To overcome this, relative importance based on profit and item quantity is considered using utility mining [3]. The utility values of itemsets are used to evaluate whether they are useful. One phenomenon of UM is that since the utility value of an itemset in a transaction may be larger than those of its subsets, it is unfair to use the same threshold to determine different itemsets. Average-utility measurement accounts for those [6, 11–13]. FUM [5] is superior to UM in that it efficiently explains quantitative information. By using the membership function of items, item quantities are transformed into fuzzy terms where they possess semantic meaning in item amounts. The FUM process derives actual itemsets with their fuzzy utility values satisfying the threshold along with the quantitative values of items, profits, and semantic meaning in item amounts. However, FUM shares the limitation of UM: the actual value for a larger itemset may be higher than that of a smaller itemset. Two fuzzy average-utility methods for FUM have thus been proposed [8, 9]. An over-estimation model is used to avoid information loss and a two-stage algorithm with this model is designed for efficient mining [9]. To improve the efficiency in [9], Hong et al. consider a two-stage tree-based method [8]. Here, we propose an alternative with shorter execution times than [8]. An external list containing mined information is embedded within each node in the tree, performing for single-stage operation.

3 Definition

Let D be a transaction database, and the items in D are represented as $I = \{i_1, i_2, ..., i_Q\}$, where each item i_n has its own profit, denoted as $p(i_n)$. The database is the set of transactions denoted as $D = \{t_1, t_2, ..., t_P\}$. Each t_m in D contains purchased item i_n with quantities v_{mn}. A set of membership functions (*MFs*) is given in advance, which represents the membership degree of each item. Given the *MF* for an item, each quantity value v_{mn} in D is converted into a fuzzy set $f_{mn} = (\frac{f_{mn1}}{R_{n1}} + \frac{f_{mn2}}{R_{n2}} + \cdots + \frac{f_{mnl}}{R_{nl}} + \frac{f_{mnh}}{R_{nh}})$, where h is the number of membership functions for i_n, R_{nl} is the l-th fuzzy term of i_n, and f_{mnl} is the fuzzy membership value of v_{mn} in R_{nl}.

Shown in Table 1 is a transaction database that contains the items and the item quantities for each transaction. Table 2 is the utility table, which records the unit profit of each item. The membership functions are shown in Fig. 1, where we assume that the *MFs* of all the items are the same. We use the *MFs* to divide the quantities into fuzzy regions *L*, *M*, and *H*. The above information is used as an example of the definition.

Table 1. A transaction database

Transaction	(Item, quantity)
t_1	(*A*, 2), (*B*, 6), (*C*, 2), (*D*, 6)
t_2	(*A*, 4), (*B*, 5), (*C*, 5), (*D*, 4)
t_3	(*B*, 1), (*C*, 8), (*D*, 4)

Table 2. A utility table

Item	Profit
A	5
B	6
C	2
D	4

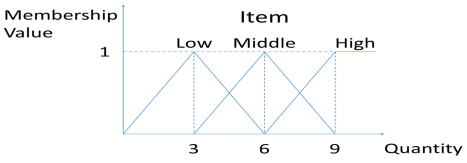

Fig. 1. Membership functions

Definition 1. The fuzzy average utility of a fuzzy item R_{nl} in i_n in t_m is $fau_{mnl} = f_{mnl} * v_{mn} * p(i_n)$, where v_{mn} is the quantity of i_n in t_m, f_{mnl} is the fuzzy value of R_{nl} according to the *MF* of i_n, and $p(i_n)$ is the individual profit for i_n. According to the *MF* in Fig. 1, {*A*} with quantity 4 in t_2 in Table 1 is converted to {0.33/*A.L*, 0.67/*A.M*}, yielding a *fau* value of 0.67 * 4 * 5 (= 13.4) for {*A.M*}. All fuzzy items from Table 1 are calculated and shown in Table 3.

Definition 2. The fuzzy average utility of each fuzzy itemset S in t_m is $fau_{mS} = \frac{1}{|S|} * f_{mS} * \sum_{R_{nl} \in S} \left[v_{mn} * p(i_n) \right]$, where |*S*| is the number of R_{nl} and f_{mS} is the minimum fuzzy value for R_{nl}, where $R_{nl} \in S$. Take $\{A.L, B.M\}$ in t_1 as an example. According to the *MF* in Fig. 1, its integrated fuzzy value is $min\{0.67, 1\}$, which is 0.67. Thus, its $fau_{1,\{A.L, B.M\}}$ is $\frac{1}{2} * 0.67 * (2 * 5 + 6 * 6) = 15.41$.

Table 3. Fuzzy average utility values

Tid.	*(Fuzzy item, fuzzy average utility value)*	$mtfau_m$
t_1	(*A.L*, 6.67), (*B.M*, 36), (*C.L*, 2.67), (*D.M*, 24)	36
t2	(*A.L*, 13.33),(*A.M*, 6.67),(*B.L*, 10),(*B.M*, 20),(*C.L*, 3.33),(*C.M*, 6.67), (*D.L*, 1.33)	20
t3	(*B.L*, 2), (*C.M*, 5.33), (*C.H*, 10.67), (*D.L*, 10.67), (*D.M*, 5.33)	10.67

Definition 3. The actual fuzzy average utility of each fuzzy itemset *S* in *D* is expressed as $afau_S = \sum_{S \subseteq t_m \cap t_m \in D} fau_{mS}$. For example, the $afau_{\{A.L, B.M\}}$ in *D* is $fau_{1,\{A.L, B.M\}} + fau_{2,\{A.L, B.M\}} = 0.5 * 0.67 * (2 * 5 + 6 * 6) + 0.5 * 0.67 * (4 * 5 + 5 * 6) = 32.16$.

Definition 4. Let *MinFAUtil* be the given threshold. A fuzzy itemset *S* is considered a high fuzzy average-utility itemset *HFAUI* and $afau_S \geq MinFAUtil$ holds. Let *MinFAUtil* = 30. Since the $afau_{\{A.L, B.M\}}$ is 32.16, $\{A.L, B.M\}$ is an *HFAUI*. However, $afau_{\{A.L\}}$ is $0.67 * 10 + 0.67 * 20 = 20.1$, so $\{A.L\}$ is not a *HFAUI*, because the DC in fuzzy average-utility mining does not hold.

To take this into account, we use the over-estimation model [8] for fuzzy average-utility mining. The definitions for this model are given below.

Definition 5. The maximum fuzzy average utility of an item i_n in t_m is $mfau_{mn} = \max_{R_{nl} \subseteq i_n \cap i_n \in t_m} \{fau_{mn1}, fau_{mn2}, \ldots, fau_{mnh}\}$. For example, the $mfau_A$ in t_2 is 13.33.

Definition 6. The maximum transaction fuzzy average utility in t_m is $mtfau_m = \max_{i_n \subseteq t_m} mfau_{mn}$. For example, the $mtfau_2$ is 20.

Definition 7. The fuzzy average-utility upper bound of a fuzzy itemset *S* is $fauub_S = \sum_{S \subseteq t_m \cap t_m \in D} mtfau_m$. Since $\{A.L\}$ exists in t_1 and t_2, its $fauub_{\{A.L\}}$ is 56.

Definition 8. The fuzzy itemset *S* is considered the high fuzzy average-utility upper-bound itemset *HFAUUBI* and $fauub_S \geq MinFAUtil$ holds. For example, $fauub_{\{A.L\}}$ is 56, which is greater than *MinFAUtil*, so $\{A.L\}$ is an *HFAUUBI*.

4 Proposed FHFAUIM Algorithm

This algorithm, called Fast High Fuzzy Average-Utility Itemset Mining (FHFAUIM), enhances the performance for fuzzy average-utility mining compared to High Fuzzy Average-Utility Itemset Mining (HFAUIM) [8]. An external list that stores fuzzy item's transaction ID, fuzzy value, and utility value is added to the tree node. Thus, the mined process can be performed directly in a single phase. Below we list the steps of the algorithm:

Step 1. Based on the *MFs* for all items, convert the quantities in *D* into a fuzzy set.

Step 2. Calculate the *mfau* value of each item in each transaction.

Step 3. Find the *mtfau* value of each transaction.

Step 4. Initialize the candidate 1-table ($HFAUUBI_1$) table into an empty table with three attributes: fuzzy itemset *S*, its $fauub_S$ value, and its frequency.

Step 5. Store fuzzy items in *D* into the $HFAUUBI_1$ table and get the $fauub_S$ for each.

Step 6. Filter each fuzzy itemset *S* in the $HFAUUBI_1$ table: if $fauub_S$ is not less than *MinFAUtil*, keep it in the table; otherwise, remove it.

Step 7. Calculate the frequency of the fuzzy items in the $HFAUUBI_1$ table. Sort all fuzzy items in the table by decreasing frequency; this is the header table.

Step 8. Trim fuzzy items in *D* that do not appear in the $HFAUUBI_1$ table as *UD*.

Step 9. Build a tree structure similar to an FP-tree. Each node in the tree stores a fuzzy item and its *mtfau* value. In addition, each node contains an external list that stores the identifier transaction, the fuzzy value of the fuzzy item, and its utility value. According to the *UD*, each fuzzy item in a transaction is inserted into the tree structure from the first transaction to the end, one by one.

Step 10. The $HFAUUBI_1$ table is considered the header table. All fuzzy items in the $HFAUUBI_1$ table are directed to the nodes of the tree's corresponding fuzzy items.

Step 11. After completing the tree structure, find *HFAUI*s. First, each fuzzy item in the $HFAUUBI_1$ table is used to establish its conditional FP-tree by traversing the tree from the bottom up. After going through the conditional FP-tree with each fuzzy item's node, the *afau* values of the fuzzy itemsets are calculated using the nodes' external lists for fuzzy itemsets. If their $afau \geq MinFAUtil$, they are considered *HFAUI*s.

Step 12. Output all *HFAUI*s.

5 Experiments

We compared the previous *HFAUIM* [8] with the proposed *FHFAUIM* on the test datasets, T25I2N1KD10K and T24I2N1KD10K [10]. Two methods were implemented in Java, and experiments were conducted on a computer with an Intel CPU at 3.00GHz and 8GB of RAM. Various thresholds were used to evaluate the performance of the two methods, with the execution time and the memory consumption results shown in Figs. 2 and 3. Moreover, to evaluate the execution time, the *FHFAUIM* uses a single-stage strategy to reduce the number of candidates generated compared to *HFAUIM*. Execution times decrease as *MinFAUtil* is increased. Also, when $MinFAUtil = 0.01$, the single-stage strategy in *FHFAUIM* generally yields significantly reduced computation times in comparison with *HFAUIM*. Therefore, the maximum efficiency improvement rate of execution time is 95.39%. In memory strategy, given different thresholds: the memory usage of *HFAUIM* is less than that of *FHFAUIM*, because *FHFAUIM* accelerates the runtimes by using an external list for each node to store mined data, which requires extra memory.

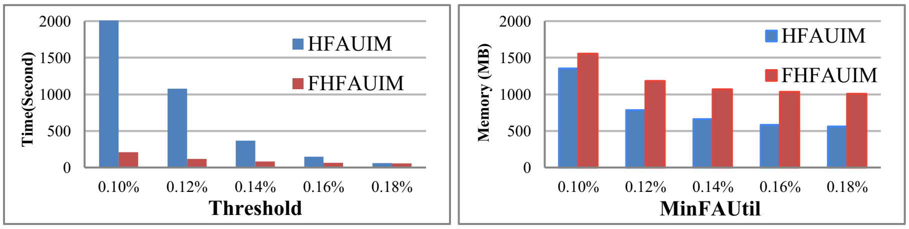

Fig. 2. Execution times and memory consumption in database T25I2N1KD10K

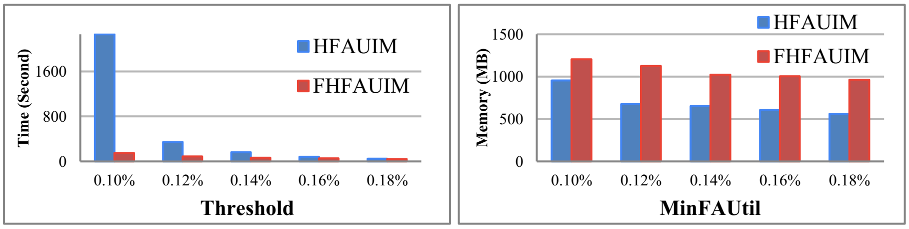

Fig. 3. Execution times and memory consumption in database T24I2N1KD10K

6 Conclusion

We propose a fast method for mining fuzzy average-utility itemsets. The proposed algorithm integrates a single-stage strategy with a tree structure to reduce the search space by storing information in node-level external lists. Experimental results show that the method requires far less computation time than the previous approach [8].

References

1. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: The ACM SIGMOD International Conference on Management of Data, pp. 207–216 (1993)
2. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: The 20th International Conference on Very Large Data Bases, pp. 487–499 (1994)
3. Yao, H., Hamilton, H., Butz, C.: A foundational approach to mining itemset utilities from databases. In: The 4th SIAM International Conference on Data Mining, pp. 211–225 (2004)
4. Liu, Y., Liao, W.-K., Choudhary, A.: A two-phase algorithm for fast discovery of high utility itemsets. In: Ho, T.B., Cheung, D., Liu, H. (eds.) PAKDD 2005. LNCS (LNAI), vol. 3518, pp. 689–695. Springer, Heidelberg (2005). https://doi.org/10.1007/11430919_79

5. Lan, G.C., Hong, T.P., Lin, Y.H., Wang, S.L.: Fuzzy utility mining with upper-bound measure. Appl. Soft Comput. **30**, 767–777 (2015)
6. Hong, T.P., Lee, C.H., Wang, S.L.: Effective utility mining with the measure of average-utility. Expert Syst. Appl. **38**(7), 8259–8265 (2011)
7. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: The ACM SIGMOD International Conference on Management of Data, vol. 29, pp. 1–12 (2000)
8. Hong, T.P., Ku, M.P., Huang, W.M., Li, S.M., Lin, J.C.W.: A tree-based fuzzy average-utility mining algorithm. In: IEEE International Conference on Data Mining (2020)
9. Hong, T.P., Ku, M.P., Huang, W.M., Li, S.M., Lin, J.C.W.: Mining high fuzzy average-utility itemsets. In: The International Conference on System Science and Engineering (2020)
10. IBM Quest Data Mining Projection, Quest synthetic data generation code (1996). http://www.almaden.ibm.com/cs/quest/syndata.htm
11. Wu, J.M.-T., Teng, Q., Lin, J.C.-W., Fournier-Viger, P., Cheng, C.-F.: Maintenance of prelarge high average-utility patterns in incremental databases. In: Fujita, H., Fournier-Viger, P., Ali, M., Sasaki, J. (eds.) IEA/AIE 2020. LNCS (LNAI), vol. 12144, pp. 884–895. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-55789-8_75
12. Truong, T.C., Duong, H.V., Le, B., Fournier-Viger, P.: Efficient high average-utility itemset mining using novel vertical weak upper-bounds. Knowl. Based Sys. **183**, (2019). https://doi.org/10.1016/j.knosys.2019.07.018
13. Yildirim, I., Celik, M.: An efficient tree-based algorithm for mining high average-utility itemsets. IEEE ACCESS **7**, 144245–144263 (2019). https://doi.org/10.1109/ACCESS.2019.2945840