Available online at www.elixirpublishers.com (Elixir International Journal)

Information Technology

Elixir Inform. Tech. 108 (2017) 47570-47575

ISSN: 2229-712X

Mining High Utility Itemsets Using FHUI From Large Database

D.Sathyavani[1] and D.Sharmila[2]

[1] Computer Sciences and Engineering, United Institute of Technology, Coimbatore, Tamil Nadu, India.

[2]Bannari Amman Institute of Technology, Sathyamangalam, Tamil Nadu, India.

ARTICLE INFO

Article history:

Received: 26April 2017;

Received in revised form:

30 June 2017;

Accepted: 8 July 2017;

Keywords

High utility itemset,

Mining algorithm,

Fuzzy Logic.

ABSTRACT

Utility mining is to discover the itemsets in a transaction database with utility values over a given threshold. Utility of itemsets are based upon user's perception such as cost, profit or revenue and are of significant importance. Utility-based data mining intends discovering the itemsets with high total utility is called High Utility Itemset mining. High Utility itemsets may contain frequent as well as rare itemsets. Classical utility mining considers items alone as discrete values. In real world applications, such utilities can be described by fuzzy sets. In this paper, proposed an algorithm, FHUI (Fuzzy High Utility Itemset) Mining is presented to mine high utility itemsets effectively from large databases, by fuzzification of utility values. FHUI extracts fuzzy high utility itemsets by integrating fuzzy logic with high utility itemset mining. This algorithm finds all utility-frequent itemsets within given threshold value. The experimental result shows that, it performs efficiently in terms of speed and memory on large databases.

1. Introduction

Data mining is a technique which extracting the data's from the large database. Association Rule Mining is used to find the interestingness relationship among the items and generates the association rules. The database DB consists of a set of transactions. Knowledge Discovery in Databases (KDD) is the non-trivial process of identifying valid, previously unknown and potentially useful patterns in data. These patterns are used to make predictions or classifications about new data and explains existing data, summarize the contents of a large database.

The objective of the first step of association rule mining is to find all frequent itemsets and the the second phase is to generate the rules based on frequent itemsets. One of the important tasks in data mining is utility mining which refers to the discovery of more profitable item. High utility mining mines the high utility itemset from the transaction database. When the utility of an item is greater than or equal to user specified minimum utility threshold, then it is a high utility item.

For example, assume the frequency of item A is 7, item B is 6, and itemset AB is 3. The profit of item A is 2, B is 5. The utility value of A is 7 * 2 = 14, B is 6* 5= 30, and AB is 3 * 2 + 3 * 5 = 21. If the minimum utility threshold is 25, B is a high utility itemset a number of algorithms have been proposed for high utility itemset mining namely two phase [1], TWU mining [2], UMMI algorithm [3].

However, frequency of an itemset alone does not guarantee of its interestingness. Because it does not contain information on its itemset of specified utility such as profit. User-defined utility is based on information not available in the transaction dataset. It often reflects user preference and can be represented by an external utility table or utility function. Utility-based data mining is a wide area that covers all aspects of economic utility in data mining.

2. Related Work

Utility Mining covers all aspects of "economic" utilities – utilities that affect a business, and helps in detecting rare high utility itemsets. High Utility Rare Itemset Mining (HURI) is very beneficial in several real-life applications. In [7], Jyothi et al presented a literature survey of the various approaches and algorithms for high-utility mining and rare itemset mining. Ashish et al presented a fast and efficient fuzzy ARM algorithm on very large datasets. The algorithm was 8 to 19 times faster than traditional fuzzy ARM on very large standard real-life datasets. In [2], unlike most two-phased ARM algorithms, the authors presented individual itemset processing as opposed to simultaneous itemset processing at each k-level, recording some performance improvements.

In [18], C. Saravanabhavan et al presented an efficient tree structure for mining of high utility itemsets. Firstly, the authors developed a utility frequent-pattern tree structure to store important information about utility itemsets. Next the pattern growth methodology was used to mine the entire utility pattern sets. Two algorithms, UP-Growth (Utility Pattern Growth) and UP-Tree (Utility Pattern Tree) are proposed in [14] for mining high utility itemsets. Also a set of effective strategies are discussed by Sadak Murali et al, for pruning candidate itemsets.

In [1], Adinarayanareddy B presented a modified UP-Growth (IUPG) algorithm for high utility itemset mining. The authors conclude that IUPG algorithm performs better than UP-Growth algorithm for different support values and also IUPG algorithm is highly scalable. [13] Ruchi Patel proposed a parallel and distributed method for mining high utility patterns from large databases.

The method also prunes the low utility itemsets from transactions at initial level by using downward closure property.

Tele:

E-mail address: sathyavani.it@gmail.com

Koh et al proposed a modified Apriori inverse algorithm to generate rare itemsets of user interest [11]. Yao et al defined Utility as a measure usefulness or profitability of an itemset [15] [16]. The authors focused on the measures used for utility-based itemset mining. Utility based measures use the utilities of the patterns to reflect the user's goals. The authors formalize the semantic significance of utility measures and classify existing measures into one of three categories: item level, transaction level and cell level. A unified framework was proposed for incorporating utility based measures into the data mining process via a unified utility function.

One of the most essential areas of the application of fuzzy set theory is Fuzzy rule-based systems [4]. These knowledge extraction tools discover intrinsic associations contained in a data base. Fuzzy systems improve the interpretation and understandability of consumer models. In [4], Casillas et al presented a new approach for consumer behavior modeling which is based on fuzzy association rules (FAS). A behavioral model was presented which centered on consumer attitude towards Internet and confidence in Internet shopping.

The contributions of this paper are summarized as Follows:

(1) We propose a new tree structure named FTN-Tree (Fuzzy based Tail node Tree) for maintaining important information related to a transaction dataset.

(2) We also give an algorithm named FT-Mine for FIM over uncertain transaction datasets based on FT-Tree.

(3) Propose an algorithm that uses TWU with pattern Growth based on a compact utility pattern tree data structure. Our algorithm implements a fuzzy tail node tree scheme to use disk storage when the main memory is inadequate for dealing with large datasets.

(4) Fuzzy tail node tree based mining the high utility itemsets with minimum no of tree creation.

3. Terms and Definitions

Let $D = \{T_1, T_2 \ldots T_n\}$ be an transaction dataset which contains n transaction itemsets and m distinct items, i.e. $I = \{i_1, i_2 \ldots i_m\}$. Each transaction itemset is represented as $\{i_1{:}p_1, i_2{:}p_2 \ldots i_v{:}p_v\}$, where $\{i_1, i_2, \ldots, i_v\}$ is a subset of I, and pu $(1 \le u \le v)$ is the existential probability of item i_u in a transaction itemset. The size of dataset D is the number of transaction itemsets and is denoted as $|D|$. An itemset $X = \{i_1, i_2 \ldots i_k\}$, which contains k distinct items, is called a k-itemset, and k is the length of the itemset X.

Definition 1: The *support number* (*SN*) of an itemset X in a transaction dataset is defined by the number of transaction itemsets containing X.

Definition 2: The *expected support number (expSN*) of an itemset X in an transaction dataset is denoted as *expSN*(X) and is defined by

$$expSN(X) = \sum_{T_d \supseteq X, T_d \in D} P(X, T_d)$$

Definition 3: Given a dataset D, the *minimum expected support threshold* η is a predefined percentage of $|D|$; correspondingly, the *minimum expected support number* (*minExpSN*) is defined by $minExpSN = |D| \times \eta$ an itemset X is called a utility itemset if its *expected support number* is not less than the value *minExpSN*.

Definition 4: The *minimum support threshold* λ is a predefined percentage of $|D|$; correspondingly, the *minimum support number* (*minSN*) in a dataset D is defined by

$$minSN = |D| \times \eta$$

Definition 5: Fuzzy utility in high utility itemset mining, item utility is equal to quantity value multiplied by profit. In fuzzy transaction, the similar procedure to find item utility will produce false results. For example, item A occurs in different transaction T1 and T5. If the fuzzy set of (A, T1) = {1/L, 0/M, 1/H} and (A, T5) = {0/L, 0/M, 1/H}. If t heir item utilities are obtained by multiplying fuzzy quantity with profit, item utilities of T1 and T5 will be same even though T5 yields a higher quantity value. The fuzzy utility is defined as,

$$fu(i_p, T_q) - f(i_p, T_q) \times S(i_p)$$

Definition 6: Fuzzy quantity The equation to find the fuzzy quantity value of item i_p in transaction T_q is denoted as f (i_p, T_q) which is defined as,

$$f(i_p, T_q) - \sum fq(i_p, T_q) \times weight(j)$$

Where fq $(i_p, T_q,$ j) is the fuzzy value of fuzzy region j and weight (j) is a variable parameter defined by the fuzzy region. If a fuzzy region is low then the weight should be low when compared to the region middle and high. Special weights are assigned for region low, medium and high.

Definition 7: Fuzzy transaction utility can be defined as the sum of the fuzzy utilities of item occurring in the particular transaction. The equation denoting fuzzy transaction utility is

$$ftu(T_q) - \sum_{i_p \in T_q} fu(i_p, T_q)$$

Definition 8: Fuzzy transaction weighted utility it is the sum of fuzzy transaction utilities of item occurring in the particular transaction for an item. The equation denoting fuzzy transaction weighted utility can be defined as,

$$ftwu(x) - \sum_{xcT_q \in D} ftu(T_q)$$

4. Proposed Approach

A novel method namely FTNT-HUI (Fuzzy TNT Tree based high utility itemsets) mine is proposed which can reflect the quantity information and also profit information. In this algorithm, itemsets with profit slightly less than the designated threshold value is also included resulting in new revenue opportunities. FTNT-HUI-Mine consists of two Fuzzy high utility itemset mining algorithm is proposed to provide more information concerning the quantity and profit information of high utility itemset. FTNT-Mine algorithm is intended for mining fuzzy high utility itemset by applying fuzzy theory to high utility itemset. a fuzzy membership function is defined to represent the quantities in fuzzy sets. Thus the transaction table is transformed into fuzzy transaction table. Then fuzzy transaction utility will be calculated from the fuzzy transaction table and utility table. Further, Fuzzy transaction weighted utility is produced to discover the phase I high utility itemset. Phase II calculates the fuzzy utility for the phase I high utility itemset to yield the fuzzy high utility itemset

4.1 Fuzzy Transaction-Weighted Utilization Itemset

Fuzzy membership function for quantity is defined to provide the fuzzy quantity region which is shown in figure 1. It can transform a quantity value in to fuzzy membership region and membership value thereby enhancing transaction database as fuzzy transaction database. The transaction database is shown in table 1.

The quantity attribute for fuzzy membership function has three fuzzy regions namely low, middle and high. Thus, fuzzy

membership value for the purchased quantity is represented as fuzzy set in terms of {fuzzy value of low/low, fuzzy value of middle/middle, and fuzzy value of high/high}. For example, the quantity value '9' is converted in to fuzzy set as {0.0/L, 0.4/M, 0.6/H}, where 'L', 'M' and 'H' are acronym of 'Low', 'Middle' and 'High'.

Table 1. Transaction database.

	A	B	C	D	E
T01	0	3	6	1	4
T02	3	0	10	0	9
T03	7	0	4	0	0
T04	6	1	0	1	0
T05	0	0	8	0	3
T06	0	2	12	1	0
T07	9	0	0	0	7
T08	2	2	0	0	6

To find the sole representing fuzzy quantity from three fuzzy regions, a maximum value is generated from the fuzzy set. For example, the sole representation for quantity value '9' is represented as Max (0.0, 0.4, 0.6) =0.6. The sole representation for the quantity value was done according to equation (2). Weight parameter was fixed for the three region as Low=0.1, Middle=0.5, High=1. According to definition (7), sole representation for quantity '9' is 0. 6. The calculation is preceded as follows. The max value in fuzzy set is 0.6 which is present in region high. The max value is multiplied with weight assigned for region high to yield the sole value for particular quantity. Hence this procedure is adopted for all quantity in transaction database to reproduce fuzzy transaction database which is shown in table 2.

Table 2. Fuzzy Transaction database.

	A	B	C	D	E
T01	0	0.06	0.5	0.1	0.3
T02	0.06	0	0.8	0	0.6
T03	0.4	0	0.3	0	0
T04	0.5	0.1	0	0.1	0
T05	0	0	0.3	0	0.06
T06	0	0.08	0.9	0.1	0
T07	0.6	0	0	0	0.4
T08	0.08	0.08	0	0	0.5

Table 3. Utility Table.

Item	A	B	C	D	E
Utility	5	11	3	20	4

After the transformation of fuzzy transaction database from transaction database, the next step is to find the fuzzy utility for each and every transaction according to equation by multiplying fuzzy quantity with profit value in utility table shown I n table 3. Fuzzy utility for the item 'B' in transaction T01 is defined as the quantity value of item 'B' is 0.06 which is multiplied by the profit value for 'B' in the utility table which is 11 to yield the fuzzy utility as 0.66. Fuzzy utility values for the corresponding transaction databases are displayed in table 4.Fuzzy utility calculated was used to find fuzzy transaction utility Fuzzy according to equation (4) for each and every transaction. Fuzzy transaction utility table is displayed in table 5.

Table 4. Utility table.

	A	B	C	D	E
T01	0	0.06	1.5	2	1.2
T02	0.3	0	2.4	0	2.4
T03	2	0	0.9	0	0
T04	2.5	1.1	0	2	0
T05	0	0	0.9	0	0.24
T06	0	0.88	2.7	2	0
T07	3	0	0	0	1.6
T08	0.4	0.88	0	0	2

Table 5. Fuzzy transaction utility.

Transaction ID	Transaction utility
T01	5.36
T02	5.1
T03	2.9
T04	5.6
T05	1.14
T06	5.58
T07	4.6
T08	3.28

Fuzzy transaction weighted utility itemset was then found from fuzzy utility. Fuzzy TWU for item 'A' has been evaluated.

4.2 Constructing a Global FT-Tree

The proposed algorithm FT-Mine mainly consists of Two procedures: (1) creating an FT-Tree; (2) mining utility itemsets from the FT-Tree. The structure of FT-Tree is designed to efficiently store the related information on tail nodes. It is constructed by two scans of dataset. In the first scan, a header table is created to maintain sorted high utility items.

A global FT-Tree is the first FT-Tree that maintainsitemset information of the whole dataset. The construction algorithm is described as follows:

CreateTree(*D*, η)

INPUT: An uncertain database *D* consisting of *n* transaction itemsets and a predefined minimum expected support threshold η .

OUTPUT: An FT-Tree *T*.

Step 1: Calculate the minimum expected support number *minExpSN*, i.e. *minExpSN* =| *D*|×η ; count the expected support number and support number of each item by one scan of dataset.

Step 2: Put those items whose expected support numbers are not less than *minExpSN* to a header table, and sort the items in the header table according to the descending order of their support numbers; finish the algorithm if the header table is null.

Step 3: Initially set the root node of the FT-Tree *T* as null.

Step 4: Remove the items that are not in the header table from each transaction itemset, and sort the remaining items of each transaction itemset according to the order of the header table, and get a sorted itemset *X*.

Step 5: If the length of itemset *X* is 0, process the next transaction itemset; otherwise insert the itemset *X* into the FT-Tree *T* by the following substeps:

Substep 5.1: Store the probability value of each item in item set *X* sequentially to a list; save the list to an array (which is denoted as ***ProArr***); the corresponding sequence number of the list in the array is denoted as *ID*.

Substep 5.2: If there has not been a tail node for the itemset *X*, the length of itemset *X*, and *N.Tail_info.Arr_ind*={*ID*}; otherwise, append the sequence number *ID* to *N.Tail_info.Arr_ind*.
Step 6: Process the next transaction itemset.

4.3 Mining Frequent utility Itemsets from a Global FTNT-Tree

After an AT-Tree is constructed, the algorithm ATMine can directly mine frequent itemsets from the tree without additional scan of dataset. The details of the mining approach are described below. The algorithm AT-Mine is similar to the algorithm FP-Growth: it creates and processes sub trees (prefix trees or conditional trees) recursively. But the condition of generating frequent itemsets is different from FPGrowth. **Mining (***T*, *H*, *minExpSN***)**
INPUT: An AT-Tree *T*, a header table *H*, and a minimum expected support number *minExpSN*.
OUTPUT: The frequent itemsets (FIs).
Step 1: Process the items in the header table one by one from the last item by the following steps (denote the currently processed item as *Z*).
Step 2: Append item *Z* to the current *base-itemset* (which is initialized as null); each new *base-itemset* is a frequent itemset.
Step 3: Let *Z.links* in the header table *H* contain *k* nodes whose item name is *Z*; we denote these *k* nodes as *N*1, *N*2, …, *Nk*; because item *Z* is the last one in the header table, all these *k* nodes are *tail nodes*, i.e., each of these nodes contains a *Tail_info*.
Substep 3.1: Create a sub header table *subH* by scanning the *k* branches from these *k* nodes to the root.
Substep 3.2: If the sub header table is null, go to Step 4.
Substep 3.3: Create sub AT-Tree
subTree = **CreateSubTree**(*Z.link*, *subH*).
Substep 3.4: Mining(*subTree*, *subH*, *minExpSN*).
Step 4: Remove item *Z* from the *base-itemset*.
Step 5: For each of these *k* nodes (which we denote as *Ni*, $1 \le i \le k$), modify its *Tail_info* by the following substeps:
Substep 5.1: Alter *Ni.Tail_info.len* values:
Ni.Tail_info.len = *Ni.Tail_info.len* -1.
Substep 5.2: Move *Ni.Tail_info* to the parent of node *Ni*.
Step 6: Process the next item of the header table *H*.
Subroutine: **CreateSubTree**(*link*, *subH*)
INPUT: A list *link* which records tree nodes with the same item name, and a header table *subH*.
OUTPUT: An AT-Tree *subT*.
Step 1: Initially set the root node of the tree *subT* as null.
Step 2: Process each node in the list *link* by the following steps (denote the currently processed node as *N*).
Step 3: Get the *tail-node-itemset* of node *N* (denote it as itemset *X*).
Step 4: Remove those items that are not in the header table subH from itemset *X*, and sort the remaining items in itemset *X* according to the order of the header table *subH*.
Step 5: If the length of the sorted itemset (denoted as *k*) is 0, process the next node of the list *link*; otherwise insert the sorted itemset *X* into the AT-Tree *subT* by the following substeps:
Substep 5.1: Get the original sequential *ID* of each item of the itemset *X* in the corresponding list of *ProArr*: *item_ind* = {*d*1, *d*2, .., *d*k} (*k* is the length of of itemset *X*).
Substep 5.2: Make a copy of *N.Tail_info*; denote the copy as *nTail_info*.
Substep 5.3: Alter *nTail_info* as the following:
(1) *nTail_info.len* = *k*.
(2) *nTail_info. Item_ind* = *item_ind*.
(3) if *nTail_info.bp* is null, set *nTail_info.bp*[*j*] to be the probability of item *Z*, i.e. *ProArr*[*nTail_info.Arr_ind*[*j*]]; otherwise, set *nTail_info.bp*[*j*] to be the product of *nTail_info.bp*[*j*] and the probability of item *Z* ($1 \le j \le bp.size$; the array *ProArr* is created when the global tree is created in Substep 5.1 in Section 4.2.1).

5. Experimental Results

In this section, evaluate the performance of the proposed algorithm FTNT-Mine. UP Growth is the state-of-the-art algorithm employing the pattern-growth approach and FTNT is a new proposed algorithm. So compare FTN-Mine with the algorithms UF-Growth, UP-Mine and FTN on both types of datasets: the sparse transaction datasets and dense transaction datasets. All algorithms were written in Java programming language. The configuration of the testing platform is as follows: Windows 7 32bit operating system, 4G Memory, Intel(R) Dual-Core CPU @ 2.60 GHz.

Table 6. Dataset Characteristics.

Dataset	|D|	T	|I|	Type
T10I4d100k	300,000	33.8	1000	sparse
mushroom	8,124	23	119	dense

Table 6 shows the characteristics of 4 datasets used in our experiments. "|D|" represents the total number of transactions; "|I|" represents the total number of distinct items; "I" represents the mean length of all transaction itemsets; "SD" represents the degree of sparsity or density. The synthetic dataset *T10I4d100ks* came from the IBM Data Generator and the datasets and *mushroom* were obtained from FIMI Repository. These four datasets originally do not provide probability values for each item of each.

Table7. Comparison algorithm of using *T10I4d100k* Vs No tree created.

	No of Tree					
Threshold	0.04	0.05	0.06	0.07	0.08	0.09
Fuzzy Tail Node Tree	170	62	29	10	6	4
UF-Growth +	369	115	39	17	8	4
UP Growth	410	232	101	52	43	21

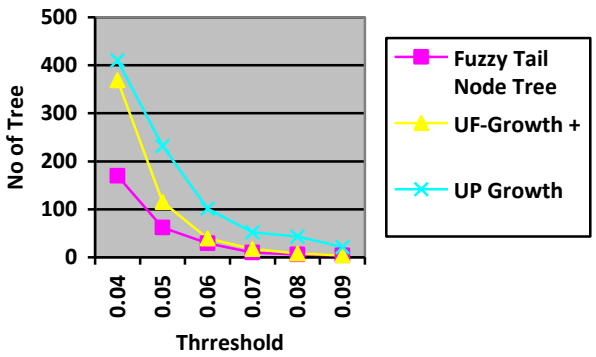

Fig 1. Comparison of different threshold value with tree creation.

Fig 1 and Table 7 Show the total number of tree nodes generated by FTNT-Mine, UF-Growth and UP, respectively, on the synthetic datasets.

Table 8. High Utility value using T10I4d100k.

	No of High Utility Itemsets					
Threshold	**0.04**	**0.05**	**0.06**	**0.07**	**0.08**	**0.09**
Fuzzy Tail Node Tree	73	44	32	23	17	11
UF-Growth +	62	33	28	19	14	8
UP Growth	48	26	22	15	9	2

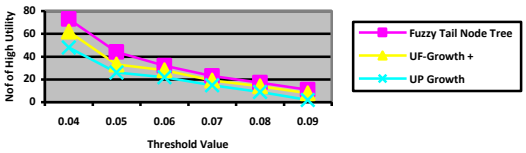

Fig 2. Comparison of different threshold value with high utility items.

Table 9. Total run time using T10I4d100k.

	No of High Utility					
Threshold	**0.04**	**0.05**	**0.06**	**0.07**	**0.08**	**0.09**
Fuzzy Tail Node Tree	11922	10209	8362	7952	7118	17897
UF-Growth +	36976	27145	23114	20317	18028	6949
UP Growth	46676	371545	33614	18317	16026	5979

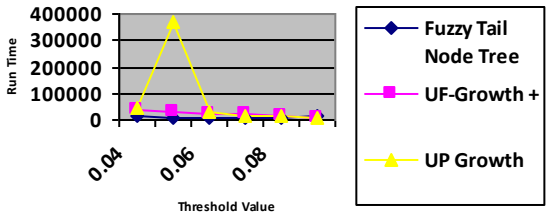

Fig 3. Run time for Different Threshold Value.

Table 10. Comparison of different algorithm using mushroom

	No of Tree					
Threshold	0.04	0.05	0.06	0.07	0.08	0.09
Fuzzy Tail Node Tree	334	254	150	123	103	83
UF-Growth +	443	345	234	189	132	112
UP Growth	546	434	367	264	212	187

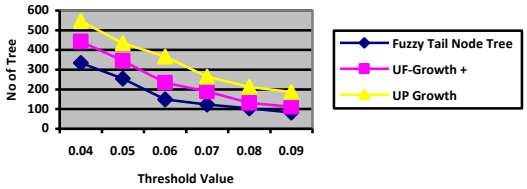

Fig 4. Comparison of different threshold value with tree creation.

Fig 4 and Table 10 Show the total number of tree nodes generated by FTNT-Mine, UF-Growth and UP, respectively, on the synthetic datasets.

Table 11. High Utility value using mushroom.

	No of High Utility					
Threshold	0.04	0.05	0.06	0.07	0.08	0.09
Fuzzy Tail Node Tree	78	46	34	27	19	13
UF-Growth +	67	38	29	21	17	10
UP Growth	49	31	28	19	10	8

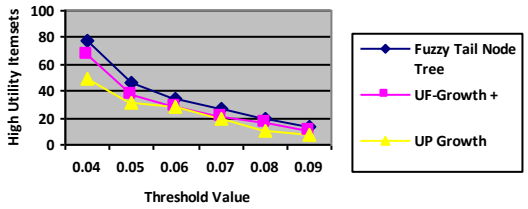

Fig 5. Comparison of different threshold value with high utility items

Table 12: Total run time using *mushroom*

	No of High Utility					
Threshold	0.04	0.05	0.06	0.07	0.08	0.09
Fuzzy Tail Node Tree	14622	12249	9372	8982	7228	6327
UF-Growth +	32956	28155	24164	21327	19048	7349
UP Growth	56576	381845	34684	19367	17126	6179

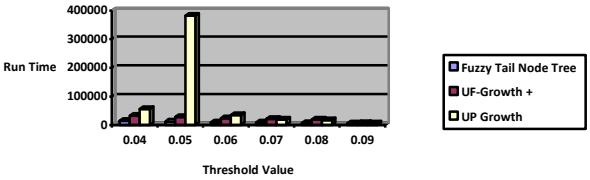

Fig 6. Run time for Different Threshold Value.

6. Conclusion

Experiments were performed on sparse and dense datasets. We compared our proposed algorithm with some state-of-the-art level-wise and pattern-growth algorithms. The experimental results show that the proposed algorithm has better performance on dense datasets and large sparse datasets, and their time performance is stable on both dense and sparse datasets along with the decreasing of the minimum expected support threshold.

References

[1]AlokChoudhary, Ying Liu, Wei- keng Liao, (2005), 'A Fast High Utility Itemsets Mining Algorithm', UBDM'05 Chicago, USA

[2]Bay VO, Huy N guyen, Bac Le (2009) 'Mining High Utility Itemset from Vertical Distributed Database', IEEE Xplore.

[3]Ming-Yen Lin, Tzer-Fu Tu, Sue-chen Hs ueh, 'High utility pattern mining using the maximal itemset property and lexicographic tree structures, Science Direct, journal of information sciences 215(2012)1-14.

[4]Tzung-Pei Hong, Kuei-Ying Lin, Shyue-Liang Wang, ' Fuzzy data mining for interesting gene ralized association rules', Elsevier on fuzzy sets and system 138(2003) 255-269.

[5]R.Agrawal, T.Imielinskl, A. Swami, 'Mining Association rules between sets of items in large databases', Proceedings of the 1993 ACM SIGMOD International Conference in Management of Data, Washington, DC , 1993, pp. 207 – 216.

[6]R.Agrawal, R.Srikant, Fast algorithms for mining association rules, in: Proceedings of 20th International Conference on Very Large Databases, Santiago, Chile, 1994, pp. 487 – 499.

[7]Alva Erwin,Raj p. Gop alan, N.R.Achunthan (2007), 'A Bottom - Up Projection Based Algorithm For mining High Utility Itemset', Workshop on Integrating AI and Data Mining (AIDM), Australia ,Vol.84.

[8]Chia-Ming Wang, Shyh-Huei Chen; Yin-Fu Huang , 'A fuzzy approach for mining high utility quantitative itemsets', Fuzzy Systems, 2009. FUZZ-IEEE 2009.

[9]T.P. Hong, C.Y. Lee, Induction of fuzzy rules and membership functions from training examples, Fuzzy Sets and Systems 84 (1996) 33 – 47.

[10]Sandeep Kumar Singh, Mr.Ganesh Wayal, Mr.Niresh sharma, 'A Review: Data Mining with Fuzzy Association Rule Mining', International Journal of Engineering Research & Technology (IJERT) Vol. 1 Issue 5, July – 2012.

[11]Stergios Papadimitriou Seferina Mavroudi, 'The Fuzzy Frequent Pattern Tree'.

[12]Karthikeyan T, Samuel Chellathurai A and Praburaj B, 'A study on a novel method of mining fuzzy association using fuzzy correlation analysis', African Journal of Mathematics and Computer Science Research Vol. 5(2), pp. 28-33, 15 January, 2012.

[13]H. Yao, H.J. Hamilton, 'Mining itemset utilities from transaction databases' , Data & Knowledge Engineering 59 (3) (2006) 603 – 626.

[14]Vincent S. Tseng and C. P. Kao, 'A Novel Similarity - based Fuzzy Clustering Algorithm by Integrating PCM and Mountain Method', IEEE Transactions on Fuzzy Systems , vol. 15, Issue 6, pp. 1188-1196.