AN ASSOCIATION RULE MINING ALGORITHM IN DISTRIBUTED DATA MINING SYSTEM

Y.Venkata Raghavarao[1], Dr.L.S.S Reddy[2], Dr.A.Govardhan[3]

[1]Research Scholar, JNT University, Hyderabad, A.P, India
newbirth.child@gmail.com

[2] Director, LBRCE, Vijayawada, A.P, India
director@lbrce.ac.in

[3] Professor in School of IT, JNT University, Hyderabad, A.P,India
govardhan_cse@yahoo.co.in

Abstract*— *Many existing data mining (DM) tasks can be proficient effectively only in a distributed condition. The ground of distributed data mining (DDM) has therefore gained growing weightage in the preceding decades. The Apriori algorithm (AA) has appeared as one of the greatest Association Rule mining (ARM) algorithms. Ii also provides the foundation algorithm in majority of parallel algorithms (PAs). The size and elevated dimensionality of datasets characteristically existing as a key to difficulty of AR finding, makes it perfect difficulty for solving on numerous processors in parallel. The main causes are the computer memory and central processing unit pace constraints looked by single workstations. This paper is based on an Optimized Distributed AR mining algorithm for biologically distributed information is used in similar and distributed surroundings so that it decreases communication costs.

***Keywords*— Association rules (ARs), Apriori algorithm (AA), distributed data mining (DDM), XML data, Parallel**

I. INTRODUCTION

ARM has turn out to be one of the hub DM tasks and has attracted marvelous interest among DM investigators. ARM is an unsupervised DM method which works on variable length data, and produces understandable results. There are two foremost approaches for utilizing numerous workstations that have appeared in distributed computer memory in which each CPU has a confidential memory; and public memory in which all CPUs access universal memory **[3, 4]**. Collective memory planning has many gorgeous assets. Each CPU has straight and identical right to use memory in the computer system. Equivalent applications are easy to execute on such a distributed system. In allocated memory design each CPU has its own restricted memory that can simply right to use directly by that CPU **[10]**. For a CPU to have contact with facts in the restricted memory of another CPU a replica of the preferred data ingredient must be sent from one CPU to the other throughout message passing. XML information is used with the optimized distributed association rule mining (ODAM) algorithm. A similar application could be divided into numeral of jobs and implemented in parallel on different CPUs in the system **[9]**. Though the performance of a similar function on a distributed system is mainly dependent on the distribution of the jobs contains the application onto the obtainable CPUs in the system.

Modern associations are biologically dispersed. Classically, each location locally stores its ever growing amount of every day data. Using centralized DM to find out useful patterns in such institutions' data isn't always practicable because integration of data sets from dissimilar locations into a centralized location earns enormous communication system costs. Information from these institutions' is not only spread to a variety of sites but also vertically incoherent, making it complex if not unfeasible to merge them in an essential site. Distributed DM therefore emerged as vigorous subarea of DM investigation. In this paper an ODAM Algorithm is used for executing the mining procedure.

II. ASSOCIATION RULE MINING ALGORITHMS

An AR is a rule which implies certain association relationships among a set of objects in a database. Given a set of transactions, where each transaction is a set of items, an AR is an expression of the form X Y, where X and Y are sets of items. The intuitive meaning of such a rule is that transactions of the database which contain X tend to contain Y **[1].**

A) Apriori Algorithm

An AR mining algorithm, Apriori has been developed for rule mining in large transaction databases by IBM's Quest project team. An *itemset* is a non-empty set of items.

They have decomposed the difficulty of mining ARs into two parts

- Find all combinations of items that have transaction support above minimum support. Call those combinations frequent item sets.
- Use the frequent item sets to generate the desired rules. The general idea is that if, say, ABCD and AB are frequent item sets, then we can determine if the rule EF GH holds by computing the ratio r = support(EFGH)/support(EF). The rule holds only if r >= minimum confidence. Note that the rule will have minimum support because EFGH is frequent. The algorithm is highly scalable **[7].** The AA used in Quest for finding all frequent item sets is given below.

B) Pseudo code

Apriori: Finds frequent item sets using an iterative level-wise approach based on candidate generation

1. In the first iteration of the algorithm, each item is a member of the set of candidate 1-item sets, C1. The algorithm simply scans all of the transactions in order to count the number of occurrences of each item.
2. Suppose that the minimum transaction support count required is two. The set of frequent 1-item sets, L1, can then be determined.
3. To discover the set of frequent 2-itemsets, L1, the algorithm uses L1*L2 to generate a candidate set of 2-item sets.
4. Next, the transactions in D are scanned and the support count of each candidate item set in C2 is accumulated.
5. The set of frequent 2-itemsets, L2, is then determined, consisting of those candidate 2-itemsets in C2 having minimum support.
6. The generation of the set of candidate 3-itemsets, C3=L2*L2.
7. The transaction in D are scanned in order to determine L3, consisting of those candidate 3-itemsets in C3 having minimum support.
8. The algorithm uses L3*L3 to generate a candidate set of 4-itemsets, C4.

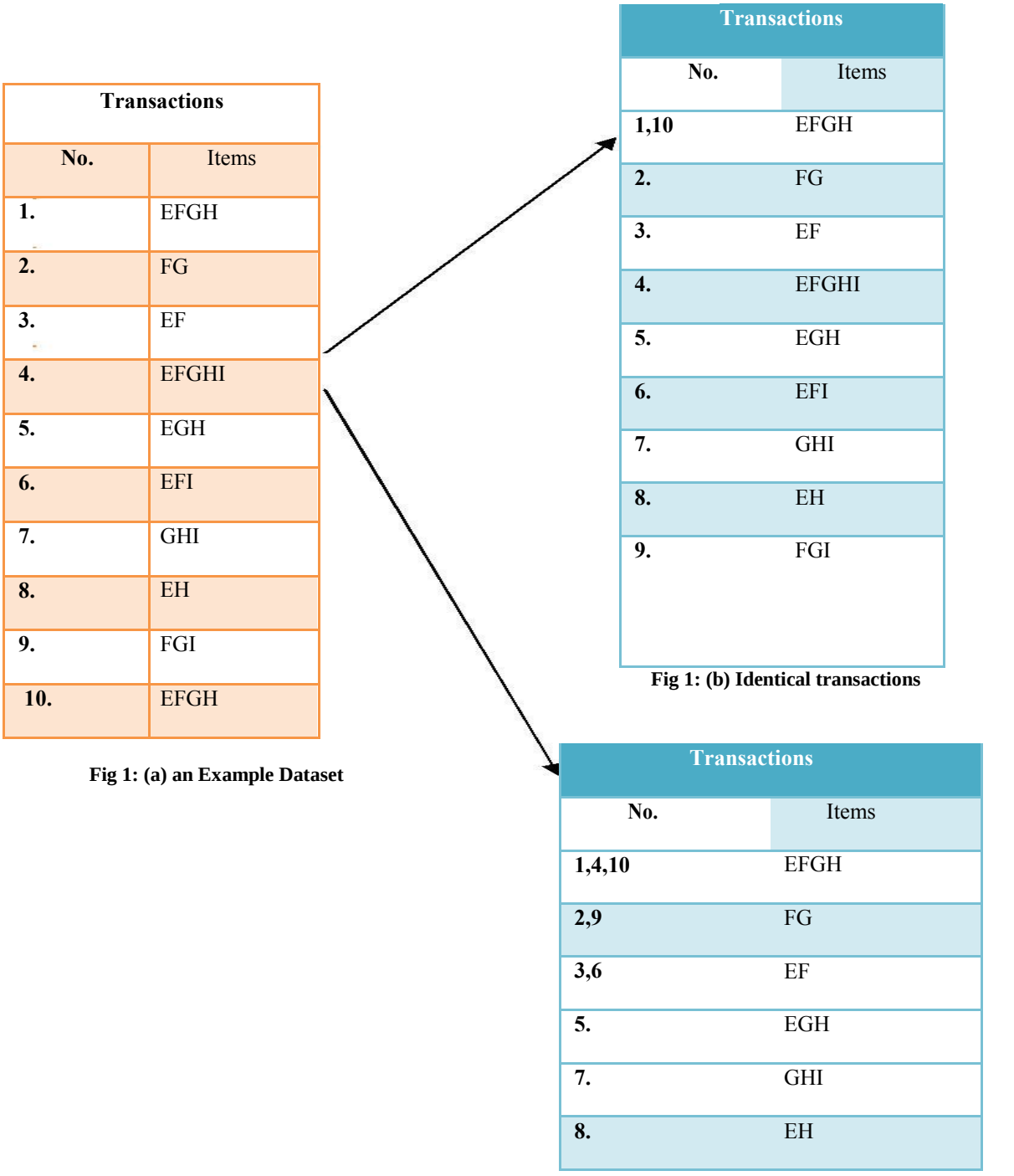
It makes numerous passes over the database. In the first pass, the algorithm simply counts item occurrences to determine the frequent 1-itemsets. A succeeding pass, say pass k, consists of two phases. First, the frequent item sets L_{k-1} found in the $(k-1)^{th}$ pass are used to produce the candidate item sets C_k, using the apriori-gen() function. This function first joins L_{k-1} with L_{k-1}, the joining condition being that the lexicographically ordered first k-2 items are the same. Next, it deletes all those item sets from the join result that have some (k-1)-subset that is not in L_{k-1} yielding C_k. The algorithm now scans the database. For each transaction, it determines which of the candidates in C_k are contained in the transaction using a hash-tree data structure and increments the count of those candidates **[8]**, at the end of the pass, C_k is examined to determine which of the candidates are frequent, yielding L_k. The algorithm terminates when L_k becomes empty.

III. OPTIMIZED DISTRIBUTED MINING ALGORITHM

The presentation of Apriori ARM algorithms degrades for diverse reasons. It requires *n* number of database scans to generate a frequent *n*-itemset. Furthermore, it doesn't distinguish transactions in the data set with identical item sets if that data set is not loaded into the main memory. Therefore, it unnecessarily occupies resources for repeatedly generating item sets from such identical transactions. For example, if a data set has 10 identical transactions, the AA not only specifies the same candidate item sets 10 times but also updates the support counts for those candidate item sets 10 times for each iteration.

Moreover, directly loading a raw data set into the main memory won't find an important number of identical transactions because each transaction of a raw data set contains both frequent and infrequent items. To conquer these troubles, candidate support counts can't be supported from the raw data set after the first pass. This technique not only reduces the average transaction length but also reduces the data set size significantly, so we can accumulate more transactions in the main memory. The number of items in the data set might be large, but only a few will satisfy the support threshold (TH).

Consider the sample data set in Figure 1a. The data set is loaded into the main memory, and then only one identical transaction (EFGH) is found, as Figure 1b shows. However, if the data set is loaded into the main memory after eliminating rare items from every transaction, more identical transactions are found (as shown in Figure 1c). This technique not only reduces average transaction size but also finds more identical transactions.



Transactions	
No.	Items
1.	EFGH
2.	FG
3.	EF
4.	EFGHI
5.	EGH
6.	EFI
7.	GHI
8.	EH
9.	FGI
10.	EFGH

Fig 1: (a) an Example Dataset

Transactions	
No.	Items
1,10	EFGH
2.	FG
3.	EF
4.	EFGHI
5.	EGH
6.	EFI
7.	GHI
8.	EH
9.	FGI

Fig 1: (b) Identical transactions

Transactions	
No.	Items
1,4,10	EFGH
2,9	FG
3,6	EF
5.	EGH
7.	GHI
8.	EH

Fig 1: (c) Transactions after pruning infrequent items

5. PADA RULE WITH XML DATA

Parallelism is predictable to relieve current ARM methods from sequential blockages, providing the ability to scale to enormous datasets and improving the response time. The parallel design space spans three main components including the hardware platform, the kind of parallelism broken and the load balancing strategy used. Shared memory architecture has all the processors access common memory. Each processor has direct and equal access to all the memory in the system. Parallel programs are easy to execute on such a system **[2]**. The data warehouse (DW) is partitioned among 'P' processors logically. Each processor works on its local partition of the database but performs the same computation of counting support. Dynamic load balancing seeks to address this issue by balancing the load and reassigning the loads to the lighter ones. The development of distributed rule mining is a challenging and vital task, since it requires knowledge of all the data stored at different locations and the ability to combine partial results from individual databases into a single result.

The AR from XML data with a sample XML document is considered. For example, the set of transactions are identified by the tag <transactions> and each transaction in the transactions set is identified by the tag <transaction>. The set of items in each transaction is: Transaction document (transactions.xml) is identified by the tag <items> and an item is identified by the tag <item>. Consider the problem of mining all ARs among items that emerge in the transactions document. With the understanding of traditional AR mining is expected to obtain the large item sets document and ARs document from the source document.

Let the minimum support (minsupp) = 35% and minimum confidence (minconfi) = 99%.

6. PERFORMANCE ASSESSMENT

The number of messages that ODAM exchanges among various locations to generate the globally frequent item sets in a distributed environment, the original data set is partitioned into five partitions. To decrease the dependency among dissimilar partitions, each one contains only 25 percent of the original data set's transactions. So, the number of identical transactions among different partitions is very low. ODAM provides a proficient method for generating ARs from different datasets, distributed among various locations.

The datasets are generated arbitrarily depending on the number of different items, the maximum number of items in each transaction and the number of transactions. The performance of the XQuery implementation is dependent on

the number of large item sets found and the size of the dataset as shown in the Fig 2.

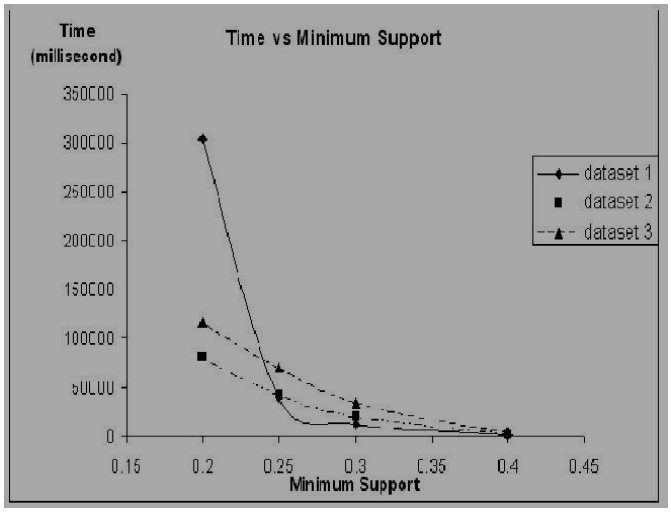


Fig 2: Time with Minimum support

The running time for dataset-1 with minimum support 20% is much higher than the running time of dataset-2 and dataset-3, since the number of large item sets found for dataset-1 is about 2 times more than the other datasets. The Response time of the parallel and distributed data mining task on XML data is carried out by the time taken for communication, computation cost involved **[6]**. Communication time is largely dependent on the DDM operational model and the architecture of the DDM systems. The computation time is the time to perform the mining process on the distributed data sets.

7. CONCLUSIONS

AR mining is a vital problem of DM. It's a new and challenging area to perform AR mining on XML data due to the difficulty of XML data. In our approach, numerous problems in XML data is handled suitably to assure the correctness of the result. The ODAM Algorithm is used for the mining process in a parallel and distributed setting. The response time with the communication and computation factors are measured to achieve an improved response time. The performance examination is done by increasing the number of processors in a distributed environment. As the mining process is done in parallel an optimal solution is

obtained.

REFERENCES

[1] R. Agrawal and R. Srikant , "Fast Algorithms for Mining Association Rules in Large Database,"*Proc. 20th Int'l Conf. Very Large Databases* (VLDB 9 4), Morgan Kaufmann, 1994,pp. 407-419.

[2] R. Agrawal and J.C. Shafer, "Parallel Mining of Association Rules,"*IEEE Tran. Know ledge and* 16 IEEE Distributed Systems Online March 2004 *Data Eng. ,* vol. 8, no. 6, 1996,pp. 962-969;.

[3] D.W. Cheung , et al., "A Fast Distributed Algorithm for Mining Association Rules," *Proc. Parallel and Distributed Information Systems,* IEEE CS Press, 1996, pp. 31-42;

[4] A. Savasere, E. Omiecinski, and S .B. Navathe , "An Efficient Algorithm for Mining Association Rules in Large Database,"P*roc. 21st Int'l Conf. Very Large Databases* (VLDB 94), Morgan Kaufmann, 1995, pp. 432-444.

[5] J. Han , J. Pei, and Y. Yin , "Mining Frequent Patterns without Candidate Generation,"*Proc. ACM SIGMOD Int'l. Conf. Management of Data,* ACM Press, 2000,pp. 1-12.

[6] M.J. Zaki and Y. Pin, "Introduction: Recent Developments in Parallel and Distributed Data Mining,"*J. Distributed and Parallel Databases,* vol. 11, no. 2, 2002, pp. 123-127.

[7] M.J. Zaki , "Scalable Algorithms for Association Mining,"*IEEE Trans. Knowledge and Data Eng.,* vol.12 no. 2, 2000,pp. 372-390;

[8] J.S. Park , M. Chen, and P.S. Yu , "An Effective Hash Based Algorithm for Mining Association Rules,"*Proc. 1995 ACM SIGMOD Int'l Conf. Management of Data,* ACM Press, 1995, pp. 175-186.

[9] M.J. Zaki , et al., *Parallel Data Mining for Association Rules on Shared-Memory Multiprocessors ,* tech. report TR 618, Computer Science Dept., Univ. of Rochester, 1996.

[10] D.W. Cheung , et al., "Efficient Mining of Association Rules in Distributed Databases,"*IEEE Trans. Knowledge and Data Eng.,* vol. 8, no. 6, 1996,pp.911-922;

