Computing and Informatics, Vol. 31, 2012, 1597–1615

A NOVEL APPROACH TO EXTRACT HIGH UTILITY ITEMSETS FROM DISTRIBUTED DATABASES

Kannimuthu Subramanian

Department of IT
Sri Krishna College of Engineering and Technology
Coimbatore-641008, Tamil Nadu, India
e-mail: kannimuthu.me@gmail.com

Premalatha Kandhasamy

Department of CSE
Bannari Amman Institute of Technology
Sathyamangalam-638401, Tamil Nadu, India
e-mail: kpl_barath@yahoo.co.in

Shankar Subramanian

Department of IT
Sri Krishna College of Engineering and Technology
Coimbatore-641008, Tamil Nadu, India
e-mail: shanx80@gmail.com

Communicated by Patrick Brézillon

Abstract. Traditional approaches in data mining focus on support and confidence measures which are just statistics based. Support and confidence measures which are based on the frequency count of the items enable us to derive the frequent itemsets. The frequency of the items as a single factor does not represent the interestingness of the items. To enhance the process of data mining tasks based on the value of the product, several researches were conducted. It resulted in utility mining which is an emerging field of research in data mining. In the recent years various data mining approaches have been implemented in order to find the high

utility itemsets. The main objective of utility mining is to identify the itemsets with highest utilities, by considering the subjectively defined utility values, as set by the user. Existing methods based on utility mining concept focus on centralized systems where the data and associated processing is pertained to a particular location. As a further step ahead we try to implement the utility mining concept in a distributed environment. In this approach we use a sophisticated way of mining high utility itemsets using a Fast Utility Mining (FUM) algorithm.

Keywords: ARM, data mining, distributed database, FUM, HUI, FUM-D, UMining, utility mining

Mathematics Subject Classification 2010: 68R05

1 INTRODUCTION

Data mining is a process of extracting information from data which are implicit, previously unknown, and potentially useful. It is an analytical tool that allows a user to analyze the data from different proportions, categorize it, and contract the relationships identified. The major components that play an important role in data mining are: clustering, classification and link analysis. Data mining tools predict future trends and behaviors that allow businesses to make anxious and knowledge-driven decisions. In the past years, most of the emerged approaches derive only the frequent itemsets in the database, for example Association Rule Mining (ARM) that describes the relationship between the disjoint sets and considerably manipulates in finding co-occurrences, frequent pattern and statistical correlation. According to users preference the ARM does not reflect the itemsets semantic significance. Applications may have different objectives for data models; thus, there is no single measure that is suitable for every application. To make it more efficient, many algorithms were developed [3, 6, 7, 8, 13, 14, 15, 16, 19, 20, 21, 22, 23, 24] based on the value of the product as prescribed by the users, which in turn resulted in utility mining.

Utility mining is followed in order to find the high utility itemsets from the given data sets. To overcome some of the limitations of traditional approaches, a new algorithm termed as Fast Utility Mining [13] was developed to find the high utility item sets. It is simpler and executes faster than UMining [21] algorithm. Existing methods based on utility mining work on centralized systems. The proposed work here is to implement the FUM algorithm in distributed environment to find the high utility itemsets over various slave sites and providing the overall utility value at the master site.

The data being stored at different locations are subjected to a two stage process. In the first stage we determine the utility values for the itemsets at each individual location in the distributed network. The utility values are calculated at

the slave sites in a parallel manner thereby improving the overall performance in the distributed network. In the second stage we calculate the total utility for the itemsets at the master site. This is done by augmenting the values from the individual slave sites using a distributed utility mining algorithm. The result is filtered by subjecting it to a comparison process with the utility threshold to result in high utility itemsets. A distributed utility mining algorithm is used globally, in order to combine, compare and generate the itemsets of high utility value from all slave sites in the distributed network.

The practical usefulness of mining high utility itemsets from the distributed databases is best clarified in the following examples.

Example 1 (Retail Marketing)**.** The most established supermarkets have their branches all over the world. A separate database is maintained in each branch to keep track of the purchasing behavior of consumers. Data analyst needs to integrate the databases spread over the world to mine high utility itemsets which takes more time and requires more memory to compute high utility itemsets. In this situation the algorithm needs to be proposed to perform high utility itemset mining from the distributed databases without database integration so that the execution time and memory consumption is reduced.

Example 2 (Web Mining)**.** High utility itemset mining approach is helpful for finding significant itemsets in many applications. Consider a table viewed as representing a set of web pages for web mining, with each column representing a keyword, each row representing a webpage, and the value in each cell indicating the number of occurrences of that keyword in the webpage. There is another table utilized to represent user's preference among these keywords. By using the utility mining algorithm the web pages matching a user's interest could be discovered. If these databases are distributed over different places, the data integration has to be done since existing utility mining algorithms only accepts centralized databases. In this way, mining of high utility itemsets from distributed databases is the need of hour.

Example 3 (E-Commerce)**.** Utility mining is a powerful tool to realize cross selling. Cross selling is a marketing strategy to sell a new product or service to the customer who already used the products of the same enterprise. An old customer is more likely to accept if the enterprise is introducing a new product so that we can increase the sales of the new product. Consider XYZ Company is well established in selling refrigerators. When a client buys a XYZ refrigerator online, he/she would definitely be in need of a stabilizer. The mostly preferred, best manufacturer of stabilizers is ABC Company. Meanwhile, XYZ Company will also be manufacturing stabilizers but it might not be in the frequently sold items list or might be newly introduced in the market. In order to promote the non-frequent item "stabilizer" of XYZ company, utility mining may be used to find the high utility itemsets in a database that is located anywhere in the location. Consider one of the High Utility Itemsets (HUI) to be a XYZ company refrigerator and ABC company stabilizer. For promoting the stabilizer manufactured by XYZ, it can be sold at discounted price.

The remaining parts of the paper are organized as follows. In Section 2, basic concepts of utility mining and algorithms are presented. Section 3 discusses distributed data mining. In Section 4, the proposed approach to mine high utility itemsets from distributed databases is explained. In Section 5 all experimental results are discussed. Section 6 concludes the paper.

2 RELATED WORK

In today's world the most provoking task is mining of high utility itemsets precisely. High utility itemsets being identified is manifested as utility mining. Utility mining is a vast area which wraps all aspects of mercantile utility in data mining. The utility value of an itemset can be computed in terms of cost, profit or other interpretation of user preferences. An itemset x is said to be a high utility itemset if and only if $u(x) \geq$ minUtil, where minUtil is a user defined minimum utility threshold. This subsection starts with the definition of a set of terms that leads to the formal definition of utility mining problem which is given in [22].

2.1 Basic Concepts and Definitions

$I = \{i_1, i_2, \ldots, i_m\}$ is a set of items, D= $\{T_1, T_2, \ldots, T_n\}$ is a transaction database where each transaction $T_i \in D$ is a subset of I. $o(i_p, T_q)$ is local transaction utility value, representing the quantity of item i_p in transaction T_q, for example, $o\,(A, T_8) = 3$, in Table 1. $s(i_p)$, external utility, is the value associated with item i_p in the utility table. This value reflects the importance of an item, which is independent of transactions; for example, in Table 2, the external utility of item A, $s(A)$ is 3.

TID	A	B	C	D	E
T_1	0	0	18	0	1
T_2	0	6	0	1	1
T_3	2	0	1	0	1
T_4	1	0	0	1	1
T_5	0	0	4	0	2
T_6	1	1	0	0	0
T_7	10	0	0	1	1
T_8	3	0	25	3	1
T_9	1	1	0	0	0
T_{10}	0	6	2	0	2

Table 1. Transaction table with 10 transactions and 5 distinct items

$u(i_p, T_q)$, utility, the quantitative measure of utility i_p in transaction T_q, is defined as $o(i_p, T_q) \times s(i_p)$; for example, $u\,(A, T_8) = 3 \times 3$ in Table 1. $u(X, T_q)$, utility of an itemset X in transactionT_q , is defined as $\sum_{i_p \in X} u(i_p, T_q)$, where $X =$

Item	A	B	C	D	E
Profit	3	10	1	6	5

Table 2. External utilities of items from table given in Table 1

$\{i_1, i_2, \ldots, i_k\}$ is a k-itemset, $X \subseteq T_q$ and $1 \leq k \leq m$. $u(X)$, utility of an itemset X, is defined as

$$\sum_{T_q \in D \wedge X \subseteq T_q} u(X, T_q). \tag{1}$$

We find all the high utility itemsets using Utility Mining. An itemset X is a *high utility itemset* if $u(X) \geq$ minUtil, where $X \subseteq I$ and minUtil is the minimum utility threshold.

For example, in Table 1, $u(A, T_8) = 3 \times 3 = 9$, $u(\{A, D, E\}, T_8) = u(A, T_8) + u(D, T_8) + u(E, T_8) = 3 \times 3 + 3 \times 6 + 1 \times 5 = 32$, and $u(\{A, D, E\}) = u(\{A, D, E\}, T_4) + u(\{A, D, E\}, T_8) = 14 + 32 = 46$. If minUtil = 120, then $\{A, D, E\}$ is not a high utility itemset.

2.2 UMining Algorithm

One of the well known algorithms used for mining all high utility itemsets is UMining [21]. Figure 1 briefly describes the UMining algorithm. More details of the UMining algorithm and detailed description of the functions Scan, CalculateAndStore, Discover, Generate, and Prune can be found in [21].

Input: database T
constraints minUtil
Output: all high utility itemsets H

```
[1]  I = Scan(T);
[2]  C_1 = I;
[3]  k = 1;
[4]  C_k = CalculateAndStore(C_k, T, f);
[5]  H = Discover(C_k, minUtil);
[6]  while (|C_k > 0 and k <= K)
[7]  {
[8]      k = k + 1;
[9]      C_k = Generate(C_{k-1}, I);
[10]     C_k = Prune(C_k, C_{k-1}, minUtil);
[11]     C_k = CalculateAndStore(C_k, T, f);
[12]     H = H ⋃ Discover(C_k, minUtil);
[13] }
[14] return H;
```

Fig. 1. Pseudo code of the UMining algorithm

The amount of memory storage required to accommodate all the distinct items at any time when the UMining algorithm accomplishes is vast. UMining algorithm does not provide a prominent foundation for dealing with duplicate itemsets which could occur in any transactional database. UMining algorithm is proved to asset approximately all high utility itemsets from a given database. Due to some instances as specified earlier, the UMining algorithm may decline to find some of the high utility item sets from the accessible transactions in the databases. It becomes clear from the above examination that the existing UMining algorithm may not be a good practice, when dealing with databases having millions of transactions with large number of items. It costs more and does not adapt completely.

2.3 FUM Algorithm

FUM [13] is developed in order to overcome and avoid the errors and pitfalls that occurred during mining. FUM is an efficient and accurate algorithm in finding every possible high utility itemset from the transactions in the database. It is simple and executes faster than UMining. FUM provides perfect reliability in finding high utility itemsets. The amount of storage and hardware required are reduced drastically. It removes the duplicate itemsets that occur during the transaction. The FUM algorithm is presented in Figure 2.

```
Task:    Discovery of High Utility Itemsets
Input:   Database DB {Set of Transactions}
         Transaction T ∈ DB
         Minimum Utility value threshold minUtil
Output:  High Utility Itemsets H
[1]  Compute the utility value ∀ single itemset
[2]  for each T ∈ DB
[3]  begin
[4]      if T ∉ S{where S ⊆ DB‖S = [0..T − 1]}
[5]      begin
[6]          Candidateset = CombinationGenerator(T)
[7]          for each C ∈ CandidateSet
[8]          begin
[9]              if (C ∉ H) ∧ (U(C, T) ≥ minUtil)
[10]             H.add(C);
[11]         end
[12]     end
[13] end
[14] return (H);
     CombinationGenerator(T) – Generate all possible combinations of
     itemset ∈ T
```

Fig. 2. Pseudo code of the FUM algorithm

The CombinationGenerator(T) is a method which is used to generate all the combinations of the items. It takes Itemid and the level as input which is generally denoted by the variable loop. The factorial computation method is defined in this, to generate the factorial of a given number. The combination generation is based on the concept proposed by Kenneth H. Rosen (1999). First the items for which the combination is to be generated is put in the form of an array. Then the getNext() method is called until there are no more combinations left. The getNext() method returns an array of integers, which tells the order in which to arrange the original array of letters.

Let us consider Table 1 and Table 2 as input to the proposed FUM algorithm. We compute the utility values of all single itemsets, say A, B, C, D and E in step 1 (as explained in Section 2.1). In step 2, we begin a loop for processing each and every transaction present in the DB one by one. In step 4, the algorithm generates the itemsets in the current transaction. For example, in Table 1, the first transaction is represented as CE according to FUM algorithm, since only those two items were purchased in that transaction. FUM algorithm omits the remaining items A, B and D. In a similar way, the remaining transactions are processed. The algorithm also checks (step 4), whether a transaction defined by an itemset purchased in it repeats its occurrence in a later transaction. If a later transaction also contains the same itemset purchased in any of the previous transactions, then that transaction is ignored from processing. In step 6, the candidate itemsets are generated using the CombinationGenerator(T) function, which takes the itemset purchased in a particular transaction as input and generates the various possible combinations of the itemset. In the consecutive steps, the algorithm analyzes each candidate belonging to the candidate itemsets generated. In step 9, the algorithm computes the utility value of each and every candidate, U(C,T) as described in Section 2. If the utility value of a candidate is found to be more than the minimum utility threshold, which is given as input by the user (say a sales manager), then that particular candidate is added to the set of High Utility Itemsets H in step 10 of the algorithm. The condition $C \notin H$ in step 9 simply ensures no duplicate high utility itemsets are generated.

3 DISTRIBUTED DATA MINING

There are various approaches for distributed data mining such as association rules [1, 4, 12, 17, 18, 25, 26], and classification [9, 11]. Vo et. al [2] proposed an approach to mine the HUIs from vertical databases using TWU (Transaction Weighted Utilization) mining algorithm [5]. The need arises to propose an algorithm which works well in vertical as well as horizontal databases.

3.1 Drawbacks in Mining HUI from Centralized Databases

Data integration when mining multiple data sources. In business enterprise applications, data is distributed over many heterogeneous sources. Distributed

data sources related to business enterprises are often complex such as data are of high frequency or density and of multiple structures. If the data analyst wanted to mine High Utility Itemsets in a centralized environment, he/she needs to integrate databases in each site together and perform data mining in centralized environment which is not feasible, since centralized system consumes enormous amount of time and requires more memory.

Communication Cost. Let us suppose a data analyst wants to perform High Utility Itemset Mining over data that is distributed through networks, centralized server needs to give request to send the data and each site sends the database accordingly using the communication line; this incurs high communication cost.

Load of the server is high if we use centralized approach. The core step of the utility mining algorithm is candidate generation which takes significant time to compute. If the utility mining process is performed in centralized environment, the system needs to have a large amount of memory and the server load increases significantly.

3.2 Benefits of HUI Mining from Distributed Databases

Data integration is not needed. The system proposed in this paper does not require data integration since the core process of utility mining called candidate generation' is performed and utility values for all itemsets are calculated in all sites and returned to the main site. Finally, the main site collects all local utility itemsets and finds the global High Utility Itemsets (HUI) effectively.

Centralized algorithm is decomposed into parts and workloads are distributed. The centralized version of algorithm performs candidate generation and utility value computation for all candidates. This utility value is checked against minUtil threshold value. Since data size is very large, the system takes enormous amount of time to mine High Utility Itemsets (HUI); but in the proposed FUM-D approach, each slave site computes utility itemsets locally and utility itemsets are sent to the master site. The master site in turn collects all local utility itemsets and mines global high utility itemsets, hence workloads are distributed effectively.

Subtasks are executed in parallel at various computing nodes; hence execution time is considerably reduced. Computation of utility value for all the candidates is done at all sites in parallel manner. Finally the utility itemsets are sent to the main site through distributed communication network. This will greatly reduce the execution time.

Lower communication cost. In order to find HUI, the proposed algorithm requires $O(nm)$ messages for utility itemset exchange, where n is the number of sites in the distributed network and m is the number of utility itemsets returned by the slave sites.

4 AN APPROACH FOR MINING HUI FROM DISTRIBUTED DATABASES

4.1 Methodology

Consider the database D is divided into m sites $D = \{D_1, D_2, \ldots D_m\}$. The transactions in D_j $(1 \leq j \leq n)$ retail dataset have k distinct items $I = \{i_1, i_2, \ldots i_k\}$. The transaction identifier, products purchased and quantity of the product bought is updated in the corresponding site wherever the customer is doing a transaction. Hence the system is not being centralized which makes the retail data store easy to manage and does not overload the system.

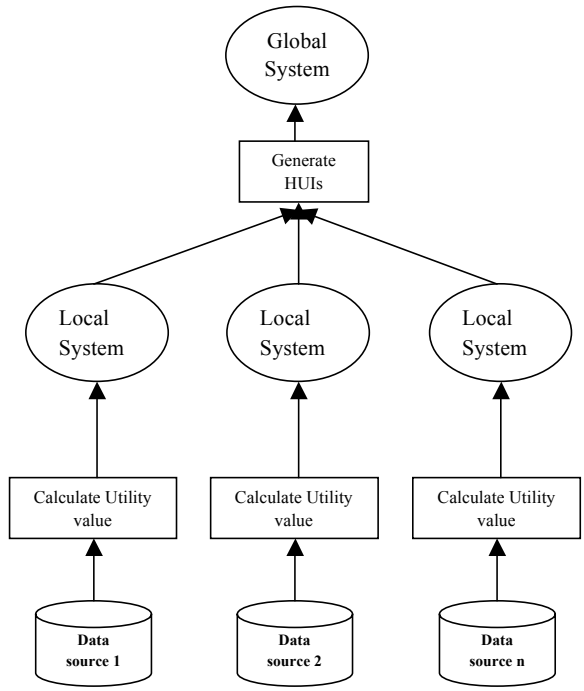

Fig. 3. General framework for mining HUIs from distributed databases

However, the issue is how to mine HUIs from the databases which are all located at different sites without integration. This issue is addressed in Figure 3. Consider the databases are distributed over many sites. Initially the master site broadcasts request for calculating the utility value of all candidates to all slave sites. When the master site receives all utility itemsets, it will mine HUIs by calling the algorithm which is given in Figure 4. The master site only triggers to compute utility itemsets

and the slave sites respond accordingly. Hence the communication cost is greatly reduced.

4.2 FUM-D Algorithm

The proposed algorithm is given in Figures 4 and 5. At the slave site, the process starts to scan each and every transaction present in the DB one by one by using loop. In step 4.4, the algorithm generates the itemsets in the current transaction. The algorithm checks (step 4.4) whether a transaction defined by an itemset purchased in it repeats its occurrence in a later transaction. If a later transaction also contains same itemset purchased in any of the previous transactions, then that transaction is ignored.

```
Input:    Database DB {Set of Transactions}
Output:   Local Utility Itemsets LUI
[4.1]     index = 0;
[4.2]     for each T ∈ DB
[4.3]     begin
[4.4]         if (T ∉ S) ∧ (T ⊄ S){where S ⊆ DB‖S = [0..T − 1]}
[4.5]         begin
[4.6]             CS = generateCombination(T);
[4.7]             for each C ∈ CS
[4.8]             begin
[4.9]                 if C ∉ LUI
[4.10]                begin
[4.11]                    LUI.add(index, C);
[4.12]                    LUI.get(index).utility = U(C, T);
[4.13]                    index++;
[4.14]                end
[4.15]            end
[4.16]        end
[4.17]    end
[4.18]    return (LUI);
          CS-Candidate Set
          LUI- Local Utility Itemset
          generateCombination(T) – Generate all possible combinations of
          itemset ∈ T
```

Fig. 4. Pseudo code for calculating utility value at slave site

The condition also checks if the utility value for any of the subsets has been computed already; then the subsets are not generated again. The candidate itemsets are generated using the generateCombination(T) function in step 4.6, which takes the itemset purchased in a particular transaction as input and generates the various

possible combinations of the itemset. Each candidate is checked for the existence of the Local Utility Itemset (LUI). If the candidate is not available, utility value is calculated for the corresponding candidate. This process is done for all the candidates and LUI is returned to the master site.

Input: Utility Itemsets of all Sites
Minimum Utility threshold α (%)
Output: High Utility Itemsets HUI
[5.1] for each site $k = 1$ to $n\{n =$ Number of sites$\}$
[5.2] begin
[5.3] for each $I \in \text{LUI}_k$
[5.4] begin
[5.5] if $I \in$ GUI
[5.6] sum the utility value of I;
[5.7] else
[5.8] GUI.add(I);
[5.9] end
[5.10] end
[5.11] minUtil = calculateMinUtil(GUI, α);
[5.12] for $i = 1$ to GUI.size()
[5.13] begin
[5.14] if GUI(i).utility $\geq$ minUtil
[5.15] HUI.add(I);
[5.16] end
[5.17] return (HUI);
LUI-Local Utility Itemset
GUI-Global Utility Itemset
HUI-High Utility Itemsets
calculateMinUtil(GUI, α) – It retrieves all single itemset from GUI and calculates minUtil value using the formula given in [21]

Fig. 5. Pseudo code for generating High Utility Itemset (HUI) of n-sites at master site

The master site receives all utility itemsets along with utility values whenever the slave site returns and checks the existence of the itemsets with Global Utility Itemset (GUI) (step 5.5). If it is found, the utility value of the corresponding itemset is summed with the utility value of the GUI; otherwise the utility itemset is added with GUI. In step 5.11, minUtil threshold value is calculated by using the calculateMinUtil(GUI, α) method. From steps 5.12–5.16, the utility value of all GUIs is compared against minUtil threshold and HUIs are returned to the data analyst.

5 EXPERIMENTAL RESULTS

We evaluated the performance of FUM-D algorithm for mining high utility itemsets from distributed databases and compared it with centralized FUM algorithm. Nodes in the network have specification 2.00 GHz Intel (R) Core (TM) 2 CPU with 1 GB RAM, and run on Windows XP. The algorithms were implemented in Java language.

5.1 IBM Synthetic Data

The data utilized in the experimental results is widely-accepted IBM synthetic data called T10I4D100K which is obtained from IBM dataset [10]. This dataset contains 1 000 000 transactions and 1 000 distinct items. T10I4D100K denotes the average size of the transactions (T), average size of the maximal potentially large itemsets (I) and the number of transactions (D). One more column called "Quantity" (value in range of 1 to 10) is added with synthetic data. The utility values for the items were assigned randomly in the profit table. The experiments were conducted in distributed environment which includes 10 slave sites and one master site. Each slave site has 10 K transaction records.

Minimum Utility Threshold (%)	**FUM**		**FUM-D**	
	#HUIs	**Execution time in minutes**	**#HUIs**	**Execution time in minutes**
0.25	3 310	1 072.12	3 310	134.18
0.5	1 298	1 069.15	1 298	136.95
0.75	748	1 075.43	748	133.66
1	306	1 074.02	306	134.83
5	2	1 071.67	2	135.07

Table 3. Comparison of execution time, High Utility Itemsets (HUI) and minimum utility threshold in FUM and FUM-D algorithms

The experiments were conducted by varying the minimum utility threshold (refer to Table 3) by keeping 1 000 distinct items as fixed. The test results are illustrated in Figure 6. It can be observed that the execution time of FUM algorithm for mining high utility itemsets from distributed databases proved to be significantly less than existing FUM algorithm on centralized environment as observed in Figure 6.

The experiments were also conducted by varying the number of distinct items from 50 to 1 000 that are totally available by keeping the Minimum Utility Threshold at 0.25 % throughout the experiment and execution time was recorded for FUM and FUM-D algorithms (cf. Table 4 and Figure 7).

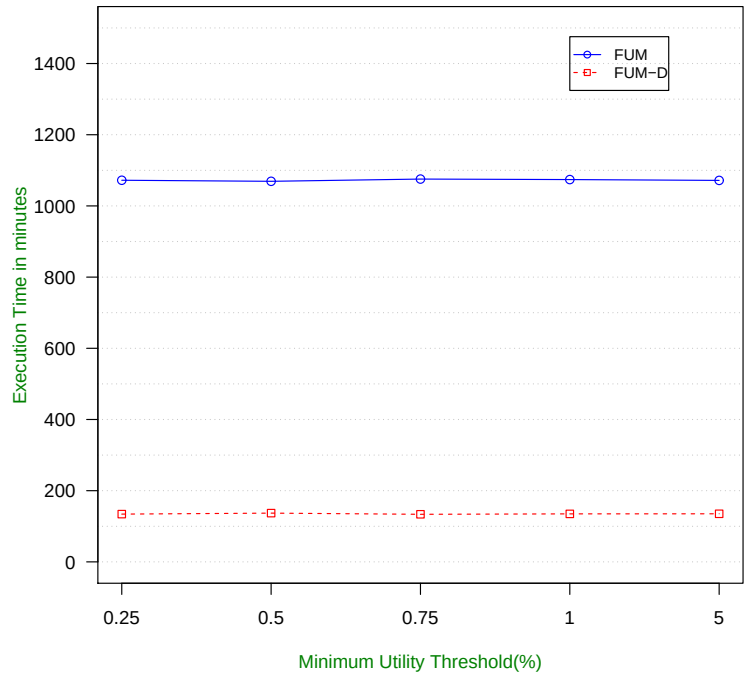

Fig. 6. Comparison of execution time and minimum utility threshold in FUM and FUM-D algorithms

Number of Items	**Execution time in minutes**	
	FUM	**FUM-D**
50	483.12	67.20
100	738.31	93.54
200	894.76	113.10
500	992.45	124.73
1 000	1 072.12	134.18

Table 4. Comparison of FUM and FUM-D algorithms based on the number of items

5.2 Real-World Supermarket Data

The proposed FUM-D algorithm is also evaluated using a real world data from a well known supermarket store in India and compared with centralized version of FUM algorithm. It contains products from various categories, such as food, health care, gifts, and others. There are 20 000 transactions and 7 007 distinct items in the database. Each transaction consists of the products and the quantity of each product purchased by a customer at a particular point of time. The average number of

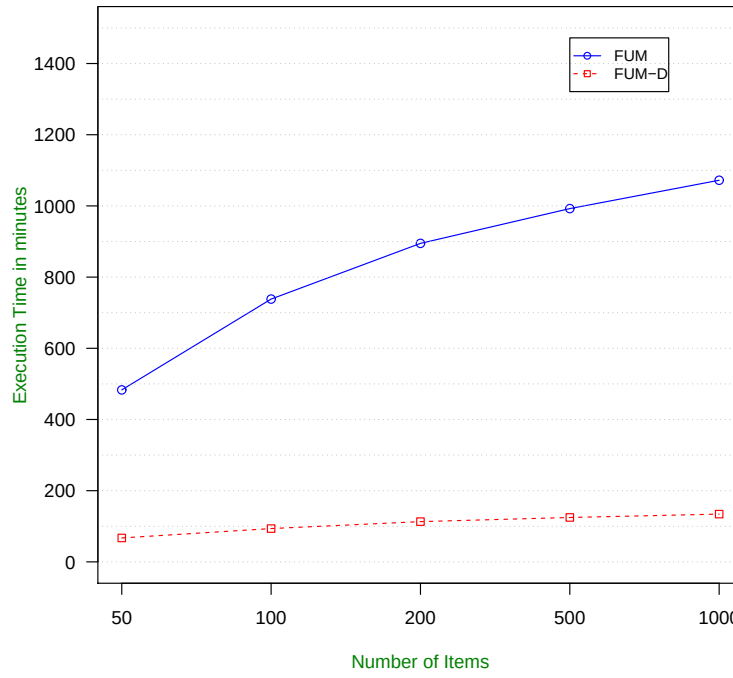

Fig. 7. Comparison of FUM and FUM-D algorithms based on the number of items

items in a transaction is 6.3. The utility table describes the profit of each product. 20 000 transactions are equally distributed over 10 slave sites and execution times are recorded using FUM-D algorithm. FUM-D algorithm is compared against centralized version of FUM algorithm by varying minUtil threshold value and keeping 7 007 distinct items fixed (Table 5). The results are illustrated in Figure 8.

Minimum Utility Threshold(%)	**FUM**		**FUM-D**	
	#HUIs	**Execution time in minutes**	**#HUIs**	**Execution time in minutes**
0.25	10 769	440.51	10 769	50.30
0.5	7 232	424.76	7 232	52.08
0.75	3 510	418.29	3 510	51.48
1	1 243	399.23	1 243	46.77
5	743	394.50	743	49.91

Table 5. Comparison of execution time, High Utility Itemsets (HUI) and minimum utility threshold in FUM and FUM-D algorithms on real-world supermarket data

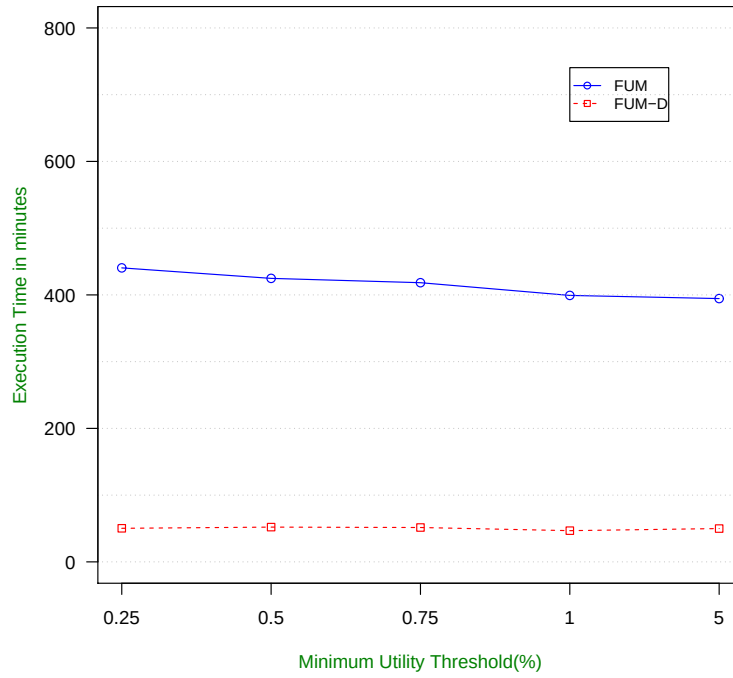

Fig. 8. Comparison of execution time and minimum utility threshold in FUM and FUM-D algorithms

The experiments were also conducted in real-world supermarket data by varying the number of distinct items from 50 to 1 000 that are totally available by keeping the minimum utility threshold at 1 % throughout the experiment and execution time was recorded for FUM and FUM-D algorithms (cf. Table 6 and Figure 9).

Number of Items	**Execution time in minutes**	
	FUM	**FUM-D**
50	316.63	21.47
100	328.46	29.27
200	341.12	37.41
500	375.49	44.23
1 000	394.50	49.91

Table 6. Comparison of FUM and FUM-D algorithms based on the number of distinct items in real-world supermarket data

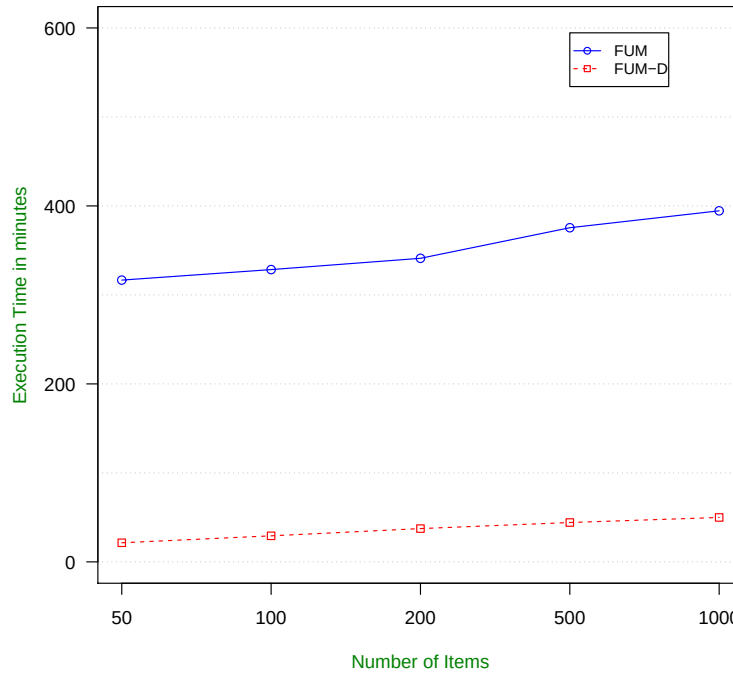

Fig. 9. Comparison of FUM and FUM-D algorithms based on the number of items

6 CONCLUSION

In this paper, a novel algorithm for mining high utility itemsets from distributed databases is proposed. The crucial step in the utility mining algorithms for centralized environment is the candidate generation process which consumes significant computation time and hardware resources. The novel algorithm proposed in this paper utilizes the concept of parallel mining for distributed environment as it generates the candidate itemsets and computes utility values at different slave sites simultaneously. Therefore the execution time required for the computation of utility values is significantly reduced compared to the centralized version of utility mining algorithms. This can be clearly observed through the experimental results from both IBM synthetic data and real world supermarket data as discussed in Section 5 of the paper.

REFERENCES

[1] Agrawal, R.—Shafer, J.: Parallel Mining of Association Rules. IEEE Trans. Knowledge and Data Eng., Vol. 8, 1966, No. 6, pp. 962–969.

[2] Vo, B.—Nguyen, H.—Le, B.: Mining High Utility Itemsets from Vertical Distributed Databases. Proc. of International Conf. on Computing and Communication Technologies 2009, pp. 1–4.

[3] Chan, R.—Yang, Q.—Shen, Y. D.: Mining High Utility Itemsets. Proceedings of 3rd IEEE Intl. Conf. on Data Mining, Melbourne, FL 2003, pp. 19–26.

[4] Cheung, D. et al.: A Fast Distributed Algorithm for Mining Association Rules. Proc. 4th Intl. Conf. Parallel and Distributed Information Systems 1996, IEEE Computer Soc. Press, Los Alamitos, California, pp. 31–42, 1996.

[5] Erwin, A.—Gopalan, R. P.—Achuthan, N. R.: CTU-Mine: An Efficient High Utility Itemset Mining Algorithm Using the Pattern Growth Approach. In: IEEE 7th International Conference on Computer and Information Technology, Aizu Wakamatsu, Japan, pp. 71–76, 2007.

[6] Erwin, A.—Gopalan, R. P.—Achuthan, N. R.: A Bottom-Up Projection Based Algorithm for Mining High Utility Itemsets. Proceedings of the 2nd International Workshop on Integrating Artificial Intelligence and Data Mining, Volume 84, Gold Coast, Australia, pp. 3–11, 2007.

[7] Le, B.—Nguyen, H.—Cao, T. A.—Vo, B.: A Novel Algorithm for Mining High Utility Itemsets. In: Proceedings of 1st Asian Conference on Intelligent Information and Database Systems, Quang Binh, Vietnam 2009, IEEE Pess, pp. 13–17.

[8] Liu, Y.—Liao, W.—Choudhary, A.: A Fast High Utility Itemsets Mining Algorithm. UBDM '05 , August 21, 2005, Chicago, Illinois, USA, pp. 90–99.

[9] Luo, P.—Xuong, H.—Lu, K.—Shi, Z.: Distributed Classification in Peer-to-Peer Networks. KDD07, San Jose, California, USA, August 2007.

[10] IBM synthetic data generation code. `http://www.almaden.ibm.com/software/quest/Resources/index.shtml`.

[11] Miller, D. J.—Zhang, Y.—Kesidis, G.: Decision Aggregation in Distributed Classification by a Transductive Extension of Maximum Entropy/Improved Iterative Scaling. EURASIP Journal on Advances in Signal Processing, Volume 2008 (doi:10.1155/2008/674974), 2008.

[12] Schuster, A.—Wolff, R.: Communication-Efficient Distributed Mining of Association Rules. In Proc. of the 2001 ACM SIGMOD Int'l. Conference on Management of Data, Santa Barbara, California, pp. 473–484, 2001.

[13] Shankar, S.—Purusothaman, T.—Jayanthi, S.: A Fast Algorithm for Mining High Utility Itemsets. IEEE International Advance Computing Conference (IACC 2009), Patiala, India.

[14] Shankar, S.—Purusothaman, T.: PA Novel Utility Sentient Approach for Mining Interesting Association Rules. IACSIT International Journal of Engineering and Technology, Vol. 1, 2009, No. 5, ISSN: 1793-8236.

[15] Shankar, S.—Purusothaman, T.: Discovering imperceptible associations based on Interestingness: A Utility-Oriented Data Mining Approach. Data Science Journal, Vol. 9, February 2010.

[16] Shankar, S.—Purusothaman, T.: Utility Sentient Frequent Itemset Mining and Association Rule Mining: A Literature Survey and Comparative Study. International Journal of Soft Computing Applications, ISSN: 1453-2277, Issue 4 (2009), pp. 81–95.

[17] Tang, P.—Turkia, M.: Parallelizing Frequent Itemset Mining with FP-trees. Technical Report, Department of Computer Science, University of Arkansas at Little Rock 2005.

[18] Sujni, P.: An Optimized Distributed Association Rule Mining Algorithm in Parallel and Distributed Data Mining with XML Data for Improved Response Time. International Journal of Computer Science and Information Technology, Vol. 2, 2010, No. 2.

[19] Liu, Y.—Liao, W. K.—Choudhary, A.: A Two-Phase Algorithm for Fast Discovery of High Utility Itemsets. In: T. B. Ho, D. Cheung, H. Liu (Eds.), 9th Pacific-Asia Conf. on Advances in Knowledge Discovery and Data Mining (PAKDD 2005), Lecture Notes in Computer Science, Vol. 3518, pp. 689–695, Springer-Verlag, Berlin 2005.

[20] Liu, Y.—Liao, W. K,.—Choudhary, A.: A Fast High Utility Itemsets Mining Algorithm. Proceedings 1st Intl. Conf. on Utility-Based Data Mining, Chicago, IL, August 2005, pp. 90–99.

[21] Yao, H.—Hamilton, H. J.: Mining Itemset Utilities from Transaction Databases. Data and Knowledge Engineering 59 (2006), pp. 603–626.

[22] Yao, H.—Hamilton, H. J.—Butz, C. J.: A Foundational Approach to Mining Itemset Utilities from Databases. Proceedings 4th SIAM Intl. Conf. on Data Mining, Lake Buena Vista, FL, April 2004, pp. 482–486.

[23] Yao, H.—Hamilton, H. J.—Geng, L.: A Unified Framework for Utility-Based Measures for Mining Itemsets. Proceedings ACM SIGKDD – 2nd Workshop on Utility-Based Data Mining, Philadelphia, Pennsylvania, August 2006, pp. 28–37.

[24] Li, Y.-C.—Yeh, J.-S.—Chang, C.-C.: Isolated Items Discarding Strategy for Discovering High Utility Itemsets. Elsevier Journal, Data and Knowledge Engineering 64, 2008, pp. 98–217.

[25] Wolff, R.—Schuster, A.: Association Rule Mining in Peer-to-Peer Systems. IEEE Trans. Systems, Man and Cybernetics, Part B, Vol. 34, 2004, No. 6, pp. 2426–2438.

[26] Zaki, M. J.: Parallel and Distributed Association Rule Mining: A Survey. IEEE Concurrency, Special Issue on Parallel Mechanism for Data Mining, 1999, pp. 14–25.

Kannimuthu Subramanian is currently working as an Assistant Professor in the Department of Information Technology at Sri Krishna College of Engineering and Technology, Coimbatore, India. He is now doing his Ph. D. in computer science and engineering at Anna University, Chennai, India. He did his Master of Engineering in CSE and Bachelor of Technology in IT at Anna University, Chennai, India. He has more than 6 years of teaching experience. He has published many papers in various international journals. He has presented a number of papers in various national and international conferences. He has delivered more than 25 guest lectures in reputed engineering colleges on various topics. He has guided a number of research-oriented as well as application oriented projects organized by well-known companies like IBM. His research interests include data mining, component based enterprise software development, web services and open source technologies.

Premalatha Kandhasamy is currently working as a Professor in the Department of Computer Science and Engineering at Bannari Amman Institute of Technology, Erode, India. She completed her Ph. D. in computer science and engineering at Anna University, Chennai, India. She did her Master of Engineering in CSE and Bachelor of Engineering in CSE at Bharathiar University, Coimbatore, India. She has 17 years of teaching experience in academic field. She published 25 papers in national and international journals and presented more than 20 papers in international and national conferences. Her research interests include data mining, information retrieval and soft computing.

Shankar Subramanian is currently working as an Associate Professor in the Department of Information Technology at Sri Krishna College of Engineering and Technology, Coimbatore, India. He did his Ph. D. in computer science and engineering at Anna University of Technology, Coimbatore, India. He did his Master of Engineering in CSE and Bachelor of Engineering in CSE at Anna University, Chennai and Bharathiar University, respectively. He has more than 11 years of teaching experience. He is an IBM certified DB2 professional and has obtained Brain Bench certification in various disciplines. He has presented a number of papers in various national and international conferences and journals. He has guided a number of research-oriented as well as application oriented projects organized by well-known companies like IBM. His research interests include data mining, soft computing and database management systems.