An Efficient Mining of Transactional Data Using Graph-based Technique

Wael Ahmad AlZoubi, Khairuddin Omar, Azuraliza Abu Bakar
Center for Artificial Intelligence Technology
Faculty of Computer and Information Technology
University Kebangsaan Malaysia
Wael.alzoubi@yahoo.com, ko@ftsm.ukm.my, aab@ftsm.ukm.my

Abstract – **Mining association rules is an essential task for knowledge discovery. Past transaction data can be analyzed to discover customer behaviors such that the quality of business decision can be improved. The approach of mining association rules focuses on discovering large itemsets, which are groups of items that appear together in an adequate number of transactions. In this paper, we propose a graph-based approach (DGARM) to generate Boolean association rules from a large database of customer transactions. This approach scans the database once to construct an association graph and then traverses the graph to generate all large itemsets. Practical evaluations show that the proposed algorithm outperforms other algorithms which need to make multiple passes over the database.**

Keywords: Apriori, clustering, graph, rule mining.

I. INTRODUCTION

Data mining has high applicability in the retail industry. The effective management of business is extensively dependent on the quality of its decision making. Therefore, it is important to improve the quality of business decisions by analyzing past transaction data to discover customer purchasing behaviors. In order to support this analysis, a sufficient amount of transactions needs to be collected and stored in a database [1]. Each transaction in the database consists of the items purchased in the transaction besides other information like transaction date and time, customer name, quantity, price, and other information. All what was taken in consideration is the set of items bought together in a transaction. Because the amount of these transactions' data can be very large, an efficient algorithm needs to be designed for discovering useful information from these huge transactional datasets.Mining frequent itemsets and association rules is a popular and well researched method for discovering interesting relations between variables in large databases [2]. Association rules, first introduced in 1993 [3], are used to identify relationships among a set of items in a database. These relationships are not based on inherent properties of the data themselves, but rather based on co-occurrence of the data items. Association rules are used to discover the relationships, and potential associations, of items or attributes among huge data [4]. These rules can be effective in uncovering unknown relationships, providing results that can be the basis of forecast and decision. They have proven to be very useful tools for an enterprise as it strives to improve its competitiveness and prosperity [4]. Association rule mining in relational database management systems generally transforms the database into (TID, item) format, where TID stands for a unique transaction identifier and item stands for different items purchased by the customers. There will be multiple entries for a given transaction ID, because one transaction ID indicates purchase of one particular customer and a customer can purchase as many items as he/ she want. Any association rule will hold if its support and confidence are equal to or greater than the user specified minimum support (S) and confidence (C). The association rules mining (ARM) problem is an NP Hard problem because finding all frequent itemsets (FIs) having a minimum support results in a search space of 2^m, which is exponential in m, where m is the number of itemsets [5]. The final step involves generating strong rules having a minimum confidence from the frequent itemsets. It also includes generating and testing the confidence of all rules. Since each subset of X as the consequent must be considered, ARM is computationally and I/O intensive. The number of rules grows exponentially with the number of items. Because data is increasing in terms of the dimensions (number of items) and size (number of transactions), one of the main attributes needed in an ARM algorithm is the ability to handle massive data stores. Traditional algorithms cannot provide such this ability, in terms of the data dimension, size, or runtime performance, for such large databases. This paper proposes a new algorithm called (DGARM) that employs both graph and clustering techniques to discover association rules. The graph technique reduces the database scans, while the clustering technique eliminates some candidate itemsets that cannot be frequent. The rest of this paper is organized as follows. In Section 2 we provide an overview of work related to the association rule mining problem. In Section 3 we discuss the proposed DGARM. In Section 4, an example on the proposed algorithm is presented. In Section 5 the analysis of the results is presented.

978-1-61284-212-7/11/$26.00©2011 IEEE

II. RELATED WORK

Agrawal and Srikant proposed the Apriori association rule algorithm [3]. Apriori discovered meaningful itemsets and constructed association rules within large databases, but the generation of candidate itemsets needs to perform contrasts against the whole database, level by level, in the process of creating association rules. Performance is considerably affected, as the database is repeatedly scanned to contrast each candidate itemset with the database. After Agrawal et al. proposed the Apriori association rule, most association rules researchers have used Apriori-like candidate generation approaches, all of these methods focus on reducing the number of candidate itemsets, and therefore reducing the number of database scans. Han et. al. [7] proposed a novel frequent-pattern tree (FP-tree) structure, which is an extended prefix-tree structure for storing compressed, crucial information about frequent patterns, and developed an efficient FP-tree based mining method, FP-growth, for mining

the complete set of frequent patterns by pattern fragment growth. FP-growth method is efficient and scalable for mining both long and short frequent patterns, and is about an order of magnitude faster than the Apriori algorithm and also faster than some recently reported new frequent-pattern mining methods. Tsay and Chiang [4] proposed an efficient cluster based association rule mining method (CBAR) for discovering the large itemsets, and the main characteristics are the following. CBAR only requires a single scan of the transaction database, followed by contrasts with the partial cluster tables. According to Tsay and Chiang, CBAR didn't only prune considerable amounts of data that leads to reducing the time needed to perform data scans and requiring less contrast, but also ensured the correctness of the mined results. The CBAR employed some efficient cluster tables to represent database D by a single scan of the database, followed by contrasts with the partial cluster tables. Figure 1 gives the algorithmic form of CBAR.

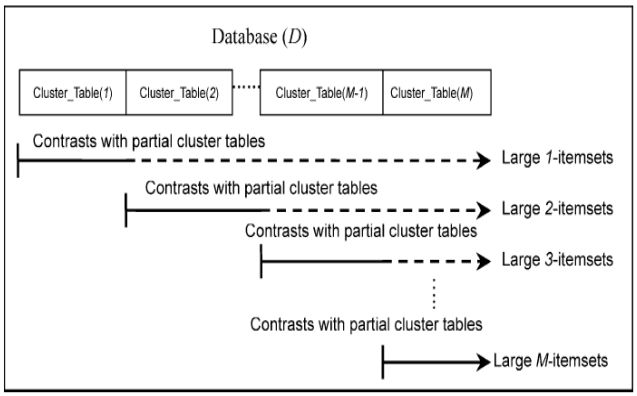


Figure 1: CBAR needs the contrasts with only partial cluster tables [4]

A graph-based approach to generate various types of association rules from a large database of customer transactions had been proposed [1]. This approach scans the database once to construct an association graph and then traverses the graph to generate all large itemsets. The association graph construction (AGC) algorithm had been applied to construct an association graph to generate primitive association patterns [8].

The AGC algorithm was described as following: For every two large items i and j. such that $i < j$, if the number of 1s in bit vector of item i (BV_i) and bit vector of item j (BV_j), i.e. $BV_i \wedge BV_j$ is not less than the user-specified minimum support, a directed edge from item i to item j is created. Also, itemset (i, j) is a large 2-itemset.

III. DYNAMIC GRAPH OPTIMIZATION AND RULE MINING PROBLEM

The proposed algorithm (DGARM) has two main advantages: one is the reduction of the database scans and the other is the elimination of candidate k-itemsets of order 3 and above. Figure 2 presents an overview of DGARM algorithm steps, where t_j is the j^{th} transaction and M is the total number of transactions.

978-1-61284-212-7/11/$26.00©2011 IEEE

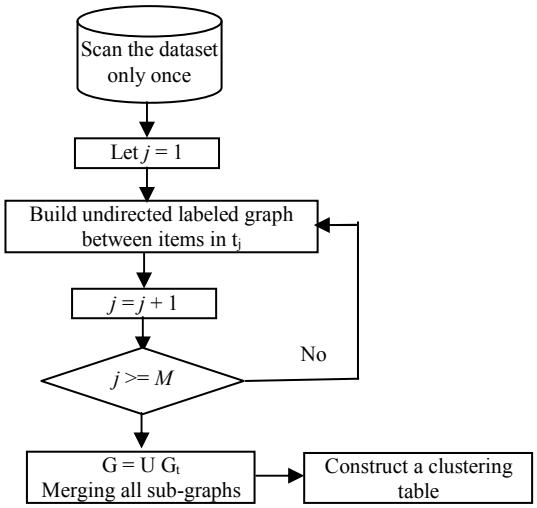


Figure 2: DGARM Steps

The DGARM for mining association rules is presented in figure 3. The algorithm scans the database once to build a graph of items and a clustering-table. This scan is enough to find frequent 1-itemsets and frequent 2-itemsets. There is no need to generate candidate 2-itemsets and hence no need to scan the database to discover frequent 2-itemsets. After that, DGARM works iteratively starting from frequent 2-itemsets (F_2) in the sense that frequent itemsets that are discovered in one iteration will be used as the basis to generate candidate itemsets in the next iteration. As will be described later, the candidate generation step is similar to that in the Apriori algorithm but here we employ clustering technique to eliminate some infrequent candidate itemsets.

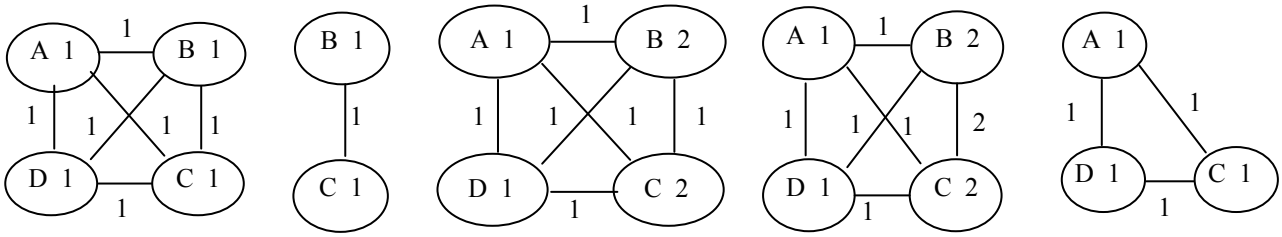
```
DGARM(int minsup)

G ← ∅
C1 = {set of all items}
for all transactions t ∈ D  do
   Build-Graph(G, t);
   Cluster(t, length(t));
GraphFrequent(G);
for (k = 3; Fk-1 ≠ ∅; k++)  do
   Ck = Gen-Candidate(Fk-1);
   for all transactions t ∈ D  do
      Ct = subset(Ck, t);
         for all candidate c ∈ Ct do
            c.count++;
   Fk = {c ∈ Ck | c.count ≥ minsup}
Answer = || Fk
Build-Graph(graph G, transaction t)
for each item i ∈ t  do
```

978-1-61284-212-7/11/$26.00©2011 IEEE

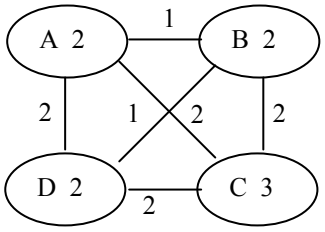
```
   V[G] ← V[G] ∪ {i}
   Count[i] = 1;
for each 2-subset itemset e ∈ t  do
   E[G] ← E[G] ∪ {e}
   Count[e] = 1;
if (there are similar vertices and edges)
   Merge(vertices and edges);
Cluster(transaction t, int n)
for each item i ∈ t  do
   Clustering-table[i][n]++;
GraphFrequent(graph G)
for each vertex v ∈ V[G]  do
   If (count[v] ≥ minsup) then
      F1 ← F1 ∪ {v};
for each edge e ∈ E[G] do
   If (count[e] ≥ minsup)then
      F2 ← F2 ∪ {e};
```

Figure 3: DGARM algorithm

The Build-Graph algorithm build a complete undirected graph $G = (V, E)$ using all transaction in the database. Initially, the graph G is the subgraph of the first transaction. For each transaction t in the database, the algorithm build a complete undirected subgraph $G_t = (V_t, E_t)$, where V_t is the set of all items in t and E_t is the set of all edges between every 2-subset itemsets in t. A counter is associated with each vertex or edge that stores the occurrences of that vertex or edge and is initialized to 1. After building the subgraph G_t, a new version of graph G is created by merging G and G_t. If there are any similar vertices and edges between G_t and G, their counters are summed up. Figure 4 shows an example of three transactions and their sub graphs.



$G = G_1$ (a) $G = G_2$ (b) Merging process (c) G (d) $G = G_3$ (e)



Final graph (f)

Figure 4: (a) T1= {A, B, C, D} (b) T2 = {B, C} (c) Merging T1 and T2 (d) G = G1 ∪ G2 (e) T3 = {A, C, D} (f) G = G1 ∪ G2 ∪ G3

978-1-61284-212-7/11/$26.00©2011 IEEE

After all transactions have been read, the graph is built. Vertices' counters hold the supports of the corresponding items. Edges' counters on the other hand hold the supports of the corresponding 2-subset itemsets. Finally, the function *Graph-Frequent* searches the graph to find frequent 1-itemsets and frequent 2-itemsets. The function *Graph-Frequent* traverses each vertex and edge in the graph, if the counter of a vertex is greater than or equal the minimum support then the corresponding item is inserted into the set of frequent 1-itemsets (F_1) and if the counter of an edge is greater than or equal the minimum support then the corresponding 2-subset itemset is inserted into the set of frequent 2-itemsets (F_2). In the clustering step, each transaction is clustered to the k-th cluster, where the length of a transaction is k. Meanwhile, a clustering table is built to count the occurrences of each item in each cluster. Figure 5 shows the clustering-table through reading the first three transactions shown in figure 4. Subsequently, the clustering-table will be used in the candidate generation.

	01	A B C D		
	C_1	**C_2**	**C_3**	**C_4**
A	0	0	0	1
B	0	0	0	1
C	0	0	0	1
D	0	0	0	1

(a)

	02	A B		
	C_1	**C_2**	**C_3**	**C_4**
A	0	1	0	1
B	0	1	0	1
C	0	0	0	1
D	0	0	0	1

(b)

	03	A C D		
	C_1	**C_2**	**C_3**	**C_4**
A	0	1	1	1
B	0	1	0	1
C	0	0	1	1
D	0	0	1	1

(c)

Figure 5: The clustering-table of (a) T1= {A, B, C, D} (b) T2 = {A, B} (c) T3 = {A, C, D}

C_1 is the cluster of length 1-transactions; C_2 is the cluster of length 2-transactions; and so on.

The Gen-*Candidate* function takes as argument F_{k-1}, the set of frequent k-1-itemsets, and returns the set of all candidate k-itemsets. The function performs two steps, namely, *join* step and *prune* step (figure 6). In the join step, two different k-1-itemsets are joined to generate k-itemset if their first k-2 items are common. In the prune step, two tests are performed. In the first test, all candidate itemsets, that have k-1-subset which is not in frequent k-1-itemsets, are deleted. In the second test, all candidate itemsets, that have an item such that the sum of occurrences of that item in cluster k to cluster m is less than the minimum support, are deleted.

```
Gen-Candidate(frequent set F_{k-1})
for all itemset p ∈ F_{k-1}   do
   for all itemset q ∈ F_{k-1}   do
      If(p[1]=q[1], …, p[k-2]=q[k-
2],p[k-1]<q[k-1])then
         c = (p[1], …, p[k-1],q[k-1]);
         If Not-Prune(c, F_{k-1}) then
            Add c to C_k;
         Else
            delete c;
Not-Prune(candidate itemset c, frequent
set Fk-1)
for all k-1-subset s ∈ c  do
   if (s ∈ F_{k-1})then
      return false;
for all item a ∈ c
   if ( Test-Cluster(a, k) < minsup)
then
      return false;
return true;
Test-Cluster(item a, int k)
Sum = 0;
for i = k to m do
   sum = sum + clustering-table[a][i];
return sum;
```

Figure 6: Gen candidate algorithm

978-1-61284-212-7/11/$26.00©2011 IEEE

IV. ASSOCIATION RULE GENERATION

Once the frequent itemsets have been found, generating interesting association rules is a straightforward step. Interesting association rules are the ones that satisfy the minimum support and the minimum confidence. The computing of support and confidence was mentioned in equation 1 and 2 in section 1. Figure 7 shows the algorithm for association rules generation.

```
Generate-Rule(float minconf)
for all frequent itemset f   do
   for all nonempty subset s of f   do
If ( support(f) / support(s) > minconf)  then
 Add "s ⇒ (f-s)" to the set of rules
```

Figure 7: Generate rule algorithm

V. EXPERIMENTS

This section introduces a comparison between our proposed DGARM algorithm and some association rules algorithms using different synthetic datasets. The association rules algorithms that we used in the comparison are Apriori [3] and FP-growth [7]. The experiments were run on Pentium IV computer with a clock rate of 1600 MHz and 256 Mbytes of main memory. The dataset used in experiments is retail real dataset [9] that consists of 93438 transactions with an average size of 10. The size of the dataset is 1.5 MB.

Figure 8 shows the execution time of DGARM algorithm on retail dataset with different support thresholds. As the minimum support decreases, the execution time of DGARM increases due to an increase in the total number of candidate and frequent itemsets. The dataset used was T10I4D65K.

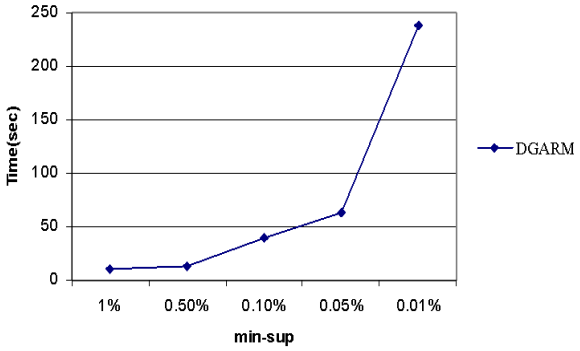


Figure 8: Execution time for DGARM at different minimum supports

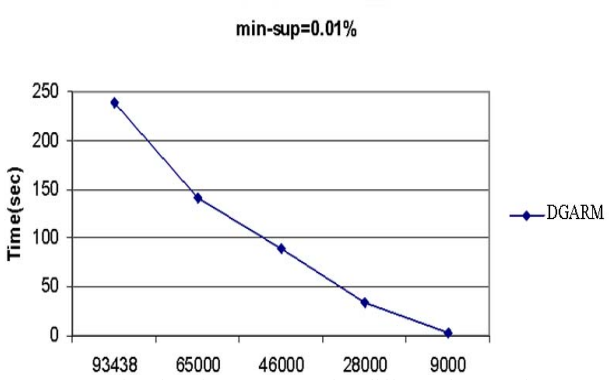


Figure 9: Execution time for DGARM using different datasets sizes

Figure 9 shows the execution time of DGARM algorithm with different number of database transactions at *min-sup* = 0.01%. As shown, the execution time scales quite linearly with the number of transactions from 9000 to 90000.

To assess the performance of the clustering technique that aims to eliminate some candidate itemsets that cannot be frequent, we present in table 1 the number of candidate itemsets, the number of pruned candidate itemsets using our clustering technique, and the number of infrequent itemsets at each database pass, where minsup = 0.01%.

978-1-61284-212-7/11/$26.00©2011 IEEE

TABLE 1: NUMBER OF ITEMSETS AT EACH DATABASE PASS

Pass	# of candidate itemsets	# of pruned itemsets (clustering technique)	# of infrequent itemsets
1	199175	220	178170
2	19856	560	7593
3	3977	220	931
4	347	0	40
5	4	0	4

As shown in table 1, the clustering technique deletes some candidate itemsets in database passes 3, 4 and 5. For example, in pass 5, quarter of the candidate itemsets are deleted without being examined in the next pass.

A comparison between the proposed DGARM algorithm, Apriori algorithm [3], and FP-growth algorithm [7] had been made using synthetic datasets.

We used four synthetic datasets that were generated as described in [9]. These synthetic datasets are widely used for evaluating association rules algorithms. Table 2 shows the names and descriptions of parameters used to generate the different datasets. For the four datasets used in the experiments N was set to 1000 and |L| was set to 2000.

TABLE 2: PARAMETERS

$\|D\|$	Number of transactions
$\|T\|$	Average size of transactions
$\|I\|$	Average size of maximal potentially large itemsets
$\|L\|$	Number of maximal potentially large itemsets
N	Number of items

Figure 10 and figure 11 respectively show the execution time for the first two synthetic datasets T10I4D65K and T10I4D100K, respectively. It is easy to see that DGARM and FP-growth beat Apriori for all minimum supports. FP-growth is comparable with DGARM as the minimum support increases, but it scales much better than DGARM as the minimum support goes down because the number as well as the length of frequent itemsets increases dramatically.

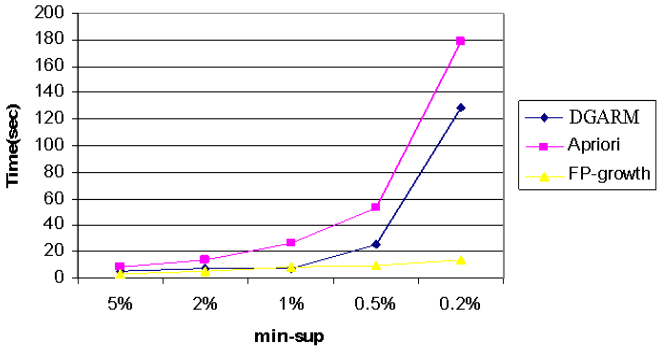


Figure 10: Execution time using (T10I4D65K) synthetic dataset

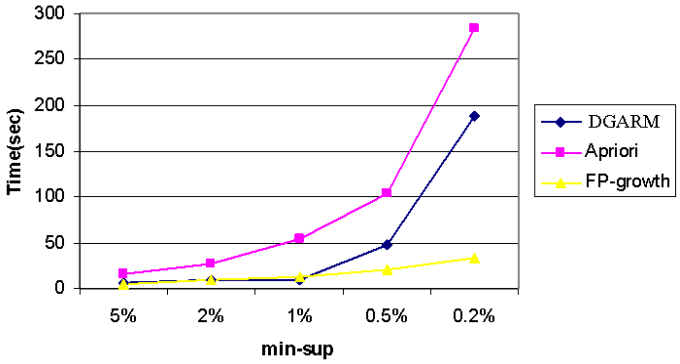


Figure 11: Execution time using (T10I4D100K) synthetic dataset

However, when we decrease the minimum support furthermore, FP-growth turns to be faster than DGARM. The reason behind this is that reducing the minimum support will produce a large number of frequent itemsets that requires DGARM to scan the database many times. Carrying out more experiments shows that the execution time using the synthetic dataset T40I10D100K. From this figure and as the minimum support reduces, one can notice that both FP-growth and Apriori algorithm are slower than DGARM. The reason behind this is that as the minimum support decreases the number of frequent itemsets increase. Thus, FP-tree has much more distinct itemsets and leads to larger FP-tree. Also, the running memory requirement of FP-growth is a mixture of the main memory and secondary memory. It has been noticed that the usage of the secondary memory degrades the performance of FP-growth.

VI. CONCLUSION

Through the experiments, we have seen that the performance of the proposed DGARM algorithm is better than Apriori algorithm, especially, when the minimum support is reduced. Also, extensive experiments have shown that a better

978-1-61284-212-7/11/$26.00©2011 IEEE

performance can be achieved than FP-growth, especially, when the size of database is large.

ACKNOWLEDGEMENT

This work has been supported by 01-01-02-SF0598.

VII. REFERENCES

1. Show-Jane Yen and Arbee L.P. Chen. A Graph-Based Approach for Discovering Various Types of Association Rules. IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 13, NO. 5, SEPTEMBER/OCTOBER 2001, pp. 839 – 845.
2. Petra Perner (Ed.). Advances in data mining, Medical Applications, E-Commerce, Marketing, and Theoretical Aspects, Book. 8th industrial Conference, ICDM 2008, Leipzig, Germany, July 2008, p 192.
3. R. Agrawal, R. Srikant, Fast algorithm for mining association rules in large databases, Proceedings of 1994 International Conference on VLDB, 1994 pp. 487–499.
4. Yuh-Jiuan Tsay, Jiunn-Yann Chiang. CBAR: an efficient method for mining association rules. Knowledge-Based Systems 18 (2005) 99–105.
5. Peter P. Wakabi-Waiswa and Venansius Baryamureeba. Extraction of Interesting Association Rules Using Genetic Algorithms. International Journal of Computing and ICT Research, Vol. 2, No. 1, pp. 26 – 33.
6. Michael Hahsler Bettina Grün Kurt Hornik. Introduction to arules - A computational environment for mining association rules and frequent item sets. Journal of Statistical Software. October 2005, Volume 14, Issue 15.
7. Jiawei Han , Jian Pei , Yiwen Yin , Runying Mao. Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach. Data Mining and Knowledge Discovery, 8, 53–87, 2004.
8. S.J. Yen and A.L.P. Chen, An Efficient Approach to Discovery Knowledge from Large Databases, Proc. IEEE/ACM Int'l Conf. Parallel and Distributed Information Systems, pp. 8-18, 1996.
9. Frequent itemset mining dataset repository. Available from URL http://fimi.cs.helsinki.fi/data/ 2010.
10. Wael AlZoubi, Khairuddin Omar, Azuraliza Abu Bakar. Scalable and Efficient Technique for Mining Association Rules. 2009 International Conference on Electrical Engineering and Informatics. 5-7 August 2009, Selangor, Malaysia, pp. 36 − 41.

978-1-61284-212-7/11/$26.00©2011 IEEE