International Journal of Advanced Information Science and Technology (IJAIST) ISSN: 2319:268
Vol.3, No.2, February 2014 DOI:10.15693/ijaist/2014.v3i2.87-92

DGU and DGN Strategies for Mining High Utility Itemsets

Rajmohan.C
Assistant Professor/Department of IT
SREC,coimbatore, India

Niveditha.C[1],Pragathi.R[2],Priya.G[3]
Student/Department of IT
SREC, Coimbatore

***Abstract*— Finding frequent patterns from the transaction tables are still an important issue in the field of data mining. For producing Itemsets which are frequently purchasing from huge amount of data sources improving the efficiency of business applications. But, Frequent Pattern data structure tree produces only frequent patterns and there is a need to generate high utility itemsets to improve the performance of market basket analysis. Producing the Itemsets with high utility means, searching the Itemset with huge Interestedness from the data source. Even though the number of frequent pattern algorithms have been used, but some problems still exist such as searching the frequent itemsets when the database consists of large amount of transactions in database. The major problem depends on large number of candidates. utility pattern tree(UP-Growth) algorithm is used to reduce the number of candidate itemsets and also it performs effectively in long transactions, thereby it reduces database scans.**

Index terms –Candidate Elimination, Frequent pattern, High Utility, Itemset Mining , Utility.

I. INTRODUCTION

Data mining, also popularly known as Knowledge Discovery in Databases (KDD), refers to the nontrivial extraction of implicit, previously unknown and potentially useful information from data in databases. Mining of Frequent Itemsets plays vital role in mining applications such as website click stream analysis, cross-marketing in retail stores, e-commerce management. The Apriori algorithm[1], which are used for association rule mining[14] in market basket analysis, all the frequent itemsets are retrieved from transactional database. In past, Some previous studies are utilized and produced with candidate and without candidate generation. But it is not producing the customer requirement like profit, sales in particular item. Itemset mining with high interestedness or utility plays vital role in mining area. The unit profits and purchased quantities of items are not considered in the framework of mining frequent itemset. Hence, it cannot satisfy the requirement of the user who is interested in discovering the itemsets with high sales profits. In view of this, utility mining emerges as an important topic in data mining for discovering the itemsets with high utility like profits. The basic meaning of utility is interestedness / importance/ profitability of items to the users. The utility of items in a transaction database consists of two aspects namely, internal utility and external utility. To improve the mining performance and avoid scanning original database repeatedly, use a compact tree structure, named UP-Tree[16], to maintain the information of transactions and high utility itemsets.

II. MATHEMATICAL BACKGROUND

The definitions for UP-Tree construction are utility of an item, utility of an item in database, transaction utility, transaction weighted utility, High utility and their related works are as follows.

A. Preliminary

Given a set of items *I={i1,i2,...im}* , each item $ip(1 \le p \le m)$ has a unit profit *pr(ip).* A transaction database *D={T1,T2,....,Tn}* contains a set of transactions, and each transaction $Td(1 \le d \le n)$ has a unique identifier d, called TID. Each item ip in transaction Td is associated with a quantity *q(ip,Td)*, (i.e) the purchased quantity of *ip* in *Td.*

Definition 1. Utility of an item ip in a transaction Td is denoted as *u(ip,Td)* and defined *as pr(ip)*$\times q(ip,Td)$.

Definition 2. Utility of an itemset X in Td is denoted as u(X, Td) and defined as $\sum \; ip \in X \wedge X \subseteq Td \; u(ip,Td)$.

Definition 3. Utility of an itemset X in D is denoted as u(X) defined as $\sum \; X \subseteq Td \wedge Td \in D \; u(X,Td)$.

Definition 4. An itemset is called a high utility itemset if its utility is no less than user-specified minimum utility threshold which is denoted as min_util. Otherwise, it is called a low-utility itemset.

Definition 5. Transaction utility of a transaction Td is denoted as *TU(Td)* and defined as *u(Td,Td)*.

Definition 6. Transaction-weighted utility of an itemset X is the sum of the transaction utilities of all the transactions containing X, which is denoted as TWU(X) and defined as $\sum \; X \subseteq Td \wedge Td \in D \; TU(Td)$.

Definition 7. An itemset X is called a high-transaction weighted utility itemset (HTWUI) if TWU(X) is no less than min_util.

B. Related Work

In frequent pattern mining, the most famous are association rule mining [1, 13, 14] and sequential pattern mining [2].

Association rule mining extracts interesting correlations, frequent patterns, associations or casual structures among sets of items in the transaction databases or other data repositories. Association rules are used in various areas such as telecommunication networks, market and risk management, inventory control etc. Association rule mining is to find out association rules that satisfy the predefined minimum support and confidence from a given database. Association rules are sometimes very large. It is nearly impossible for the end users to comprehend or validate such large number of complex association rules, thereby limiting the usefulness of the data mining results. Several strategies have been proposed to reduce the number of association rules, such as generating only interesting rules, generating only non redundant rules, or generating only those rules satisfying certain other criteria such as coverage, leverage, lift or strength. There are two important basic measures for association rules, support(s) and confidence(c). Since the database is large and users concern about only those frequently purchased items, usually thresholds of support and confidence are predefined by users to drop those rules that are not so interesting or useful. The two thresholds are called minimal support and minimal confidence respectively. Support(s) of an association rule is defined as the percentage/fraction of records that contain $X \cup Y$ to the total number of records in the database. Suppose the support of an item is 0.1%, it means only 0.1 percent of the transaction contain purchasing of this item. Confidence of an association rule is defined as the percentage/fraction of the number of transactions that contain $X \cup Y$ to the total number of records that contain X. Confidence is a measure of strength of the association rules, suppose the confidence of the association rule $X \Rightarrow Y$ is 80%, it means that 80% of the transactions that contain X also contain Y together.

One of the well-known algorithms for mining association rule is Apriori [1]. Pattern growth based association rule mining algorithms [14] such as FP-Growth [14] were afterward proposed. FP-Growth achieves a better performance than Apriori-based algorithms since it finds frequent itemsets without generating any candidate itemset and scans database just twice. In the frequent itemset mining, the importance of items to users is not considered. Thus, weighted association rule mining was brought to attention [4]. Cai et al. first proposed the concept of weighted items and weighted association rules [4]. However, since the weighted association rules does not have downward closure property,mining performance can not be improved. To address this problem, Tao et al. proposed the concept of weighted downward closure property [16]. By using transaction weight, weighted support can not only reflect the importance of an itemset but also maintain the downward closure property during the mining process. Thus, the issue of high utility itemset mining is raised and many studies [3, 5, 10, 16] have addressed this problem. Liu et al. proposed an algorithm named Two-Phase [15] which is mainly composed of two mining phases. In phase I, it employs an Apriori-based level-wise method to enumerate high utility items. Their TWUs are computed by scanning the database once in each pass. After the above steps, the complete set of high utility items is collected in phase I. In phase II, high transaction weighted utility itemsets(HTWUIs) that are high utility itemsets are identified with an additional database scan. Although Two-Phase algorithm reduces search space, it still generates too many candidates to obtain HTWUIs and requires multiple database scans. To overcome this problem, Li et al. [17] proposed an isolated items discarding strategy (abbreviated as IIDS) to reduce the number of candidates. By pruning isolated items during level-wise search, the number of candidate itemsets for HTWUIs in phase I can be reduced. However, this algorithm still scans database for several times and uses a candidate generation.

Apriori is more efficient during the candidate generation process. Apriori uses pruning techniques to avoid measuring certain itemsets, while completeness. These are the itemsets that the algorithm can prove will not turn out to be large. However there are two bottlenecks of the Apriori algorithm. One is the complex candidate generation process that uses most of the time, space and memory. Another bottleneck is the multiple scan of the database. Based on Apriori algorithm, many new algorithms were designed with some modifications or improvements.

FP-Tree, frequent pattern mining, is another milestone in the development of association rule mining, which breaks the main bottlenecks of the Apriori. The frequent itemsets are generated with only two passes over the database and without any candidate generation process. FP-tree is an extended prefix-tree structure storing crucial, quantitative information about frequent patterns. Only frequent length-1 items will have nodes in the tree, and the tree nodes are arranged in such a way that more frequently occurring nodes will have better chances of sharing nodes than less frequently occurring ones. FP-Tree scales much better than Apriori because as the support threshold goes down, the number as well as the length of frequent itemsets increase dramatically. The candidate sets that Apriori must handle become extremely large, and the pattern matching with a lot of candidates by searching through the transactions becomes very expensive. The frequent patterns generation process includes two sub processes: constructing the FT-Tree and generating frequent patterns from the FP-Tree. The mining result is the same with Apriori series algorithms.

III. PROPOSED WORK

Utility mining in knowledge discovery has recently become a prominent re-search issue due to its many practical applications. A high utility itemset in utility mining considers not only quantities but also profits of items in transactions. utility mining is much harder than traditional association-rule mining, as the former lacks the downward closure property. Thus, proposed a two-phase utility mining algorithm to find high utility itemsets in a database by adopting a new downward-closure property. The above is called the transaction-weighted utilization ,which mainly uses the summation of the utility values of all the items in a transaction as the upper bound of any itemset in that transaction to keep

the downward-closure property. With the aid of the TWU model, the whole process of the mining algorithm can be divided into two phases. In the first phase, the possible candidate itemsets are found in a transaction database by the TWU model. Then, in the second phase, the database is rescanned to find the actual utility value of each candidate itemset and identify the high utility itemsets with actual utility values larger than or equal to a pre-defined threshold.

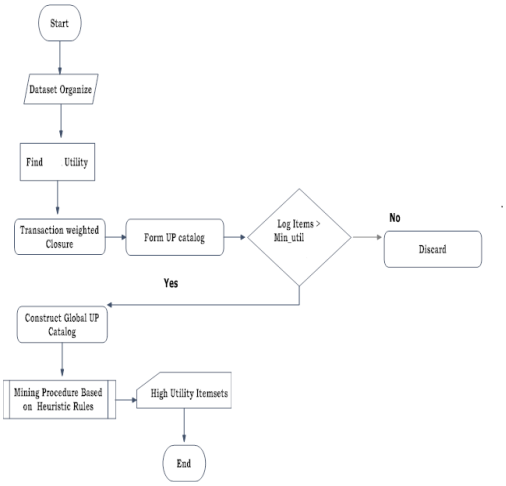


Figure 1- UP-Tree Flow

Subroutine: UP-Tree
Input: Transaction database D, user specified threshold.
Output: high utility itemsets.
1. Scan database of transactions Td ϵ D
2. Determine transaction utility of Td in D and TWU of itemset (X)
3. Get user specified threshold
4. If (TWU(X) ≤ min_sup) then Remove Items from transaction database
5. Else insert into header table H and to keep the items in the descending order.
6. Repeat step 4 & 5 until end of the D.
7. Insert Td into global UP-Tree
8. Apply DGU and DGN strategies on global UP- tree
9. Re-construct the UP-Tree

Figure 2. Subroutine of UP-Tree

To maintain the information from transactional database , UP-Tree is used. In UP-Tree, each node consists of item name, count, utility. A header table consists of information about the traversal of UP-Tree.

Transaction table
Table1

TID	TRANSACTION	TU
T1	(1,1)(3,1)(4,1)	8
T2	(1,2)(3,6)(5,2)(7,5)	27
T3	(1,1)(2,2)(3,1)(4,6)(5,1)(6,3)	28
T4	(2,4)(3,3)(4,3)(5,1)	20
T5	(2,2)(3,2)(5,1)(7,2)	11

Profit Table
Table 2

Item	1	2	3	4	5	6	7
Profit	5	2	1	2	3	5	1

A.UP-Tree Elements:

In a UP-Tree, a node N constitutes of N.name, N.count, N.nu, N.parent, N.hlink and a set of child nodes. N.name is the node's item name. N.count is the node's support count. N.nu is the node's node utility, i.e., overestimated utility of the node. N.parent records the parent node of N. N.hlink is a node link which points to a node whose item name is the same as N.name.

B. Strategy DGU: Discarding Global Unpromising Items

The construction of a global UP-Tree can be performed with two scans of the original database. In the first scan, *Transaction Utility* (also abbreviated as TU) of each transaction is computed. At the same time, *Transaction-Weighted Utility* (also abbreviated as TWU) of each single item is also accumulated. By *transaction-weighted downward closure* (also abbreviated as TWDC) property, an item and its supersets are unpromising to be high utility itemsets if its also TWU is less than the minimum utility threshold. Such an item is called an unpromising item. An item is called a promising item if TWU >= min_util. Otherwise, it is called an un promising item. Without loss of generality, an item is also called a promising item if its *overestimated utility* is no less than min_util. Otherwise, it is called an unpromising item.

Consider an example, the transaction database in Table 1 and the profit table in Table 2. Suppose the minimum utility Threshold (*min_util)* is 50. In the 1st scan of database, Transaction Utility and the TWUs of the items are computed.{*F*} and {*G*} are unpromising items since their TWUs are less than *min_util*. The promising items are reorganized in header table in the descending order of TWU. Table 3 shows the reorganized transactions and their RTUs for the database in Table 1. As shown in Table 3, unpromising items {*F*} and {*G*} are removed from the transactions second, third and fifth Transactions (*T2*, *T3* and *T5*) respectively. The utilities of {*F*} and {*G*} are eliminated from the TUs of *T2*, *T3* and *T5*, respectively. The remaining promising items {*A*}, {*B*}, {*C*},{*D*} and {*E*} in the transaction are sorted in the descending order of TWU. Then, we insert reorganized transactions into the UP-Tree.

After pruning items, sort the items in descending order based on TWU value.

Table 3

ITEM	TWU
3	94
5	86
1	63
2	59
4	56

Re-Organized transactions and their RTUs
Table 4

TID	REORGANIZED TRANSACTION	RTU
T1	(1,1)(3,1)(4,1)	8
T2	(1,2)(3,6)(5,2)	22
T3	(1,1)(2,2)(3,1)(4,6)(5,1)	25
T4	(2,4)(3,3)(4,3)(5,1)	20
T5	(2,2)(3,2)(5,1)	9

From the above table 4, UP-Tree will be constructed.

R-root node, {3}:4,96 –{item}:count,TWU

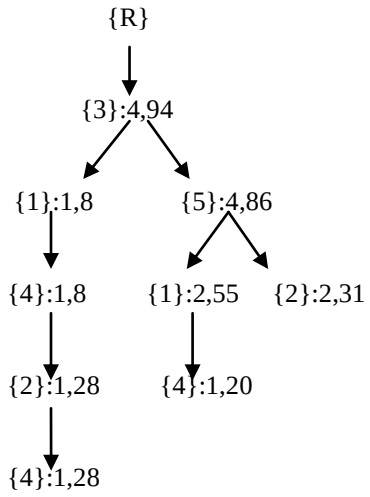


Figure 3. An UP-Tree when min_util=50

C. Strategy DGN: Decreasing Global Node Utilities

By actual utilities of descendant nodes during the construction of global UP-Tree we can decrease global node utilities. By applying strategy DGN, the utilities of the nodes that are closer to the root of a global UP-Tree are further reduced. DGN is especially suitable for the databases containing lots of long transactions. In other words, the more items a transaction contains, the more utilities can be discarded by DGN. On the contrary, traditional TWU mining model is not suitable for such databases since the more items a transaction.

Subroutine: Insert _Reorganized_Transaction(N,i_x)

Line 1: If N has a Child N_{ix} such that N_{ix}.item=i_x, increment N_{ix}.Count by 1. Otherwise, create a new child node N_{ix} with N_{ix}.item=i_x, N_{ix}.Count=1, N_{ix}.Parent=N and N_{ix}.nu=0.
Line 2: Increase N_{ix}.nu by (RTU(tj') - $\sum^{n}_{p=x+1}$ u(ip,tj')), where ip ϵ tj
Line 3: If x≠n, call Insert_Reorganized_Transaction (N_{ix} , i_x+1)

Figure3. Subroutine of Insert_Reoraganisation_Transaction

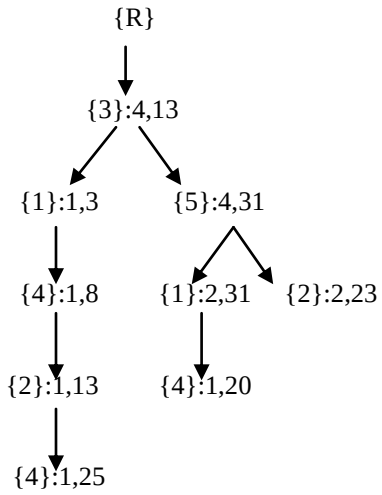


Figure 4. A UP-Tree by applying Strategies DGU and DGN

IV. IMPLEMENTATION AND EVALUATION

The experiments were performed on a Intel CORE i3 processor with 1.5 GB memory. The operating system is Windows 7. The algorithm is implemented in Java language. The construction of UP-Tree can be performed with two scans of database. In First scan, Transaction utility and Transaction Weighted Utility is accumulated. Input file, output file, minimum utility will be chosen. Input file consists of the items and their quantities in the transaction table. Profit table is also taken as input. Minimum utility value is taken is shown in figure 6. Output file consists of high utility items by eliminating unpromising items which are below the minimum threshold value. If the input is taken as transaction table and profit table (i.e) Table1 and Table 2. The Output is constructed by applying DGU and DGN strategies in Table 3. The output will be Up-Tree is shown in figure 4.

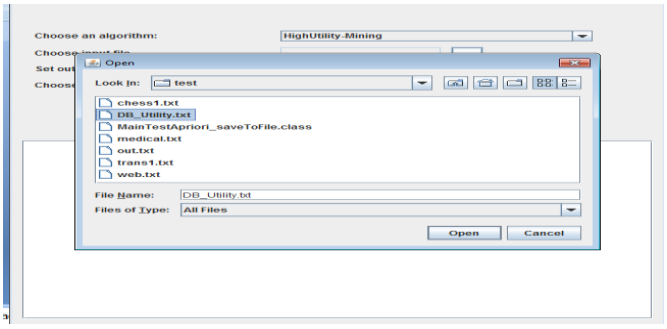


Figure 5. Input selection

A. UP-Tree construction:

Before constructing UP-Tree, utility for each item will be calculated. By using those item utilities, transaction utility can be calculated. Transaction weighted utility will be calculated for each item is shown in fig 7.

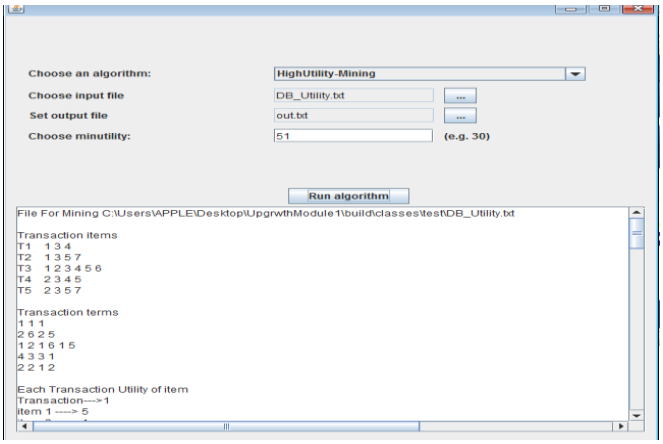


Figure 6. UP-Tree construction phase-I

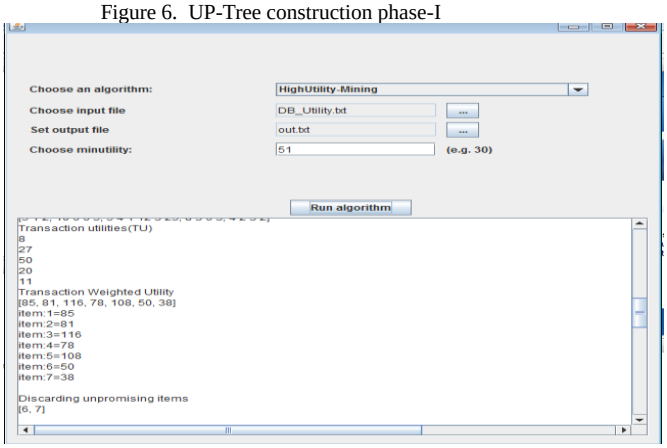


Figure 7. UP-Tree construction phase-II

By TWU value, promising items and unpromising items can be identified. After that, unpromising items will be discarded is shown in fig 8

.

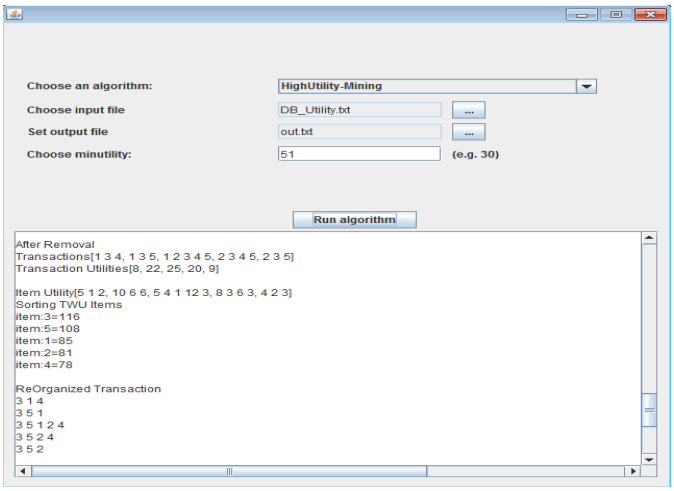


Figure 8. DGU Strategy

Sorting items based on TWU value (i.e) Table 3.
From that table3, up-tree will be constructed shown in fig 9.

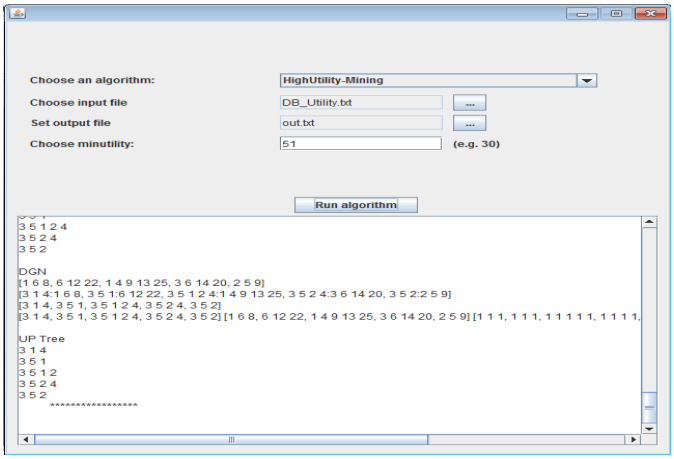


Figure 9. DGN Strategy

The transaction items are taken and UP-Tree is constructed by eliminating the unpromising items using DGU principles and it is implemented using Java language. Database scans are reduced by using UP-Growth algorithm.

V. CONCLUSION

In this paper, we have developed an efficient algorithm named UP-Growth for mining high utility itemsets from transaction databases. A data structure named UP-Tree is proposed for maintaining the information about Transaction database and nining high utility itemsets by applying DGU and DGN strategies in that UP-Tree. Hence, the potential high utility itemsets can be efficiently generated from the UP-Tree with only two scans of the database. Besides, we develop strategies to decrease the estimated utility value and enhance the mining performance in utility mining. The mining

performance is enhanced significantly since both the search space and the number of candidates are effectively reduced by the proposed strategies. Mining Frequent patterns using UP-Growth is more efficient than Apriori and FP-Growth.The experimental results show that UP-Growth outperforms the state-of-the-art algorithms significant, when the data source contains huge amount of transactions. By simulation results, we have shown that the high utility items for the Transaction database by constructing UP-Tree.

REFERENCES

[1]. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of the 20th Int'l Conf. on Very Large Data Bases*, pp. 487-499, 1994.

[2]. C. F. Ahmed, S. K. Tanbeer, B.-S. Jeong, and Y.-K. Lee. Efficient tree structures for high utility pattern mining in incremental databases. In *IEEE Transactions on Knowledge and Data Engineering*, Vol. 21, Issue 12, pp. 1708-1721, 2009.

[3]. R. Chan, Q. Yang, and Y. Shen. Mining high utility itemsets. In *Proc. Of Third IEEE Int'l Conf. on Data Mining*, pp. 19-26, Nov., 2003.

[4]. A. Erwin, R. P. Gopalan, and N. R. Achuthan. Efficient mining of high utility itemsets from large datasets. In *Proc. of PAKDD 2008,LNAI 5012,* pp. 554-561.

[5]. J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proc. of the ACM-SIGMOD Int'l Conf. on Management of Data*, pp. 1-12, 2000.

[6]. Y.-C. Li, J.-S. Yeh, and C.-C. Chang. Isolated items discarding strategy for discovering high utility itemsets. In *Data & Knowledge Engineering*, Vol. 64, Issue 1, pp. 198-217, Jan., 2008.

[7]. Y. Liu, W. Liao, and A. Choudhary. A fast high utility itemsets mining algorithm. In *Proc. of the Utility-Based Data Mining Workshop*, 2005.

[8]. B.-E. Shie, V. S. Tseng, and P. S. Yu. Online mining of temporal maximal utility itemsets from data streams. In *Proc. of the 25th Annual ACM Symposium on Applied Computing*, Switzerland, Mar., 2010.

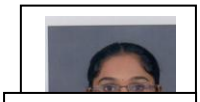
[9]. H. Yao, H. J. Hamilton, L. Geng, A unified framework for utility-based measures for mining itemsets. In *Proc. of ACM SIGKDD 2nd Workshop on Utility-Based Data Mining*, pp. 28-37, USA, Aug., 2006.

[10]. S.-J. Yen and Y.-S. Lee. Mining high utility quantitative association rules. In *Proc. of 9th Int'l Conf. on Data Warehousing and Knowledge Discovery, Lecture Notes in Computer Science 4654*, pp. 283-292, Sep.,2007

Authors Profile

C.Rajmohan received the **M.E.** degree in Information Technology from the Manonmaniam Sundaranar University, Tirunelveli, India. Currently doing **Ph.D** in Anna University, India. His research interest includes wireless communication, Data mining, Soft Computing.

C.Niveditha Currently doing **B.Tech.** in Information Technology in Sri Ramakrishna Engineering College, Coimbatore, India.



R.Pragathi Currently doing **B.Tech.** in Information Technology in Sri Ramakrishna Engineering College, Coimbatore, India.

G.Priya Currently doing **B.Tech.** in Information Technology in Sri Ramakrishna Engineering College, Coimbatore, India.