TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol.12, No.4, April 2014, pp. 2941 ~ 2949
DOI: http://dx.doi.org/10.11591/telkomnika.v12i4.4810

A High Efficient Association Rule Mining Algorithm Based on Intelligent Computation

Wu Fengxiang
North China Career Academy of water Resources, HenanZhengzhou, China
E-mail: yancaifeng2002@163.com

Abstract

Data mining is to use automated data analysis techniques to uncover previously undetected relationships among data items. In data mining, association rule mining is a prevalent and well researched method for discovering useful relations between variables in large databases. In this paper, we investigate the principle of Apriori, direct hash and pruning and also study the drawback of them. The first is constructing hash table without confliction is theoretically optimal, but it needs consume a lot of memory space and space utilization is low. The second is that it does not have hash tree data structure leading to too long insert and search time. So we propose a new association rule mining algorithm based on differential evolutionary computation. The experiment results show that our proposed algorithm has better execution time and accuracy, which can be used in electronic commerce system.

Keywords: *apriori, association rule, direct hash and prunning, differential evolutionary computation*

1. Introduction

Nowadays, with the rapid development of information technology, especially the web service-based application, service-oriented architecture and cloud-computing, continually expanding data are integrated to generate useful information. There are many databases and data warehouses available all around the world. The core task is to make good use of the information or knowledge from these databases. Implicit knowledge of the databases can provide important patterns like association rules which may lead to decision support making, medical diagnosis and many other applications. Association rule mining is task of finding interesting association or correlation relationships among large databases [1]. Association rules are thought to be interesting as well as useful if they meet both a minimum support threshold and a minimum confidence threshold [2, 3]. A major concern in association rules mining today is to continue to improve algorithm performance.

Association rules can be defined by formal definition as Let $I=\{i_1,i_2,\ldots,i_m\}$ be a set of items. Let D be a set of database transactions where each transaction T is a set of items such that $T\subseteq I$. Each transaction must have an identifier, called TID. Let A be a set of items. A transaction T is said to contain A if and only if $A\subseteq T$. An association rule is of the form $A\Rightarrow B$, where $A\subset I$, $B\subset I$ and $A\cap B=\varnothing$.

Association rule mining [8-12] has attracted a lot of intention in research area of data mining and generation of association rules is completely dependent on finding frequent item sets. Many algorithms are available for this purpose. In [1], an algorithm for frequent item set generation is proposed, which is the first algorithm being available to us for frequent item set discovery and is known as AIS algorithm.

The Apriori-based algorithms find frequent item sets based upon an iterative bottom-up method to generate candidate item sets. Since the first proposal of association rules mining by R. Agrawal [1, 2], many researches have been done to make frequent item sets mining scalable and efficient. But there are still some deficiencies that Apriori-based algorithms suffered from, which include too many scans of the transaction database when seeking frequent item sets, large amount of candidate item sets generated unnecessarily and so on. In [4], an algorithm has been suggested by using bottom up method along with using matrix and reduced transactions. In [5], a new algorithm has been given for reducing the candidate item sets by reducing the

Received September 6, 2013; Revised November 10, 2013; Accepted November 23, 2013

connecting item sets i.e. For large database, this optimized algorithm can save cost as well as time and hence increase the efficiency than the Apriori algorithm. In [6], a method has been proposed to improve the efficiency of Apriori Algorithm using transaction reduction.

In the next section, we introduce principle of Apriori, direct hash and pruning. In Section 3 we propose a efficient algorithm based on differential evolutionary computation [19-22]. In Section 4, we test the performance of three algorithms. In Section 5 we conclude the paper and give some remarks.

2. Principle of Association Rule Mining

The process of commonly used Apriori algorithm is as follows:

input: Database, D, of transactions; minimum support threshold, min_sup.
output: L, freuqent itemsets in D.
C_k: Candidate itemset of size k
L_k:frequent itemset of size k
L_1 = find_frequent_1_itemsets(D);
for (k = 2; L_{k+1} !=∅; k++) do begin{
C_k = apriori_gen(L_{k-1}, min_sup);
for each transaction t in database D do{scan D for counts
Ct =subset(C_k, t); get the subsets of t that are candidates For each candidate c∈ C_t
c.count++;
}
L_k = candidates in C_k with min_support
}end
return L=∪$_k$ L_k;

In every scanning of finding association rules, frequent item set L_i is needed to generate candidate item set C_{i+1}, meaning combine two L_i frequent item sets($L_1 \cdot L_1$) with $i-1$ number of common items. Then scan database and statistic support degree of each item set in C_{i+1} to determine L_{i+1}. Larger number of item set in C_i results in higher cost to determine L_i. In order to understand the principle of Direct Hash and Pruning, we give out an example that generates a candidate 2-itemset by means of DHP algorithm. A bucket is made up of the following three parts. Bucket number is corresponding value of Hash function. The second part is element stored in the Hash table. The third part is the number of cell elements in the table. Hash table is made up of the third part of all the buckets. All the buckets consists a bit vector. Value of each bit in the bit vector is relevant to the number of element in the bucket. If the number is greater than or equal to $\min\sup$, it is 1, otherwise it is 0.

Table 1. Transaction Database Example

TID	ITEMS
100	ACD
200	BCE
300	ABCE
400	BE

For database in Table 1, the premise condition of algorithm is minimal support degree is 2 and hash function is (1).

$$h\{x,y\} = ((order\quad of\quad x)\cdot 10 + (order\quad of\quad y)) \bmod 7 \cdot \tag{1}$$

$order\quad of\quad x$ is the sequence number of x in all value sequences. Such as transaction item of the database is A, B, C, D and E and order of A is 1, order of D is 4. Firstly generate candidate 1-itemset, that is $C_1 = \{\{A\},\{B\},\{C\},\{D\},\{E\}\}$. Secondly scan all

transactions in the database, statistic support degree of all these candidate 1-itemset to generate L_1. At the same time C_1 sets up hash table H_2 for fast statistics. In order to construct candidate 2-itemset C_2 to set up H_2, database is decomposed as shown in Table 2. For 2-itemset {A C}, substitute into hash function and obtain:

$$\begin{aligned} h\{\{A\ \ C\}\} &= ((order\ \ of\ \ A)\cdot 10+(order\ \ of\ \ C)) \bmod 7 \\ &= (1\cdot 10+3) \bmod 7 \\ &= 6 \end{aligned} \tag{2}$$

Direct hash and pruning detect each item to determine whether it is in the hash table. If it is in the hash table, value of count of this item is increased by 1. Otherwise it put this item in the hash table and value of count is set to 1. Schematic diagram of hash table H_2 is shown in Table 3.

Table 2. Database Decomposition

100	{A C},{A D},{C,D}
200	{B C},{B E},{C,E}
300	{A B},{A C},{A E},{B C},{B E},{C E}
400	{B E}

Table 3. Database Decomposition

Hash value	0	1	2	3	4	5	6
element	{CE}{CE}{AD}	{AE}	{BC}{BC}		{BE}{BE}{BE}	{AB}	{AC}{CD}{AC}
Number of element	3	1	2	0	3	1	3

Because minimum support is 2, bit vector is <1, 0, 1, 0, 1, 0, 1> and $L_1 = \{\{A\}\{B\}\{C\}\{E\}\}$. Then do $L_1 \cdot L_1$ and obtain $L_1 \cdot L_1 = \{\{AB\}\{AC\}\{AE\}\{BC\}\{BE\}\{CE\}\}$.

Substitute 2-itemset in $L_1 \cdot L_1$ into hash function to obtain hash address of each 2-itemset. For each transaction, when all 1-itemsets statistics completely, all 2-itemsets of this transaction generate. Then according to value of bit vector, filter 2-itemsets from $L_1 \cdot L_1$ and 2-itemsets whose bit vector is 0 are deleted. At last we obtain $C_2 = \{\{AC\},\{BC\},\{BE\},\{CE\}\}$.

3. An Improved Scheme Based on Differential Evolutionary Algorithm

3.1. Principle of Differential Evolutionary Algorithm

In essence, direct hash and pruning uses association operation that is the same as the method that Apriori algorithm uses to determine candidate item sets [13-18]. There are two problems. The first problem is that constructing hash table without confliction is theoretically optimal, but it needs consume a lot of memory space and space utilization is low. The second is that it does not have hash tree data structure leading to too long insert and search time.

Direct hash and pruning owns fast exaction time at the cost of calculating hash table and storage space of the database table. This requires that database can be stored in memory. If the database is too large, it can not be placed in memory. It will waste a lot of time reading from the disk and cost of establishing hash table is expensive. Through the analysis of degree of database pruning of direct hash pruning algorithm, we find that pruning technology can further strengthen, so that we proposed an improved algorithm based on direct hash pruning. Direct hash and pruning improves efficiency through pruning of candidate item set. In order to deal with great database, direct hash and pruning is combined with random sampling technology to improve application range and execution efficiency. Data in the database is distributed randomly, meaning each transaction in the database perhaps appears in the any record of database. Because in the beginning, there are many data items which make another

item set from each transaction many. This increases the possibility of confliction of hash function, which makes performance of direct hash and pruning worse. To reduce the cost of establishing Hash table, and reduce the execution time, database can be dealt with principle of random sampling. In theory, the method of discovering association rules from samples exists a problem and there is a balance between time and accuracy. Since there is no search based on the whole data, some information may be lost. From implementation results of multiple databases, as long as the number of sample and selection of relaxation factor is appropriate and get good results.

Differential evolutionary computation is one of the currently popular intelligent population evolutionary algorithm. Its solving efficiency is high and has a better solution space search performance and strong robustness, so using differential evolutionary computation to solve the problem of association rule extraction is feasible.

Differential evolution algorithm produced in 1995, which is proposed by Rainer Storn and KennethPrice in the United States on the basis of evolutionary ideas such as genetic algorithm. This algorithm is also a kind of bionic intelligent algorithm stimulating natural biological evolution mechanism. It is the main operating thought is based on individual difference degree of the population generate temporary individual, and use one-to-one greedy selection competition mechanism to realize population evolution through the random restructuring. The concrete process is as follows:

Step 1. Population random initialize. $rand(0,1)$ returns random number of $(0,1)$, $t=0$.

$$P(0)=\{X_i(0)\,|\,x_{ij}(0)=rand(0,1)\cdot(U_j-L_j)+L_j\Lambda 1\le j\le n\}. \tag{3}$$

Step 2. It is the mutation process. Choose three individuals $X_a(t), X_b(t), X_c(t)$ random in population $P(t)$. Three individuals are different with $X_i(t)$.Operate with the follow expression.

$$D_i(t+1)=(d_{i1}(t+1),d_{i2}(t+1),\ldots,d_{in}(t+1)). \tag{4}$$

$$d_{ij}(t+1)=x_{aj}(t)+F\cdot(x_{bj}(t)-x_{cj}(t)). \tag{5}$$

$1\le i\le s, 1\le j\le n$.

Step 3. It is the crossover process. r_1, r_2 is random value of between 0 and 1. Calculate temporary individual according to (6) and (7).

$$E_i(t+1)=(e_{i1}(t+1),e_{i2}(t+1),\ldots,e_{in}(t+1)). \tag{6}$$

$$e_{ij}(t+1)=\begin{cases} d_{ij}(t+1), & r_1\le C \\ x_{ij}(t), & r_1>C \end{cases}. \tag{7}$$

Step 4. It is the selection process according to (8).

$$X_i(t+1)=\begin{cases} E_i(t+1), & f(E_i(t+1))<f(X_i(t)) \\ X_i(t), & else \end{cases}. \tag{8}$$

Step 5. Calculate individual with the smallest fitness $X_b(t+1)$ in $P(t+1)$. If meet the termination condition, output X_b and $f(X_b)$. Otherwise $t=t+1$ and turn back to step 2 to continue. $F\in(0,2)$ is perturbation scale factor and $C\in(0,1)$ is cross factor.

3.2. An Efficient Association Rule Mining Algorithm Based on Differential Evolutionary Computation

How the original solution is represented by differential evolution algorithm coding form is the most important link. Individual of the algorithm is set into a binary code structure, which represents decision attribute and task attribute structure containing item sets and before and after part. $\{A_1, A_2, \cdots, A_n, B_1, B_2, \cdots B_m\}$, A_i is decision attribute and B_i is task attribute. The individual coding is $\{x_1, x_2, \cdots, x_n, b_1, b_2, \cdots b_m\}$. $x_1, x_2, \cdots, x_n$ is decision attribute and $b_1, b_2, \cdots b_m$ is task attribute. Attribute population and rule population is generated. Attribute population is used to generate frequent item sets and rule population is used to obtain association rule. Frequent item sets are generated by attribute population and the fitness function that choose high support degree and abandon low support degree is shown in (9).

$$f_1(R) = w\,\mathrm{supp}(R) = \sum_{i_j \in X \cup Y} w_j(\sup(R) / len(R)) \cdot \tag{9}$$

For rule population used to generate association rule, screening is done according to credibility of generated rule and the corresponding fitness function is (10):

$$f_2(R) = \alpha\, conf(R) + \beta\, \mathrm{int}\, er(R) \cdot \tag{10}$$

$$conf(R) = \frac{\sup(R)}{\sup(R_c)} . \tag{11}$$

$$\mathrm{int}\, er(R) = \frac{conf(R) - \sup(R_0)}{\max\{conf(R), \sup(R_0)\}} . \tag{12}$$

$1 \le \alpha, 1 \le \beta$. R represents rule and $len(R)$ represents the length of rule. $\mathrm{int}\, er(R)$ is interestingness degree of rule, which is a number more than 0. R_c is data set which represents successful matching between rule R and decision attribute in the R, R_0 is data set which represents successful matching between rule R and task attribute in the R. The process of our proposed algorithm is as follows:

Step 1. Initialize the population.

Step 2. x^k is transformed to $cx^k \in [0,1]$ according to (13).

$$cx_j^k = (x_j^k - x_{\min, j}) / (x_j^k - x_{\max, j}) . \tag{13}$$

$j = 1, 2, \cdots n, k = 0$.

Step 3. Calculate the next individual according to (14) and (15).

$$cx_j^{k+1} = 4 \cdot cx_j^k (1 - cx_j^k) . \tag{14}$$

$$x_j^{k+1} = x_{\min, j} + cx_j^{k+1} (x_{\max, j} - x_{\min, j}) . \tag{15}$$

Step 4. If $f(x_j^{k+1}) < f(x_0)$, the program stops. Otherwise it turns to step 2 to go on running.

Step 5. Keep 80% excellent individuals and abandon other individuals to generate new population.

Step 6. For the new population, repeat step 2 to step 4. If it meets the terminal condition, record current best individual X_{best} and stop the program. Otherwise do as (16) and (17). Randomly generate 50% individuals in $[x_{\min,j}, x_{\max,j}]$ and turn to step 2 to go on.

$$X_{\min,j} = \max\{x_{\min,j}, x_{bset,j} - r(x_{\max,j} - x_{\min,j})\}. \quad (16)$$

$$X_{\max,j} = \min\{x_{\max,j}, x_{bset,j} - r(x_{\max,j} - x_{\min,j})\}. \quad (17)$$

$0 < r < 1.$

4. Experiment and Analysis

In this experiment, hash table H_k with k+1-item set is generated. Transaction $t = ABCDEF$. Hash table established by C_2 contains five number of 2-item set (AC, AE, AF, CD, EF). According to our proposed algorithm, A, C, E and F are kept and B and D are deleted. t is labeled as t'. It can be seen that not all the items in t can be used to generate item set. In fact, C does not belong to any item set, because only AC and CD are 2-candidate item sets. So C is deleted from t'.

Execution time comparison of three algorithm is shown in Table 4. Execution time comparison of three algorithms under the same support and different database is shown in Figure 1. Execution time comparison of three algorithms under different support and the same database is shown in Table 5. Execution time comparison of three algorithms under different support and the same database is shown in Figure 2. The blue line represents Apriori algorithm, the red line represents direct hash and pruning algorithm DHP, and the yellow line represents our proposed algorithm based on differential evolutionary IDE. It can be seen that execution time of our proposed algorithm based on differential evolutionary is small. Execution time comparison of generating k frequent set of three algorithms under the same database and the same minimal support is shown in Table 6 and Figure 3. Taking database of 50000 line for example, execution time of IDE and DHP is much smaller than that of Apriori. But 1-frequent item set, execution time of IDE and DHP is longer than that of Apriori. Because Apriori has the best performance at the first iteration and DHP needs to construct hash table at the first iteration.

Table 4. The Execution Time of the Three Algorithms

Data size(line)	Execution time(s) Apriori	DHP	IDE
4000	73	37	28
5000	91	45	38
6000	114	55	46
8000	153	72	60
10000	192	94	72
15000	285	133	94
20000	324	185	124
30000	572	277	156
50000	955	472	206
100000	1833	954	264
150000	2765	1422	332

These three algorithms are used in the electronic commerce recommendation system, aiming to find the association mode and knowledge between browsed WEB pages in trading things. It can predict user interesting pages and analysis users' preference to contribute to the relatively large size of purchase. Experimental data comes from logs of WEB server of an enterprise WEB containing 726 WEB pages. Five weeks visit records are extracted and 6200 user transactions are obtained after data preprocessing. Relation between execution time and minimum support degree is shown in Figure 4 and relation between execution time and the number of transaction is shown in Figure 5. In Figure 4, the red line represents Apriori algorithm,

and yellow line represents direct hash and pruning algorithm and the blue line represents proposed algorithm IDE. In Figure 5, the yellow line represents Apriori algorithm, and red line represents direct hash and pruning algorithm and the blue line represents proposed algorithm IDE. It can be seen that running time of each algorithm becomes smaller with the increase of minimum support. When the minimum support is the same, running time of Apriori is longest, running time of DHP is middle and running time of IDE is shortest. Running time of each algorithm becomes larger with the increase of number of transaction and running time of our proposed algorithm is shortest. Efficiency of proposed algorithm based on differential evolutionary computation is the best, which can be used in electronic commerce system.

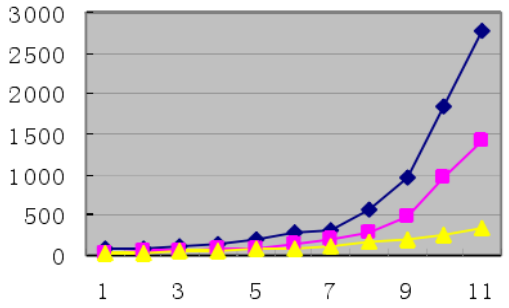

Figure 1. Execution Time Comparison of Three Algorithms under the Same Support and Different Database

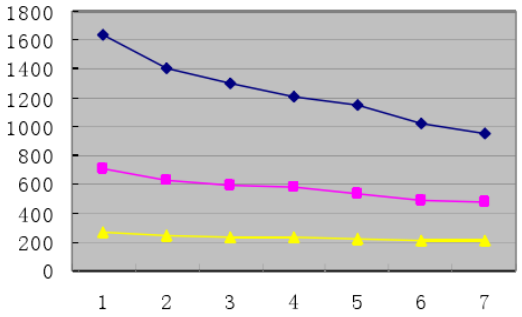

Figure 2. Execution Time Comparison of Three Algorithms under Different Support and the Same Database

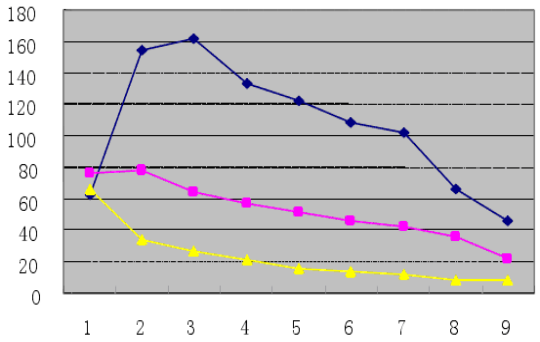

Figure 3. Execution Time Comparison of Generating k Frequent Set of Three Algorithms

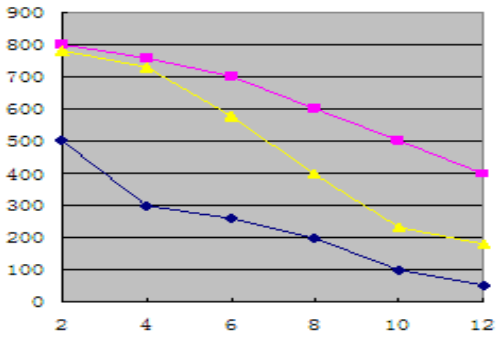

Figure 4. Relation between Execution Time and Minimum Support

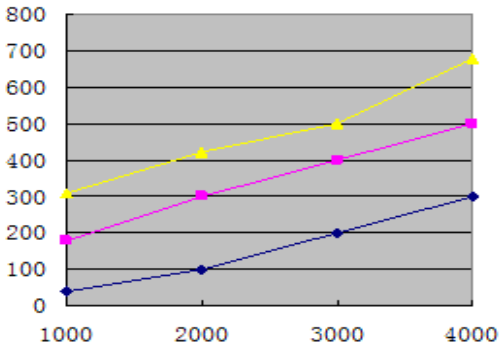

Figure 5. Relation between Execution Time and the Number of Transaction

Table 5. Execution Time Comparison of Three Algorithms under Different Support and the Same Database

Minimal support	Execution time(s) Apriori	DHP	IDE
0.5%	1643	704	272
1%	1408	632	248
1.5%	1296	598	236
2%	1205	576	234
3%	1148	532	218
5%	1024	486	210
6%	955	472	206

Table 6. Execution Time Comparison of Generating k Frequent Set

The number of frequent item sets k	Execution time(s) Apriori	DHP	IDE
1	62	76	66
2	154	78	34
3	162	64	27
4	133	57	21
5	122	51	16
6	108	46	14
7	102	42	12
8	66	36	8
9	46	22	8

5. Conclusion

This paper investigates the principle of Apriori, direct hash and pruning and also studies their drawbacks. Then we propose an improved algorithm based on differential evolutionary algorithm. The experiment results show that our proposed algorithm has better execution time and accuracy and proposed algorithm based on differential evolutionary computation is can be used in electronic commerce system.

References

[1] Rakesh Agrawal, Tomasz Imielinski, Arun Swami. *Mining Association Rules between Sets of Items in Large Databases.* Proceedings of the 1993 ACM SIGMOD Conference Washington DC, USA. 1993.

[2] Jiawei Han, Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, Champaign: CS497JH, fall 2001.

[3] Rakesh Agrawal, Ramakrishnan Srikant. *Fast Algorithms for Mining Association Rules.* Proceedings of the 20th VLDB Conference Santiago, Chile. 1994.

[4] Sunil Kumar, Shyam Karanth, Akshay K, Ananth Prabhu, Bharathraj Kumar M. Improved Apriori Algorithm Based on bottom upapproach using Probability and Matrix*.* 2012; 9(2): 1694-0814.

[5] Jiao Yabing. Research of an Improved Apriori Algorithm in Data Mining Association Rules. *International Journal of Computer and Communication Engineering*. 2013; 2(1).

[6] Jaishree Singh, Hari Ram, Dr JS Sodhi. Improving Efficiency of Apriori Algorithm using Transaction Reduction. *International Journal of Scientific and Research Publications.* 2013; 3(1), ISSN 2250-3153.

[7] Anurag Choubey, Ravindra Patel, JL Rana. A Survey of Efficient Algorithms and New Approach for Fast Discovery of Freqent itemset for Association Rule Mining. *IJSCE*, ISSN: 2231-2307, 2011; 1(2).

[8] AMJ Md Zubair Rahman, P Balasubramanie, P Venkata Krihsna. A Hash based Mining Algorithm for Maximal Frequent Itemsets using Linear Probing. *Infocomp Journal of Computer Science*. 2009; 8(1): 14-19.

[9] Y LIU, B YANG. Research of an Improved Apriori Algorithm in Mining Association Rules. *Journal of Computer Applications*. 2007; 27: 418-420.

[10] Lei Ji, Baowen Zhang, Jianhua Li. *A New Improvement on Apriori Algorithm.* Computational Intelligence and Security, International Conference. 2006; 1: 840-844.

[11] Sujni Paul, V Saravanan. *Hash Partitioned Apriori in Parallel and Distributed Data Mining Environment with Dynamic Data Allocation Approach.* Computer Science and Information Technology, ICCSIT'08. International Conference. 2008: 481-485.

[12] Bo Wu, Defu Zhang, Qihua Lan, Jiemin Zheng. *An Efficient Frequent Patterns Mining Algorithm based on Apriori Algorithm and the FP-tree Structure.* Convergence and Hybrid Information Technology, ICCIT'08. Third International Conference. 2008; 1: 1099-1102.

[13] Somboon Anekritmongkol, Kulthon Kasemsan. SQL Model in Language Encapsulation and Compression Technique for Association Rules Mining*. IJIPM*l. 2013; 4(1): 65-75.

[14] Naili Liu, Lei Ma. Improved algorithm for mining frequent itemsets based on binary*. International Journal of Advancements in Computing Technology*. 2013; 5(9): 1077-1084.
[15] Mohamad Farhan Mohamad Mohsin, Mohd Helmy Abd Wahab, Mohd Fairuz Zaiyadi, Cik Fazilah Hibadullah. An Investigation into Influence Factor of Student Programming Grade Using Association Rule *Mining. Advances in Information Sciences and Service Sciences*. 2010; 2(2): 19-27.
[16] Yuan Wang, Lan Zheng. Endocrine Hormones Association Rules Mining Based on Improved Apriori Algorithm. *Journal of Convergence Information Technology*. 2012; 7(7): 72-82.
[17] YiJie Chen. The Development of The Commodity Flow Analysis System Based On Association Rule Mining. *International Journal of Advancements in Computing Technology.* 2012; 4(13): 430-436.
[18] Changjiang Li, Xianfeng Yang. The Research of Recommendation Systems in E-Commerce Based on Association Rule Improved-Apriori Algorithms. *International Journal of Advancements in Computing Technology*. 2012; 4(21): 354-361.
[19] J Rönkkönen, S Kukkonen, KV Price. *Real-parameter optimization with differential evolution.* Proc. IEEE Congr. Evolut. Comput., Edinburgh, Scotland. 2005: 506–513.
[20] AK Qin, VL Huang, PN Suganthan. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Trans. Evol. Comput.,* 2009; 13(2): 398-417.
[21] S Das, A Abraham, UK Chakraborty, A Konar. Differential evolution using a neighborhood-based mutation operator. *IEEE Trans. Evol. Comput.,* 2009; 13(3): 526–553.
[22] J Brest, S Greiner, B Boskovic, M Mernik, V Zumer. Selfadapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Trans. Evol. Comput.,* 2006; 10(6): 646–657.