Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 4, April 2014, pg.346 – 358

SURVEY ARTICLE

A SURVEY ON IMPROVING PERFORMANCE OF APRIORI ALGORITHM

Er.Rajvir Kaur[1], Er.Nishi Madaan[2]
[1]Department of Computer Science and Engineering, DAV University, Jalandhar, Punjab, India
[2]Department of Computer Science and Engineering, DAV University, Jalandhar, Punjab, India
[1] rajvir.parmar33@gmail.com; [2] nishi.02.bti@gmail.com

Abstract — Association rule mining concept is used to show relation between items in a set of items. Apriori algorithm is one the mostly used algorithm is association rule mining. It extracts the frequent and useful patterns from the large databases. It is easy to understand and implement yet it has some drawbacks. Many algorithms are being given to improve its performance. This paper does a survey on different improved approaches of Apriori algorithm which improves its performance by removing its drawbacks and reduces its execution time by parallel implementation of it.

Keywords – data mining; association rules; Apriori algorithm; Mapreduce; Hadoop Cluster; CUDA

I. INTRODUCTION

Data mining is gaining more importance nowadays as abundant of data is generated with the rapid development of networks and information technology. Data is collected in large repositories which are seldom visited by decision makers because data collected is not information rich data. Data mining is defined as a process of extracting useful information from large repositories which can be efficiently and effectively used.[1][6] It's an interesting as well as important topic for researchers for discovering hidden patterns from large stores of data. One of the approaches of data mining is Association Rule Mining i.e. defining association rules.

A. Association rules

Association rule mining is process of extracting interesting relations, patterns between the unrelated data in the transactional databases or other data repositories. It is mostly used in "market basket analysis". It deals with the problem of discovering the features that "happen together" in huge transaction databases. Association rules have two main characteristics support and confidence.[1][2]

Let LI={$li_1,li_2,li_3,\ldots\ldots.li_m$} be the set of items, m binary attributes. D is the set of transactional database. D={$t_1,t_2,t_3,\ldots\ldots\ldots t_m$}. An association rule is defined X=>Y where X, Y are subsets of LI and X ∩ Y is a null set. X is called antecedent and Y is called consequent. This rule is read as if a transaction contain X then it also contain Y.

Support for an association rule is defined as a ratio of number of transaction that holds association rule to total number of transaction.

$$Sup(X)= (N(X \cup Y)/N(D))*100\%$$

Confidence of a rule is defined as ratio of number of transactions which hold rule to number of transaction containing antecent.

$$Con(X=>Y)=(N(X \cup Y)/N(X))*100\%$$

Confidence is the condition probability, if transaction contain X then it will also contain Y. Let us take an example of association rule
{Bread, butter}=> {milk} support 20% confidence 60%.

It states that when bread and butter are brought together by customer he would likely to purchase milk also. Like this we generate the different rules based on data collected in transaction database.

An association rule is said to be strong if its support and confidence is exceed minimum support and confidence. The minimum support and confidence is set by users or domain experts which are the constraints of the rules. If minimum support threshold is set high then important and meaningful rules may not be discovered; if minimum support is set low large number of rules will be produced which may decrease mining efficiency[2].

B. Apriori Algorithm

Apriori Algorithm is one of the prominent algorithms used for mining of frequent item sets in transactional databases. It was introduced by R.Agarwal and R Srikant for mining frequent item sets [3]. Its uses bottom-up iterative approach. As this algorithm needs forehand information about frequent item set properties, it's called Apriori Algorithm. It uses breath-first search where k-1 item set are used to find k item sets. This algorithm makes use of Apriori Property which states that "all non empty subsets of frequent item set must also be frequent". There are two main steps in Apriori. i)The join step. ii) The prune step.[1][5][6]

The join step: The item sets of level L_k are joined to generate the candidate item sets of length k+1for level L_{k+1}. The set of these candidate item sets is denoted by C_{k+1}.

The prune step: Scan the database for calculating the count for each candidate in C_k and delete those candidate item set whose support count is less than minimum threshold support as they are not frequent. The complete algorithm is discussed in next section.

C. APRIORI ALGORITHM:

Input:
D, a database of transactions;
min sup, the minimum support count threshold.[6]

```
Method:
(1) $L1$ = find frequent 1-itemsets(D);
(2) for ($k$ = 2;$L_{k-1}$ 6= f;$k$++) f (3) $C_k$ = apriori gen($L_{k-1}$);
(4) for each transaction $t$ 2 $D$ f // scan $D$ for counts
(5) $C_t$ = subset($C_k$, $t$); // get the subsets of $t$ that are candidates
(6) for each candidate $c \in C_t$
(7) c.count++;
(8) }
 (9) $L_k$ = {c ϵ $C_k$|c.count>= $min sup$}
 (10) {
```

```
(11) return $L = U_k L_k$;

Procedure apriori gen($L_{k-1}$:frequent ($k$-$1$)-itemsets)
(1) for each itemset $l_1 \in L_{k-1}$
(2) for each itemset $l_2 \in L_{k-1}$
(3) if $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge ::: \wedge (l_1[k-2] =$
 $l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$ then { $c = l_1 \bowtie l_2$; // join step:

generate candidates
(5) if has infrequent subset($c$, $L_{k-1}$) then
(6) delete $c$; // prune step: remove unfruitful candidate
(7) else add $c$ to $C_k$;
(8) {
(9) return $C_k$;

Procedure has infrequent subset($c$: candidate $k$-itemset;
$L_{k-1}$: frequent ($k$-1)-itemsets); // use prior knowledge
    (1) for each ($k$-1)-subset $s$ of $c${
(2) if $s \notin L_{k-1}$ then
(3) return TRUE ;}
(4) return FALSE;

OUTPUT: L, frequent item set in D.
```

Let us illustrate the working of Apriori algorithm by taking a transactional database of 10 records; each of which has transaction id and items purchased by customer in that particular transaction. Minimum support count is 3.

The process of finding the frequent item set is described in following steps:

1) In level L_1 count the occurrences of each item and eliminate the items whose count is less than 3.

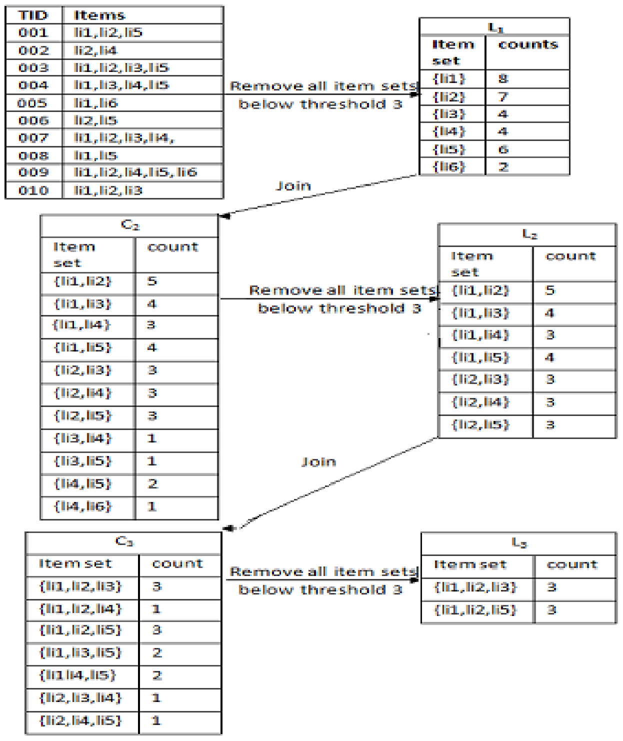
2) Generate the candidate set C_2 by joining item sets in L_1. Calculate each candidate item set count and remove the sets whose support count is less than 3 to generate level L_2.

3) Join the items set of L_2 to form candidate set C_3; find all the occurrences of item set in candidate set C_3. Prune the item sets whose count is less then minimum support count, it produce level $_{L3}$.

4) Joint the item sets of L_3 to produce item sets each containing four items. It is found that all the occurrence of C_4 is less then threshold minimum support i.e. 3.

We stop the iterative process here.

In this shopping market basket example, we found that not every item combination exist in transactional database. The Apriori algorithm removes such a combination if it does not exist or its count is below minimum threshold. Working of Apriori algorithm is illustrated in figure 1.


Remove all item sets below threshold 3
Join
Remove all item sets below threshold 3
Join
Remove all item sets below threshold 3

TID	Items
001	Ii1,Ii2,Ii5
002	Ii2,Ii4
003	Ii1,Ii2,Ii3,Ii5
004	Ii1,Ii3,Ii4,Ii5
005	Ii1,Ii6
006	Ii2,Ii5
007	Ii1,Ii2,Ii3,Ii4,
008	Ii1,Ii5
009	Ii1,Ii2,Ii4,Ii5, Ii6
010	Ii1,Ii2,Ii3

L_1

Item set	counts
{Ii1}	8
{Ii2}	7
{Ii3}	4
{Ii4}	4
{Ii5}	6
{Ii6}	2

C_2

Item set	count
{Ii1,Ii2}	5
{Ii1,Ii3}	4
{Ii1,Ii4}	3
{Ii1,Ii5}	4
{Ii2,Ii3}	3
{Ii2,Ii4}	3
{Ii2,Ii5}	3
{Ii3,Ii4}	1
{Ii3,Ii5}	1
{Ii4,Ii5}	2
{Ii4,Ii6}	1

L_2

Item set	count
{Ii1,Ii2}	5
{Ii1,Ii3}	4
{Ii1,Ii4}	3
{Ii1,Ii5}	4
{Ii2,Ii3}	3
{Ii2,Ii4}	3
{Ii2,Ii5}	3

C_3

Item set	count
{Ii1,Ii2,Ii3}	3
{Ii1,Ii2,Ii4}	1
{Ii1,Ii2,Ii5}	3
{Ii1,Ii3,Ii5}	2
{Ii1Ii4,Ii5}	2
{Ii2,Ii3,Ii4}	1
{Ii2,Ii4,Ii5}	1

L_3

Item set	count
{Ii1,Ii2,Ii3}	3
{Ii1,Ii2,Ii5}	3

Figure 1. Generation process of candidate frequent item sets[1][5]

Apriori algorithm is simple to implement and understand but there are some shortcomings which are discussed below: [5][19]
1) Scanning database: Transaction database need to be scanned repeatedly in order to find the frequent itemset. If there are n items in the database, it require the database to be scanned atleast n times.
2) Setting of minimum support: in real time world, some patterns are frequent and some are rare. If minimum support is set too low then large set of candidate item set is generated which decreases the efficiency and availability of the rule. If minimum support is set too high then meaningful rules may not be discovered.
3) the fitness landscape of the algorithm is narrow: the algorithm consider only a single Boolean association rule mining but in real world problems, there might be multi dimensional, multi-volume and multi layer association rules.

II. Literature Review

Goswami D.N., Chaturvedi Anshu., Raghuvanshi C.S. [2], proposed an algorithm for frequent pattern mining based on Apriori algorithm. He has given three approaches: record filter, intersection and proposed the third one which is the combination of record filter and intersection. In record filter we count the support of candidate set only in those transactions whose length is greater than length of candidate set, because the candidate set of length k, cannot exist in transaction of length less than k. In intersection approach we uses vertical data layout. In this we calculate the support we count the common transaction that contains in each element's of candidate set by using intersect query of SQL. The proposed approach we have features of both record filter and intersection approach, we calculate the support by counting common transaction that contains in each element's of candidate set with the help of intersect query of SQL and only those transactions are considered that contain at least k items.

Sheila A. Abaya in her paper [3], has given an improved apriori algorithm based on set theory. It takes for the size which is the number of items per transaction and set size frequency which is the count of transactions having at least "set size" items.

Maria S. Perez in her paper [10], presented an optimization of apriori algorithm making use of MAPES(Multi –Agent Parallel File System) a high performance parallel file system, which distributes the data and provides parallel access to files data, what reduces the bottleneck that constitutes the access to servers.

Hassan M. Najadat in his paper [10], has proposed an approach to reduce the spent in searching frequent item sets in transactions database.

Jaishree Singh paper [16],aimed to improve the performance of apriori algorithm by reducing the size of database. He introduced a new attribute Size of transaction (SOT) that takes the value of number of items in individual transaction in database. he deletion process of the transaction takes place accordance to the value of k. If the value of K matches with the value of SOT then remove only those transactions from the database.

Nilesh.S.Korde and prof. Shailendra.W.Shende [4], presented parallel implementation of apriori algorithm on quad core processer. Multi core processor shares the load among cores with single thread or multiple threads which decrease processing time and increase performance.

Ning Li, Li Zeng, Qing He, Zhongzhi Shi [8], altogether, implemented a parallel Apriori algorithm based on MapReduce. Mapreduce is patented software frameworks introduces by Google in 2004.it is an associated implementation for processing and generating large data sets in a massively parallel manner. It specifies the computation in terms of a map and reduce function. Map takes an input pair and produces a set of intermediate key/value pairs and map function is used to take a single key/value pair and gives a list of new key/value pairs as an output.

A. Ezhilvathani, Dr. K. Raja [9] implemented parallel Apriori Algorithm using Apache Hadoop software framework which improves the performance. Hadoop is the software framework for writing applications that rapidly process vast amounts of data in parallel on large clusters of compute nodes. Its working is based on MapReduce model. MapReduce is a distributed programming model intended for large cluster of systems that can work in parallel on a large dataset.

Mamta Dhanda [4] presented a paper proposing an approach based o n weight factor and utility for mining of high utility patterns. She calculated profit ratio by using Q-factor

$\text{Q-Factor}=P/(\sum P_i)$

Where P is profit of item=1 to n(no of items)

J. Hossen in his paper [7], proposed algorithm for rule formation. In Engine(MAFIE). In this the clusters are identified in fuzzy c-means clustering method, which is unsupervised clustering method whose aim is to establish a fuzzy partition of a set of cluster prototypes towards the local minimum of their objective function. An objective function J_m defined by measures the fitting between the clusters and their cluster prototypes.

$J_m(M,v)=\sum\sum(u_{ik})^m(d_{ik})^2$

Where $u_{ik}\in[0\ 1]$ is a membership degree of the kth pattern vector to the ith cluster represented by its cluster prototype v_i, and v is a cluster prototype of u_i. The distance measure d_{ik} used in the Fuzzy C-means clustering is the Euclidean norm $d_{ik}=||x_k - v_i||$

Abhaya Kumar Sahoo [12], improved the performance of Apriori algorithm by implementing it on CUDA platform. CUDA is a parallel computing platform and programming model invented by NVIDIA. It increases computation performance by harnessing the power of graphics processing unit(GPU). It is collection of threads running in parallel.

N. Badal [13], presented the paper implementing Apriori algorithm in MATLAB. The efficiency and performance of Apriori algorithm increases. The execution of Apriori algorithm is 80% less than its execution in C.

Yan Zhang, Jing Chen in their paper [17] proposed an AVI algorithm. Transaction database is vertical, itemset union and indentification intersection is used.For itemset X, t (X) = {tid | tid is transaction id, t belongs to D and t supports X}; For the identification set Y, i(Y) = y belongs to Y itemset(y), itemset (y) that y corresponding to the transaction itemset[4]. For example:

$t(\{A,B,D\}) = \{2,6\}$

$i(\{2,6\}) =\{A,B,D\}\cap\{A,B,C,D,E\}=\{A,B,D\}$

Lingjuan Li Min Zhang paper [18], presented a strategy for mining association rules on cloud computing. Cloud computing provides the efficient storage and analysing large data. It distributes the computational tasks to resource pool, which consist of large number of computers. We partition the data for parallel computing under cloud computing environment. To improve algorithm we combine it with Mapreduce programming model, Hadoop.

III. Critical Review on the various improvements of Apriori Algorithm

Several improved algorithms have been proposed to remove the shortcomings of Apriori algorithm and enhance its performance by reducing its execution time or reducing the number of database scanning passes. Some of the algorithms are discussed below in table I:

Sr No.	Name to the technique	Description	Advantages	Disadvantages
1	Apriori algorithm[1][2]	It was proposed by R Agarwal and R Srikant in 1994 for discovering frequent item set for Boolean association rule.	It gives a frequent item set in transactional database as an output. It's an efficient algorithm for fining frequent items	1) Generate large number of candidate item set. 2) Repeatedly scanning the transaction databases.
2	Record filter approach[3]	Only those transactions are considered to calculate the support count of candidate set whose length is greater than the length of candidate item set. If length of candidate item set is k, only transaction whose length is at least k is considered as k length candidate set cannot exist in the transaction record whose length is is then k.	1. This approach takes less time as compared to classical apriori algorithm this it improves the efficiency of apriori algorithm and memory management. 2. It removes the complexity of process	Memory optimization is done but needs further optimization.
3	Intersection Approach [3]	The intersection Algorithm is made to improve the efficiency, memory management and complexity of Apriori algorithm. It is more appropriate for vertical data layout. We calculate the support by counting the number of transactions that contain in each element's of candidate set by using intersect query of SQL.	This requires very less time then classical algorithm and record filter approach.	Data must be in vertical layout.

4	Combination of both Record filter and intersection approaches[3]	It was given by Goswami. We use set theory concept of intersection with the record filter approach. To calculate the support, we count the common transaction that contains in each element's of the candidate set, with the help of the intersect query of SQL. In this approach we have applied constraints that we will consider only those transactions that contain at least k items, not less than k in process of support counting for candidate set of length.	Its execution time is far less than intersection and record filter. Execution time relation of all three approaches is given as: $T_{combine}<T_{intersection}<T_{record}$	Memory optimization is done but still it needs more optimization.
5	AVI algorithm [17]	Transaction database is vertical, itemset union and indentification intersection is used. For itemset X, t (X) = {tid | tid is transaction id, t belongs to D and t supports X}; For the identification set Y, i(Y) = y belongs to Y itemset(y), itemset (y) that y corresponding to the transaction itemset[4]. For example: t({ A ,B ,D}) = { 2 , 6} i({2,6}) ={A,B,D}∩{A,B,C,D,E}={A,B,D}	Moderate memory overheads and reduces I/O frequency.	Databases must be in vertical form.
6	Data partition method[7]	Large capacity databases will be logically divided into several disjoint blocks which are used to generate locally frequent item sets, and then these local frequent item sets to get the final global frequent items sets through testing their support.	Improves the performance of apriori algorithm as mining of frequent patterns from partitioned data sets can be executed parallel.	May some items gets missed due to partitioning.
7	Hashing method.[5]	When scanning each transaction in databases to hash them to different barrels of the different barrels and increase the corresponding table count. In the hash table the 2-itemset with corresponding hash bucket count less than the values of support must not be a frequent, should be removed from candidate item set.	It can greatly compress the k-item set to be examined.	Need extra time for hashing the transaction in the database to different barrels of the different barrels.

8	Sampling method[6]	Select the random sample S for given database D, and then search frequent item sets in the S rather in D.	As we have to scan only sample S instead of whole database D, it saves time.	This approach sacrifices some accuracy. It may lose some global frequent items sets.
9	Based on set theory.[7]	It considers the set size which is the number of transaction and set size frequency which is the count of transaction that have at least "set size" items. Remove the items with frequency less than the minimum support value. Determine initial set size to get the highest set size whose frequency is greater than minimum support. It is seen that as number of items per transaction decreases the favourable result will be from original algorithm.	This algorithm requires less database access as compared with the original one. Its execution is faster than classical algorithm.	Ideal starting size of combination size for pruning candidate keys is not given. When the number of items per transaction decreases its performance decreases as a lot of execution time due to pruning starts with the k(n)-1 where n is the maximum set size with set size frequency ≥minimum support.
10	Improved apriori algorithm proposed by Hassan M. Najadat	We firstly scan all transactions to produce L_1 which contains the items, their support count and Transaction ID where the items are found. And then we use L_1 to generate L2, L3 ... L_k. When we want to generate C_2, we make a self join L_1 * L_1 to construct 2-itemset C (x, y), where x and y are the items of C_2. Before scanning all transactions to count the support count of each candidate, use L_1 to get the transaction IDs of the minimum support count between x and y, and thus scan for C_2 only in these specific transactions and repeat these steps until no new frequent item sets are identified.	It reduces the time required in transaction scanning for generating candidate itemset by reducing the number of transaction to be scanned.	Need to store the intermediate result as all the further processing depends on level $L_{1.}$
11	Using parallel I/O and hints[10]	It makes use of MAPES (Multi-Agent Parallel File System), a high performance parallel file	Its improves Apriori algorithm global time by means of both parallel	Needs to maintain hints.

		system, which distributes the data and provides the parallel access to files data. The main configuration parameters first one is i/o caching and prefetching and second is hints, the previous task can be made in a more efficient way using hints on future access patterns	access and using hints for removing data blocks.	
12	Parallel implementation using multi core processor[11]	Parallelism is used to reduce time and increase performance. Multicore processor is used for parallelization. Multicore processor share load among cores with single thread or multiple threads, which reduces time for processing and increase performance. The performance of multicore processor depends on the software used.	In load balancing the work is equally shared among the processors which minimizes the execution time and improve performance.	It is observed that the real time decreases with parallelization than serial execution and it is increases with the increase of number of threads. There is only a slight reduction in the user time with parallelization compared to sequential execution but the benefit of parallelization is not observed in user time, system time as in the case of real time and the system time is slightly increased in parallel execution.
13	Parallel implementation on Hadoop Cluster[11]	Apache Hadoop software framework is used to build the cluster. It processes vast amount of data in parallel on large cluster of computer nodes. It is open source java-based programming framework that supports the processing of large data sets in a distributed computing environment. It works on MapReduce model. In this database is divided into n subset	It drastically improves the performance of apriori algorithm. It can be easily implemented and parallelized. For finding the frequent item set it traverse the database only once.	The performance degrades with Sector file system due to I/O overhead as there is large amount of data transfer between distributed and local file system.

		and distributed to m nodes. Data in transformed in format <key, value> pair. Execute Map function to generate candidate itemset. Combine function is performed which gives output <itemset, supportcount>. Finally execute reduce task at reduce function.		
14	Apriori algorithm on cloud computing environment[18]	Cloud computing provides the efficient storage and analysing large data. It distributes the computational tasks to resource pool, which consist of large number of computers. We partition the data for parallel computing under cloud computing environment. To improve algorithm we combine it with Mapreduce programming model, Hadoop.	Under the cloud computing environment, the improved algorithm can effectively mine the frequent itemsets from mass data, and the dataset partition method and distribution method can improve the efficiency of the improved algorithm in the heterogeneous cluster environment.	As Hadoop software is used . The performance degrades with Sector file system due to I/O overhead as there is large amount of data transfer between distributed and local file system.
15	Implementation of Apriori in MATLAB[13]	Apriori Algorithm is implemented by using attribute affinity matrix. This matrix is generated from attribute usage matrix, matrix contains usage of attributes in important transactions; its each row contain single transaction. Attribute affinity can be find using the formula: $Aff(A_i, A_j)=\sum\sum ref_1(q_k)acc_1(q_k)$ A text file is created which contains the transactions in matrix form; on this file the Apriori algorithm is applied.	Implementation of Apriori in MATLAB reduces execution time and scans the database only once.	MATLAB works on only numeric data. So the categorical data needs to be converted in advance which consume time if data is large.
16	A method proposed by Mamta Dhanda[4]	In this method we first computed profit ratio form all items based on Profit which is calculated by Q-factor. Q-factor=$(P)/(\sum P_i)$ Where i=1 to n and n= number of items P is profit of the item. Perform the joining step followed by pruning step. Calculate the profit and weighing factor (PW-factor)for each frequent patterns selected on the basis of confidence.	This method aims to mine the association patterns that help in improvement of business utility. It finds all the patterns which are statistically and semantically important for business utility.	With the increase in data and item, it leads to increase in computation time.

		PW=∑frequency*Q-factor Sort the frequent patterns whose PW-factor is greater then predefined PW-factor.		
17	Implementation of Apriori algorithm using Mapreduce[8]	Mapreduce is the patented software framework given by Goggle in 2004. It is a programmable model and associated implementation for processing and generating large data sets parallel manner. Its computational is specified in terms map and reduce function. Map takes an input and gives output in form of key/value pairs and mapreduce function take one key/value pair and give a list of new key/value pairs as output. Map:: $(key_1,value_1)$=>list$(key_2,value_2)$ All map operations are independent of each other and fully parallelized.	For the datasets either have many short transactions with less frequent items or large database with larger transaction having many frequent items, proposed method has good performance.	It is performance is good when large data intensive computation is required and ceases as data decreases.
18	Using ACO, Ant Colony Optimization[22]	Apriori algorithm performance is improved using ACO. ACO was introduced by Dorigo. ACO algorithm is inspired from the behaviour of ant colonies [20]. It uses two rules; 1) local pheromone update rule which is applied in building solution. 2) Global pheromone update rule which is applied in ant conduction.ACO is used for specific problem of minimizing the number of association rules.	Efficiency of Apriori algorithm is increased using Ant colony optimization algorithm.	Information needs to be represented in form of decision table to represent the dependency of minimum set attributes and class numbers.
19	Implementation of Apriori using CUDA[12]	CUDA, Compute Unified Device Architecture is a parallel computing platform and programming model introduced by NVIDIA [25]. It provides dramatic increase in computing performance by harnessing the power of GPU (graphics processing unit).GPU and CUDA all together provides tremendous memory bandwidth and computing power.	It dramatically increases the computation power. Hence the execution time of Apriori algorithm reduces.[23] Its superior floating point computation capability and low cost enables it use in medium sized business and individuals.	

		CUDA model is collection of thread running in parallel. A whole task is given to CPU and GPU; the number of blocks and threads per block is decided before kernel start as it depends on the data size. In CUDA the proper combination of threads and blocks gives us much better result.[12]		

Table I: Comparison of different Improved Apriori algorithms

IV. CONCULSION

After doing the survey on various improvement methods of Apriori algorithm, it can be concluded all the algorithms work to improve the performance of the Apriori algorithm by reducing the candidate set, database or parallel execution of Apriori algorithm. Implementation of the Apriori algorithm in CUDA dramatically increases the performance of it by reducing the execution time of Apriori algorithm. We will get the better performance by upgrading our parallel algorithm based on release of CUDA graphic cards.

REFERENCES

[1] R. Agrawal and R. Srikant. *Fast algorithms for mining association rules.* IBM Research Report RJ9839, IBM Almaden Research Center, San Jose, California, June 1994.
[2] Goswami D.N., Chaturvedi Anshu.,Raghuvanshi C.S.,*" An Algorithm for Frequent Pattern Mining Based On Apriori"*, In: Goswami D.N. et. al. / (IJCSE) International Journal on Computer Science and Engineering ,,Vol. 02, No. 04, 2010, 942-947, ISSN : 0975-3397
[3] Sheila A. Abaya, *"Association Rule Mining based on Apriori Algorithm in Minimizing Candidate Generation"*,In:International Journal of Scientific & Engineering Research Volume 3, Issue 7, July-2012
[4] Mamta Dhanda*," An Approach To Extract Efficient Frequent Patterns From Transactional Database"*,In: International Journal of Engineering Science and Technology (IJEST), Vol.3 No.7 July 2011, ISSN:0975-546.
[5] Yanxi Liu*," study on Application of Apriori Algorithm in Data Mining"* in 2010 Second International Conference on Computer Modeling and Simulation
[6] J. Han, M. Kamber, ―Data Mining: Concepts and Techniques,‖ Morgan Kauffman, San Francisco, 2000
[7] J. Hossen, A. Rahman, K. Samsudin, F. Rokhani, S. Sayeed, and R. Hasan, ― *A Novel Modified Adaptive Fuzzy Inference Engine and Its Application to Pattern Classification*, World Academy of Science, Engineering and Technology 80, 2011
[8] Ning Li, Li Zeng, Qing He, Zhongzhi Shi, *"Parallel Implementation of Apriori Algorithm Based on Mapreduce"*, International journal of Networked and Distributed Computing, vol.1, No. 2,pp.89-96, April 2013.
[9] A. Ezhilvathani, Dr. K. Raja, *"Implementation of Parallel Apriori Algorithm on Hadoop Cluster",* IJCSMC,Vol. 2, Issue. 4, pp. 513-516, April 2013.
[10] Maria S. Perez, Ramon A. Pons, "*An Optimization of Apriori Algorithm through the Usage of Parallel I/O and Hints".*
[11] Nilesh. S. Korde, Prof. Shailendra. W. Shende, "Parallel Implementation of Apriori Algorithm", ICAET-2014, pp. 01-04, 2014.
[12] Abhaya Kumar Sahoo,Amardeep Das, Mayank Tiwary, *" Parallel Optimized Algorithm for Apriori Association Rule Mining on Grapgics Processing Unit with Compute Unified Device Architecture(CUDA)"*, IJARCSSE, volume3, issue 10, pp 1214-1219, October 2013.

[13] N. Baddal, S. Bagga, *"Implementation of Apriori Algorithm in MATLAB using Attribute Affinity Matrix"* IJARCSSE, vol. 4, issue 1, jan 2013.
[14] A. Chatterjee and A. Rakshit, *"Influential Rule Search Scheme (IRSS) –A New Fuzzy Pattern Classifier,"* IEEE Transaction on Knowledge and Data Engineering, vol. 16, no. 8, pp. 881-893Aug. 2004.
[15] ChangSu L., Anthony. Z., Tomas B., *"An Adaptive T-Stype Rough-Fuzzy Inference System (ARFIS) for Pattern Classification"*, Fuzzy Information Society, IEEE Explorer, pp. 117-122, 2007.
[16] Jaishree Singh*, Hari Ram**, Dr. J.S. Sodhi,*" improving th efficiency of Apriori algorithm by transaction reduction"*, International Journal of Scientific and Research Publications, Volume 3, Issue 1, January 2013
[17**]** Yan Zhang, Jing Chen, *"AVI:Based on the vertical and intersection operation of the improved Apriori algorithm"*, IEEE 2010.
[18]. Lingjuan Li, Min Zhang, *"The Strategy of Mining Association Rule Based on Cloud Computing",* 2011 International Conference on Business Computing and Global Informatization,IEEE.
[19]. Zhou L, Lai KK, Yen J (2009) Credit scoring models with AUC maximization based on the weighted SVM. Int J InfTechnolDecis Mak 8(4):677–696
[20]. Anshuman Singh Sadh, Nitim Shukla*," Apriori and Ant Colony Optimization of Association Rules"*, International Journal of Advanced Computer, Volume-3 Number-2 Issue-10 June-2013.
[21]. Zaki MJ (1999) Parallel and distributed association mining: a survey. IEEE Concurr7(4):4–25, *Special issue on Parallel Mechanisms for Data Mining*
[22] Huang Qiu-yong, Tang Ai-long and Sun Zi-guang, *"Optimization Algorithm of Association Rule Mining Based on Reducing the Time of Generating Candidate Itemset",* IEEE 2011.
[23]. Gaber MM, Yu PS (2006) *Detection and classification of changes in evolving data streams*. Int J InfTechnolDecis Mak 5(4):659–670
[24]. Liu Y, Pisharath J, Liao WK, Memik G, Choudhary A, Dubey P (2004*) Performance evaluation and characterization of scalable data mining algorithms*. In: 16th IASTED international conference on parallel and distributed computing and systems (PDCS). MIT, Cambridge, pp 620–625
[25]. NVIDIA (2008) CUDA programming guide 2.1. http://www.nvidia.com/object/cuda_develop.html