ICBAR: An Efficient Mining of Association Rules in Huge Databases

Reza Sheibani and Amir Ebrahimzadeh, *Member, IACSIT*

***Abstract*—In this paper the problem of discovering association rules among items in extremely large databases has been considered. A novel mining algorithm named Improved Cluster Based Association Rules (ICBAR) has been proposed which can explore efficiently the large itemsets. Achieving this and initializing the cluster table (where transaction records with length k are placed in kth cluster table), database will be once scanned. Simultaneously an array with appropriate size for each itemset (named itemset array (IA)) will be created. Here kth element in the array of each itemset indicates number of transaction records in kth cluster table which have that itemset. Presented method not only prunes considerable amounts of data by comparing with the partial cluster tables but also reduces the number of large candidate itemset that must be checked in each cluster through itemset arrays. Performance and efficiency of proposed method has been compared with CBAR and Apriori algorithms. Experiments illustrate that ICBAR will do better than both of them.**

***Index Terms*—Association rule, data mining, cluster table, itemset array.**

I. Introduction

Data mining is the process of discovering useful knowledge from databases. Based on the kinds of knowledge we are seeking, tasks in data mining can be categorized into summarization, classification, clustering, association and trend analysis [1].

Association rules exploring is an important data mining task and was introduced in [2]. Formally, the problem is stated as follows :

Let I=[$i_1,i_2,\dots, i_m$] be a set of literals, called items, and D be a set of transactions and each transaction T is a set of items that $T \subset I$. each transaction has unique identifier TID . We can say that a transaction T, contains A, a set of some items in I, if $A\subset T$. An association rule is an implication of the form $A\rightarrow B$, where A, $B\subset I$ and $A\cap B=\emptyset$. if C% of transactions in D that contain A also contain B the rule $A\rightarrow B$ can be obtained from D with confidence . the rule $A\rightarrow B$ has support s in D if s% of transactions in D contain AUB.

For mining association rules initially we find set of items that their support is greater than or equal to user_specified minimum support (minsup). We call them large itemsets. If there are K items in a large itemset we call it a large k_itemset.

Subsequently, we use the large itemsets to generate effective association rules. If the confidence of an association rule is greater than or equal to user_specified minimum confidence (minconf), then it is effective. The key work for finding the association rules is to generate all large itemsets.

Several algorithms for mining association have been proposed. The importance is that algorithms must be efficient. In this paper, we present a new algorithm called Improved Cluster Based Association Rule (ICBAR) for efficient association rules mining.

The rest of this paper is organized as follows: in Section 2, previous works has been reviewed. In Section 3 we propose our algorithm and give an example. Experimental studies are presented in Section 4, and finally Section 5 include conclusion.

II. Previous Work

A grawal et al. proposed the Apriori association rule algorithm [3], [4]. Their algorithms for discovering large itemsets make multiple passes over the data. In the first pass they count the support of individual items and determine which of them are large. In each subsequent pass, they start with large itemsets that are generated at the end of the previous pass and generate new potentially large itemsets, called candidate itemsets and counted the actual support for these candidate itemsets during the pass over the hole data. They determine which of the candidate itemsets are actually large and they become the seed for the next pass. This process continued until no new large itemsets are found [3]. They need to contrast with the whole data base level by level in the process of mining the association rules.

Savasere et al. proposed the partition algorithm to reduce both CPU and I/O over heads. Their algorithm executes in two phases. In the first phase, the partition algorithm logically divides the database in to a number of on overlapping partitions. The partition is considered one at a time and all large itemsets for that partition are generated. At the end of phase 1, these large itemsets are merged to generate a set of all potential large itemsets. In phase 2, the actual support for theses itemsets are generated and the large itemsets are identified. The partition sizes are chosen such that each partition can be accommodated in the main memory so that the partitions are read only once in each phase [5].

Pork et al. proposed an effective algorithm DHP (direct hashing and pruning) for the initial candidate set generation. This method efficiently controls the number of candidate 2_itemsets, pruning the size of database [6]. Like Apriori it requires as many database passes as the largest itemset.

Agrawal et al. proposed the mining sequential patterns

Manuscript received June 25, 2012; revised August 10, 2012.

R. Sheibani is with the Department of Computer, Mashhad Branch, Islamic Azad University, Mashhad, Iran (e-mail: reza.shni@gmail.com)

A. Ebrahimzadeh is with the Sama technical and vocational training college, Islamic Azad University, Mashhad branch, Mashhad, Iran (e-mail: ebrahimzadehamir@gmail.com).

algorithm that included time attributes to discover sequential patterns [7]. Cheung et al. proposed the fast distributed algorithm (FDA) to efficiently discover association rules in a distributed system [8]. Toivonen proposed a sampling algorithm which can reduce the number of database scans to a single scan but still wastes considerable time on candidate itemsets [9].

Brin et al. proposed the dynamic itemset count (DIC algorithm). DIC algorithm dynamically counts candidates of varying length as the database scan progresses, and thus is able to reduce the number of scans over Apriori [10]. Similarly, other algorithms were proposed for reducing either the CPU computation time or the disk access overhead [11]-[15].

Tsay et al. proposed cluster_based association rule (CBAR) algorithm. it creates cluster tables to aid discovery of large itemsets. The large itemsets are generated by contrasts with the partial cluster tables [11].

III. Improved Cluster Based Association Rule (ICBAR)

If an algorithm can both reduce the number of database scans, and also the number of candidate itemsets, its efficiency will be improved. Thus we present ICBAR algorithm for discovering the large itemsets. ICBAR not only requires a single scan of the transaction database -followed by contrasts with the partial cluster tables- but also reduces the time needed to generate candidate large itemsets in each cluster by using itemset arrays.

A. ICBAR algorithm

Algorithmic form of ICBAR is shown below. (For ease of presentation some statements are labeled.)

```
Large_itemsets ICBAR_ algorithm (database, minsup)
{
    1.  Create M cluster tables;
    2.  Generate $L1$ (set of large 1_itemsets);
    3.  For ($k = 2$; $L_{k-1} <> Ø$; $k$++)
        {
    4.      Create $c_k$ from $L_{k-1}$;
    5.      For each candidate large itemset $CL$
            {/*for creating its itemset array $IA_{cl}$ and sum of
            elements of its itemset array $sum_{cl}$, consider its
            constructive large itemsets are ($C_1$, $C_2$) and
            their itemset arrays are $IA_{C1}$ and $IA_{c2}$ */
    6.          For ($i = k$; $i <= m$; $i$++)
                {
    7.              $IA_{CL}[i] = \min(IA_{c1}[i], IA_{C2}[i])$;
    8.              $Sum_{Cl}$ += $IA_{CL}[i]$
                }//for $i$
            }// for each
    9.      For ($i = k$, $I <= m$; $i$++)
            {
                For each candidate large itemset $CL$
                {
                    if ($sum_{CL}$ >= minsup)
                    {
                        res = number of appearance of $CL$ in cluster_table($i$);
                        $Sum_{cl}$ -= $IA_{cl}[i]$ - res;
                        Update $IA_{cl}[i]$ ($IA_{CL}[i]$ = res).
                    }
                    If ($sum_{CL}$ < minsup) delete $CL$ from Candidate large itemsets.
                } //for each
            }// for $i$
            For each candidate large itemset like $CL$
                If ($sum_{CL}$ >minsup)
                    Add $CL$ to $L_k$;
        }// for $k$
} // ICBAR algorithm
```

It will first scan the database once, and cluster the transactions. If the length of transaction record is k, the transaction record will be stored in cluster table (k) $1 \le k \le m$, where m is the length of the largest transaction record in database (statement1).

The set of large 1_itemset, $L1$, is generated. For each large 1_itemset like $C1$, we create its itemset array (IA_{C1}) with M element.

K_th element of IA_{C1} indicates number of transaction records in cluster_table(k) that contain $C1$ (statement 2).

We generate the set of candidate k_itemset C_k similar to the candidate generation of Apriori algorithm (statement 4).

For each candidate large itemset Like CL,

first we create its itemset array (IA_{C1}) and the sum of elements of its itemset array (sum_{C1}).

Second, the number of transaction records in each cluster_table that contain this CL will be estimated by considering IAs correspond to constructive large itemsets (statements 5-8).

For each candidate large itemset like CL, when sum_{CL} is less than minsup, CL from candidate large itemsets will be deleted. If real number of appearance of CL in cluster_table (i) is less than $IA_{CL}[i]$, we update $IA_{CL}[i]$ and sum_{cl}. CL may be deleted from candidate large itemsets by reducing sum_{CL} (statements 9-15).

B. An example of ICBAR

We provide an example to further explain the application of our algorithm. There are 24 transactions in the database. An example transaction database is show in Table I.

Cluster_tables are shown in table II. Minsup is set at 30%. The large 1_Itemsets are {1}, {2}, {3}, {4} and their itemset arrays are shown in table III.

For k=2 we generate candidate 2_itemsets C2. They are {1,2}, {1,3}, {1,4}, {2,3}, {2,4}, {3,4}. Their itemset array and sum of elements of itemset array (sum_{CL}) are shown in table IV. All of them may be large because their sum_{CL} are greater than or equal to minsup. For i=2, it is necessary to compute the number of occurrences of each candidate large itemset (CL) in the cluster_table(2) and update IA_{CL} {2} and sum_{CL} (Table X). {3, 4} and {1, 4} are deleted from candidate large itemsets.

For i=3, it needs to compute the number of appearance of each candidate large itemset, CL, in the cluster_table(3) and update their itemset array and sum_{CL} (Table VI).

{2, 4} is deleted from candidate large itemsets. After execution iterations for i=4 and i=5 our arrays and sum of their elements are shown in Table VII. The large 2_itemsets {1, 2}, {1, 3}, {2, 3} are generated.

For k=3, only candidate 3_tiemset (C3) is {1, 2, 3}. Its itemset array and sum of its elements are shown in Table VIII. The minsup is specified as 20%. {1,2,3} can be large itemset. We execute statements 9-15 for 3≤*i*≤5. {1,2,3} occurs only once in cluster_table(3) and occurs three times in the cluster_table(4) and occurs once in cluster_table(5) (Table IX). Its support is greater than 20%. Thus {1,2,3} is large 3_itemset and our algorithm is terminated. Therefore the large itemsets in this example are {1,2}, {1,3}, {2,3} and {1,2,3}. Then we can transform each large itemset into an association rule.

TABLE I: AN EXAMPLE OF TRANSACTION DATABASE

TID	Items	TIS	Items	TID	Items	TIS	Items
10	1,2,3	70	5	130	1,2,3,4	190	1,3,5
20	1,2,3,4,5	80	1,2	140	2,3	200	2,3,4
30	1,3	90	4	150	1,3,4	210	1
40	2,4	100	1,3	160	3	220	1,2,3,4
50	1	110	2,3,5	170	1,2	230	1,2,4,5
60	2	120	1,2,3,5	180	1,2	240	1,3,4,5

TABLE II: FOUR CLUSTER TABLES

Cluster_table(1):	50 60 90 160 210
Cluster_table(2):	30 40 80 100 140 170 180
Cluster_table(3):	10 110 150 190 200
Cluster_table(4):	20 120 130 220 230 240
Cluster_table(5):	20

TABLE III: ITEMSET ARRAYS OF LARGE 1_ITEMSETS

	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
{1}	2	5	3	5	1
{2}	1	5	3	4	1
{3}	1	3	5	4	1
{4}	1	0	2	3	1

TABLE IV: ITEMSET ARRAYS OF CANDIDATE 2_ITEMSETS AND SUM OF THEIR ELEMENTS

	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Sum
{1,2}	5	3	4	1	13
{1,3}	3	3	4	1	11
{1,4}	1	2	4	1	8
{2,3}	3	3	4	1	11
{2,4}	1	2	4	1	8
{3,4}	1	2	4	1	8

TABLE V: ARRAYS AFTER EXECUTION OF STATEMENTS (9-15) FOR *i* =2

	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Sum
{1,2}	3	3	4	1	13
{1,3}	2	3	4	1	11
{1,4}	0	2	4	1	8
{2,3}	1	3	4	1	11
{2,4}	1	2	4	1	8
{3,4}	0	2	4	1	8

TABLE IV: ARRAYS AFTER EXECUTION OF STATEMENTS (9-15) FOR *i* =3

	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Sum
{1,2}	3	1	4	1	9
{1,3}	2	3	4	1	11
{2,3}	1	3	4	1	9
{2,4}	1	1	4	1	7

TABLE VII: ARRAYS AFTER EXECUTION OF STATEMENTS (9-15) FOR *i* =4 AND *i* =5

	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Sum
{1,2}	3	1	4	1	9
{1,3}	2	3	4	1	11
{2,3}	1	3	3	1	8

TABLE VIII: ITEMSET ARRAY OF CANDIDATE 3_ITEMSET

	Cluster 3	Cluster 4	Cluster 5	Sum
{1,2,3}	1	4	1	6

TABLE IX: ITEMSET ARRAY AFTER EXECUTION OF STATEMENTS (9-15) FOR 3≤*i*≤5

	Cluster 3	Cluster 4	Cluster 5	Sum
{1,2,3}	1	3	1	5

IV. EXPERIMENTED RESULTS

To evaluate the efficiency of the proposed method, we have implemented the ICBAR along with apripri and CBAR algorithms.

All programs were developed on a Pentium III,1.1 6ooMHZ PC with 256MB Main memory, running on windows XP operating system. All programs were developed using Delphi 7. The test database is real_life database. The efficiency of ICBAR is compared to Apriori and CBAR algorithms.

1) 10000 transaction records of experimental data are sampled randomly from the real_life database. The test database contains 1500 items, in which longest transaction records contains 19 items. The performance of ICBAR is compared with CBAR and Apriori under various user specified minimum support, such that 0.60, 0.55, 0.50, 0.45, 0.40%. The results are shown in Table X. We can show that whenever the minsup is deceases the gaps between algorithms are evident.

TABLE X.

0.60	0.55	0.50	0.45	0.40
0	500	1100	1900	4500
0	200	400	600	1000
0	150	220	310	570

2) 30000 transaction records of experimental data are randomly sampled from real_life database. The test database contains 1500 items in which the longest transaction record contains 19 items. The performance of ICBAR is compared to Apriori and CBAR algorithms under the various minimum supports are set at 0.42%, 0.40%, 0.38%, 0.36% and 0.34%. The results are shown in Table XI.We can show that whenever the minsup is decrease, again the gap between algorithms increases too.

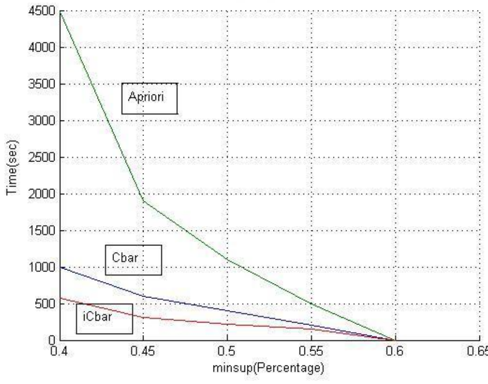
3) 25000, 35000, 45000 and 55000 transaction records of experimental data are randomly sampled from real_life database. The test database contains 1500 items, in which the longest transaction record contains 19 items. The performance of ICBAR is compared with Apriori and CBAR algorithms, where minsup is set to 0.45% and the number of transaction records is varied at levels 25000, 35000,45000, and 55000. The results are shown in Table XII.

TABLE XI.

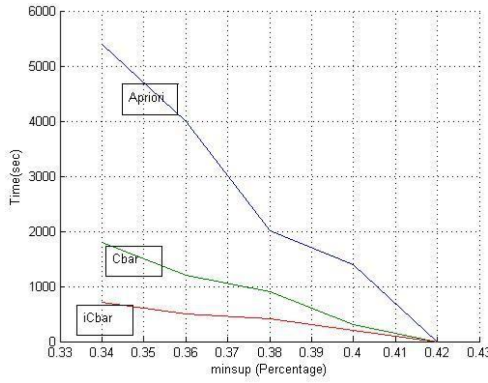
0.40%	0.42%	0.38%	0.36%	0.34%
0	1400	2010	4000	5400
0	300	900	1200	1800
0	200	415	500	710

TABLE XII

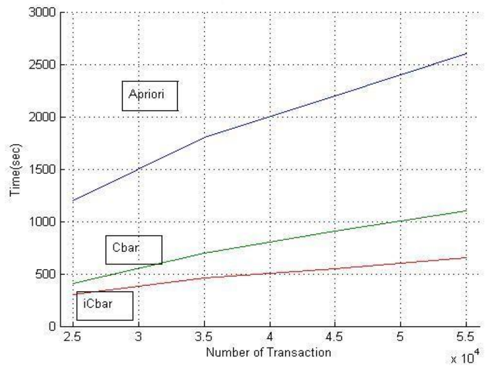
2500 0	35000	45000	55000
1200	1800	2200	2600
410	700	910	1100
300	460	550	650



Plot of Table X



Plot of Table XI



Plot of Table XII

V. Conclusion

The ICBAR algorithm creates cluster_tables and IAs to aid discovery of large itemsets. ICBAR not only requires a single scan of the transaction database, followed by contrasts with the partial cluster tables, but also reduces the time needed to generate large itemsets by pruning the candidate large itemset through IAs.

Experiments show that the ICBAR algorithm has better performance than Apriori and CBAR algorithms. Specially, when there is an increase in the number and the size of discovered patterns the performance efficiency of proposed algorithm and two others will be clearly visible.

References

[1] M. S. Chen, J. Han, and P. S. Yu, "Data mining: an overview from a database perspective," *IEEE Transaction on Knowledge and Data Engineering* vol. 8, no. 6, pp. 866-883, 1996.

[2] R. Agrawal, T. Imielinsi, and A. Swami, "Mining association rules between sets of items in very large database," *ACM SIG, MOD Conf. Management of Data*, pp. 207-216, 1993.

[3] R. Agrawal and R. Srikant, "Fast algorithm for mining association rules in large databases," *Proceedings of 1994 International Conference on VLDB*, pp. 487-499, 1994.

[4] R. Srikant and R. Agrawal, "Mining generalized association rules," *Proceedings of the 21st International Conference on VLDB*, Zurich, Switzerland, pp. 407-419, 1995.

[5] A. Savasere, E. O miecinski, and S. Navathe, "An efficient algorithm for mining association rules in large database," *Proceeding of 21st VLDB Conference Zurich*, Switzerland, pp. 432-444, 1995.

[6] J. S. Pork, M. S. Chen, and P. S. Yu, "An effective hash based algorithm for mining association rules," *ACM SIGMOD*, pp. 175-186, 1995.

[7] R. Agrawal and R. Srikant, "Mining sequential patterns," *Proceedings of the 11th International Conference on Data Engineering (ICDE)*, 1995.

[8] D. W. Cheung, J. Han, V. T. Ng, A. W. Fu, and Y. Fu, "A fast distributed algorithm for mining association rules," *Proceeding of International Conference on PDIS' 96*, Miami beach, Florida, USA, 1996.

[9] H. Toivonen, "Sampling large database for association rules," *Proceeding of 22nd VLDB Conference Mumbai*, India, pp.134-145, 1996.

[10] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur, "Dynamic itemset counting and implication rules for market basket data," *Proceedings of the ACM SIGMOD International Conference on Management Of Data*, pp. 255-264, 1997.

[11] Y. Tsay and J. Chiang, "CBAR an efficient method for mining association rules," *Knowledge Based Systems* vol. 18, PP. 99-105, 2005.

[12] F. Bodon, "A fast Aprioro Implementation," *Proc 1st IEEE ICDM Workshop on Frequent Itemset Mining Implementations(FIMI2003, Melbourne, FL)*.

[13] M. H. Margahny and A. A. Mitwaly, "Fast algorithm for mining association rules," *AIML 05 Conference*, CICC, Cairo, Egypt, 2005.

[14] J. Han, J. Peiand and Y. Yin, "Mining frequent patterns without candidate generation," *Proc. ACM SIGMOD* 2000.

[15] K. Palshikar, S. Kale, and M. Apte, "Association rules mining using heavy itemsets," *Advances in Data Management*, pp. 148-155, 2005.

Reza Sheibani is a Faculty Member of Computer Engineering Department of Islamic Azad University, Mashhad Branch, Mashhad, Iran. He is born on 20 April in Mashhad. He gained his B.S. Degree in Computer Software engineering, Ferdowsi University of Mashhad, 1998. And his M.S. Degree in Computer Software engineering at Islamic Azad University South Tehran Branch, 2002.His major field of study is data mining.

Amir Ebrahimzadeh is a Faculty member of Sama technical and vocational training college, Islamic Azad University, Mashhad branch, Mashhad, Iran. He is born on 12 September in Mashhad. He gained his B.S. Degree in Computer Software engineering, Azad University of Mashhad, 2003. And his M.S. Degree in Computer Software engineering at Islamic Azad University Mashhad Branch, 2005.His major field of study is data mining.