Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 79 (2016) 127 – 134

Procedia Computer Science

7th International Conference on Communication, Computing and Virtualization 2016

Performance Evaluation of Distributed Association Rule Mining Algorithms

Ms. Vinaya Sawant[a]*, Dr. Ketan Shah[b]

Assistant Professor, IT Department, D. J. Sanghvi College of Engineering, Mumbai, India
Professor, IT Department, MPSTME, Mumbai, India

Abstract

Association Rule Mining (ARM) is a popular and well researched method for discovering interesting relations between variables in large databases. It is intended to identify strong rules discovered in databases using different measures of interestingness. Most ARM algorithms focus on a sequential or centralized environment where no external communication is required. Distributed ARM algorithms (DARM), aim to generate rules from different data sets spread over various geographical sites; hence, they require external communications throughout the entire process. DARM algorithm efficiency is highly dependent on data distribution. The Classical algorithms used in DARM are Count Distribution Algorithm (CDA), Fast Distributed Mining (FDM) Algorithm and Optimized Distributed Association Mining (ODAM) Algorithm. This paper presents the implementation details and experimental results of above mentioned algorithms. The paper also highlights the issues of message exchange size in a distributed environment of current DARM algorithms that can affect the communication costs in a distributed environment.

Peer-review under responsibility of the Organizing Committee of ICCCV 2016

Keywords: Association Rule Mining, Distributed Data Mining

1. Introduction

Distributed ARM algorithms aim to generate rules from different data sets spread over various geographical sites; hence, they require external communications throughout the entire process. DARM algorithms must reduce

* Corresponding author. Tel.:91 9867248114
E-mail address: vinaya.sawant@djsce.ac.in

Peer-review under responsibility of the Organizing Committee of ICCCV 2016
doi:10.1016/j.procs.2016.03.017

communication costs so that generating global association rules costs less than combining the participating sites' data sets into a centralized site. However, most DARM algorithms don't have an efficient message optimization technique, so they exchange numerous messages during the mining process.

In this study, I assume that the database to be studied is a transaction database although the method can be easily extended to relational databases as well. The database consists of a huge number of transaction records, each with a transaction identifier (TID) and a set of data items. Further, I assume that the database is "horizontally" partitioned (i.e., grouped by transactions) and allocated to the sites in a distributed system which communicate by message passing. Based on these assumptions, we examine distributed mining of association rules. It has been well known that the major cost of mining association rules is the computation of the set of large itemsets (i.e., frequently occurring sets of items) in the database. Distributed computing of large itemsets encounters some new problems. One may compute locally large itemsets easily, but a locally large itemset may not be globally large. Since it is very expensive to broadcast the whole data set to other sites, one option is to broadcast all the counts of all the itemsets, no matter locally large or small, to other sites.

However, a database may contain enormous combinations of itemsets, and it will involve passing a huge number of messages. Based on our observation, there exist some interesting properties between locally large and globally large itemsets. One should maximally take advantages of such properties to reduce the number of messages to be passed and confine the substantial amount of processing to local sites.

DARM refers to the mining of association rules from distributed datasets. The datasets are stored in local databases hosted by local computers which are connected through a computer network. Typical DARM algorithm involves local data analysis from which a global knowledge can be extracted using knowledge integration techniques [1].

2. DARM Algorithms

Most of the research efforts are focusing on the generation of frequent itemsets as compared to generation of association rules. Following paragraph describes the experimental results of various Distributed Association Rule Algorithms used in research work.

2.1. Count Distribution Algorithm (CDA)

One data parallelism algorithm is the count distribution algorithm (CDA). The database is divided into p partitions, one for each processor. Each processor counts the candidates for its data and then broadcasts its counts to all other processors. Each processor then determines the global counts. These then are used to determine the large itemsets and to generate the candidates for the next scan [2].

2.2. Fast Distributed Mining Algorithm (FDM)

The generation of candidate sets is in the same spirit of Apriori. However, some interesting relationships between locally large sets and globally large ones are explored to generate a smaller set of candidate sets at each iteration and thus reduce the number of messages to be passed. After the candidate sets have been generated, two pruning techniques, local pruning and global pruning, are developed to prune away some candidate sets at each individual site [3].

2.3. Optimized Distributed Association Mining Algorithm (ODAM)

To efficiently generate candidate support counts of later passes, ODAM eliminates all infrequent items after the first pass and places those new transactions into the main memory. 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. Moreover, the number of infrequent itemsets increases proportionally for higher support thresholds [4].

3. Parameters for Evaluation

One of the parameters used for evaluation of the above mentioned algorithms is total number of frequent itemsets generated in a given time period by an algorithm using varying values of support count in a distributed environment using single node, two nodes and three nodes. Another parameter used for evaluation is the messages (size in bytes) passed in a distributed environment between the nodes.

4. Experimental Setup

The algorithms were tested in a distributed environment developed using JAVA programming language. It was implemented on one to three nodes and a server. The configuration of each workstation on the network was an Intel Core i3-2100 CPU @3.10GHZ, 4GB RAM, 32-bit OS, and Windows 7. Remote Method Invocation Mechanism (RMI) is used for communications between the nodes in the network.

The number of sites used in the distributed environment is decided first. The aim is to perform the comparison with respect to support thresholds and database sizes. The datasets were horizontally partitioned depending upon the number of processors in the distributed environment and the results are analyzed. The candidate itemsets that each site generates will be based on global frequent itemsets of the previous pass, the dataset is divided equally among the sites.

5. Datasets

The datasets from UCI Machine Learning Repository was used for testing the performance of CDA, FDM and ODAM in a distributed environment [5]. The following are the different datasets used for testing the results.

A. Zoo Dataset
   - 17 items
   - 101 transactions (4KB)

B. Tic-Tac-Toe Dataset
   - 9 items
   - 958 transactions (18KB)

C. Plant Signaling Dataset
   - 43 items
   - 5456 transactions (469KB)

D. 10,000 x 8 Databases
   - 8 items
   - 10000 transactions(157 KB)

E. The USCensus1990 raw data set was obtained from the (U.S. Department of Commerce) Census Bureau website using the Data Extraction System.
   - 69 items
   - 2458211 transactions (344 MB)

6. Experiments Performed

The CDA, FDM and ODAM algorithm were implemented according to the experimental setup given in Section 4. The Server and the nodes communicate in a network to generate the frequent itemsets after each pass. Initially, the complete dataset at single node is used to find the frequent k-itemsets. The input to the algorithm consists of two files: config file and transaction file. The config file contains the details about the number of items and the support count and the transaction file contains all the transactions. The output is the time taken to generate the frequent k-

itemsets. For two nodes, the transaction file is horizontally fragmented into two files and assigned to two nodes for finding the frequent k-itemsets. For three nodes, the transaction file is horizontally fragmented into three files and assigned to three nodes for finding the frequent k-itemsets.

7. Experimental Results

7.1. Total number of Frequent Itemsets generated in a Time Period

The following results show the number of frequent itemsets generated in a given time period for Zoo and Tic Tac Toe dataset on single node, two nodes and three nodes using CDA, FDM and ODAM Algorithms.

The Table 1 shows the comparison between CDA, FDM and ODAM execution time for Zoo Dataset on a single node, Table 2 shows on 2 nodes and Table 3 shows on 3 nodes.

Table 1: CDA, FDM and ODAM for Zoo Dataset on Single Node

Algorithms		CDA		FDM		ODAM	
Parameters		TIME (sec)	No. of Frequent Itemsets	TIME (sec)	No. of Frequent Itemsets	TIME (Sec)	No. of Frequent Itemsets
	Support	1 Node	1 Node	1 Node	1 Node	1 Node	1 Node
	20%	1.51	683	1.472	679	1.48	654
Zoo	10%	1.491	1436	1.422	1521	1.45	1514
Dataset	8%	1.823	1689	1.786	1688	1.79	1650
	6%	1.456	1903	1.831	1968	1.76	1945
	4%	2.411	2899	2.409	2899	2.4	2884

Table 2: CDA, FDM and ODAM on Zoo Dataset using 2nodes

Algorithms		CDA		FDM		ODAM	
Parameters		TIME (sec)	No. of Frequent Itemsets	TIME (sec)	No. of Frequent Itemsets	TIME (Sec)	No. of Frequent Itemsets
	Support	2 Nodes	2 Nodes	2 Nodes	2 Nodes	2 Nodes	2 Nodes
	20%	1.571	356	1.207	380	1.082	373
Zoo	10%	1.4455	688	1.393	749	1.2895	738
Dataset	8%	1.721	779	1.637	782	1.5865	765
	6%	1.625	956	1.784	962	1.679	952
	4%	2.522	1387	2.303	1387	2.217	1374

Table 3: CDA, FDM and ODAM on Zoo Dataset using 3nodes

Algorithms		CDA		FDM		ODAM	
Parameters		TIME (sec)	No. of Frequent Itemsets	TIME (sec)	No. of Frequent Itemsets	TIME (Sec)	No. of Frequent Itemsets
	Support	3 Nodes	3 Nodes	3 Nodes	3 Nodes	3 Nodes	3 Nodes
	20%	1.318	396	0.965	383	0.731	370
Zoo	10%	1.541	696	1.3	695	1.253	682
Dataset	8%	1.973	695	1.557	695	1.6253	681
	6%	2.04	941	1.735	941	1.5424	939
	4%	2.302	940	1.984	941	1.875	931

The Table 4 shows the comparison between CDA, FDM and ODAM execution time for Tic Tac Toe Dataset on a single node, Table 5 shows on 2 nodes and Table 6 shows on 3 nodes.

Table 4: CDA, FDM and ODAM on Tic Tac Toe Dataset using 1 node

Algorithms		CDA		FDM		ODAM	
Parameters		TIME (sec)	No. of Frequent Itemsets	TIME (sec)	No. of Frequent Itemsets	TIME (Sec)	No. of Frequent Itemsets
	Support	1 Node	1 Node	1 Node	1 Node	1 Node	1 Node
TTT Dataset	20%	0.203	9	0.23	9	0.27	9
	10%	0.25	45	0.25	45	0.39	45
	8%	0.281	53	0.29	53	0.44	53
	6%	0.328	53	0.32	53	0.42	53
	4%	0.391	89	0.4	89	0.45	89

Table 5: CDA, FDM and ODAM on Tic Tac Toe Dataset using 2 nodes

Algorithms		CDA		FDM		ODAM	
Parameters		TIME (sec)	No. of Frequent Itemsets	TIME (sec)	No. of Frequent Itemsets	TIME (Sec)	No. of Frequent Itemsets
	Support	2 Nodes	2 Nodes	2 Nodes	2 Nodes	2 Nodes	2 Nodes
TTT Dataset	20%	0.3285	14	0.316	21	0.2825	22
	10%	0.219	39	0.331	46	0.311	45
	8%	0.2895	41	0.305	53	0.356	54
	6%	0.3205	62	0.363	62	0.375	62
	4%	0.445	94	0.381	94	0.3775	95

Table 6: CDA, FDM and ODAM on Tic Tac Toe Dataset using 3 nodes

Algorithms		CDA		FDM		ODAM	
Parameters		TIME (sec)	No. of Frequent Itemsets	TIME (sec)	No. of Frequent Itemsets	TIME (Sec)	No. of Frequent Itemsets
	Support	3 Nodes	3 Nodes	3 Nodes	3 Nodes	3 Nodes	3 Nodes
TTT Dataset	20%	0.25	5	0.2803	21	0.2389	21
	10%	0.265	42	0.3843	42	0.38	42
	8%	0.307	56	0.298	56	0.2757	55
	6%	0.344	71	0.3927	63	0.3267	73
	4%	0.373	91	0.3283	91	0.354	95

All the algorithms require more time when generating longer candidate itemsets. Generating support counts of candidate itemsets for each iteration takes approximately three times longer than it takes for the previous iteration. This is identical for all algorithms. However, ODAM removes a significant number of infrequent 1-itemsets from every transaction after the first pass, so it finds a significant number of identical transactions.

After eliminating infrequent items, ODAM doesn't enumerate candidate itemsets multiple times for any identical transaction. Furthermore, it requires a minimal number of comparison and update operations to generate support because it doesn't require comparison and update operations multiple times for similar transactions. In contrast,

CDA takes longer because each transaction contains all items, thus requiring numerous comparison and update operations to candidate itemsets' generate support counts.

The Figure 1 represents the time taken by CDA, FDM and ODAM on Zoo Dataset using 3 nodes and Figure 2 represents total number of frequent itemsets generated on Zoo dataset using 3 nodes. The Figure 3 represents the time taken by CDA, FDM and ODAM on Tic Tac Toe Dataset using 2 nodes and Figure 4 represents total number of frequent itemsets generated on Tic Tac Toe dataset using 2 nodes.

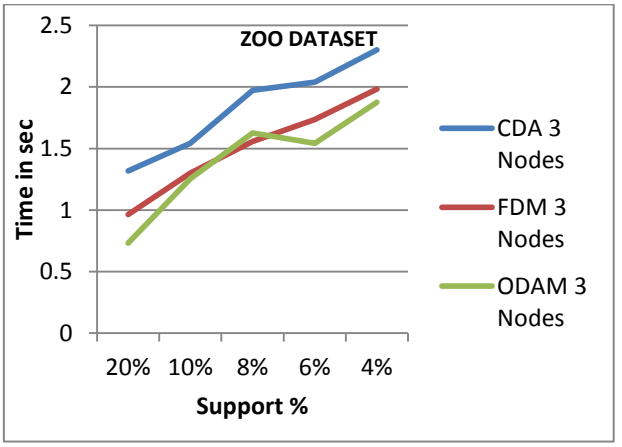

Figure 1: CDA, FDM and ODAM Comparison on Zoo Dataset using 3 nodes

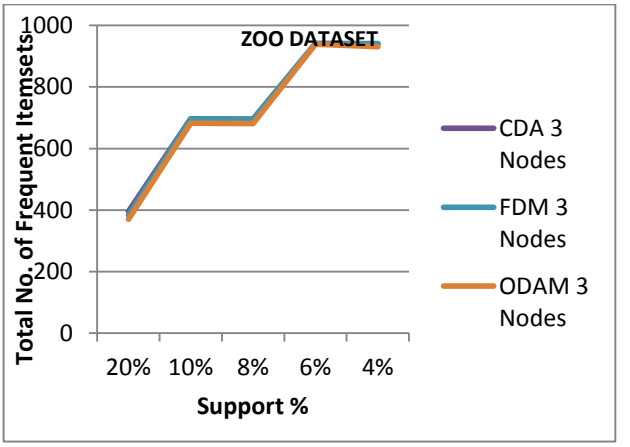

Figure 2: CDA, FDM and ODAM Comparison on Zoo Dataset using 3 nodes

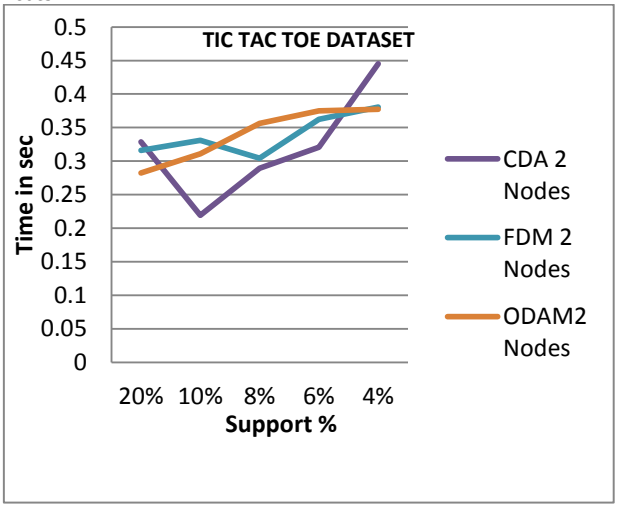

Figure 3:CDA, FDM and ODAM Comparison on Tic Tac Toe Dataset using 2 nodes

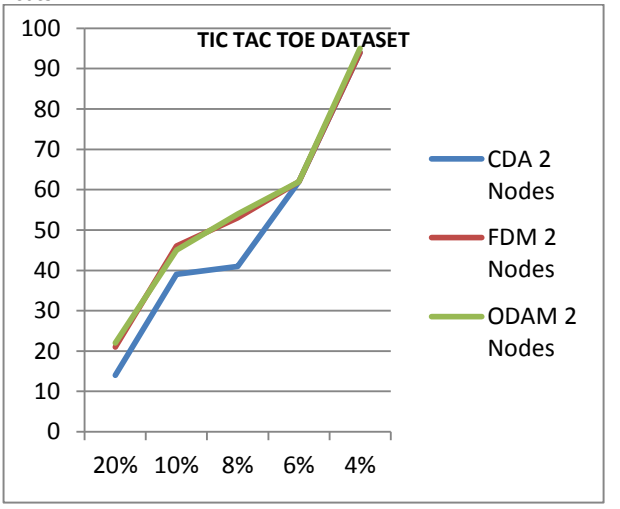

Figure 4:CDA, FDM and ODAM Comparison on Tic Tac Toe Dataset using 2 nodes

7.2. *Message Exchange Optimization*

In the CDA algorithm, each local site generates support counts and broadcasts them to all other sites to let each site calculate globally frequent itemsets for that pass. So, the total number of messages broadcast from each site equals (n - 1 * $|C|$). We can calculate the total message size using

$$T=\sum_{i=1}^{n}(n-1)*C$$

Where n is the total number of sites and C is number of candidate itemsets.

In FDM, each site broadcasts locally large frequent large itemsets to all other sites. After receiving a request, each

polling site sends a polling request to all remote sites other than the originator site. Upon receiving the polling request from all other sites, the polling site computes whether that candidate itemset is globally frequent and broadcasts only globally frequent itemsets to all other sites. Hence, it exchanges more messages because each polling site sends and receives support counts from remote sites. It also needs to send global support counts to all participating sites when a candidate itemset is heavy and subsequently increases the communication cost. Furthermore, each polling site receives polling requests only from one site.

$$T=\sum_{i=1}^{n}(n-1)*LL+(n-1)*FG$$

Where n is the total number of sites and LL is the locally large items and FG is globally large frequent itemsets.

In contrast with CDA, ODAM sends support counts of candidate itemsets to a single site, which calculates the globally frequent itemsets for that pass. We refer to the sites that send local support counts as the *sender* and the site that generates the globally frequent itemsets is the *receiver*. For example, with three sites, two broadcast their local support counts of candidate itemsets to the third site. The third site is responsible for generating that iteration's globally frequent itemsets. The total number of messages broadcast from a sender site to a receiver site equals (1 * |*C*|).

Once the receiver site generates globally frequent itemsets, it broadcasts them to all sender sites. The total number of messages broadcast from the receiver is (*n* - 1 * |*FG*|). We can calculate the total message broadcasting size (the aggregate of sender and receiver sites messages) using

$$T=\sum_{i=1}^{n}C+(n-1)*FG$$

Where n is the number of sites, *C* is the candidate itemsets, and *FG* is the globally frequent itemsets.

To compare the number of messages that ODAM, FDM, and CDA exchange among various sites to generate the globally frequent itemsets in a distributed environment, we partition the original data set into three partitions. Figure 5 and Figure 6 depicts the total size of messages (that is, number of bytes) that ODAM, FDM, and CDA transmit to generate the globally frequent itemsets with different support values for two different datasets.

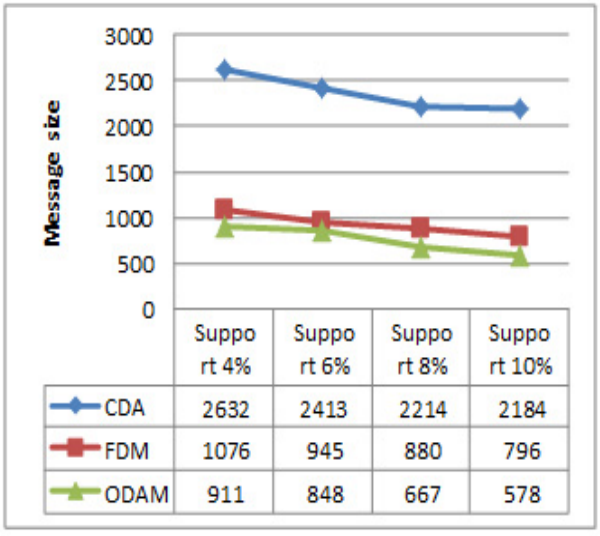

Figure 5:Total Message Size (bytes) for TTT dataset

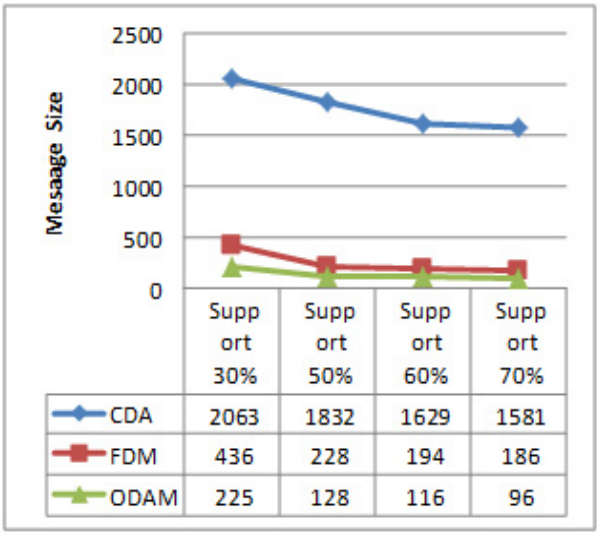

Figure 6:Total Message Size (bytes) for US Census dataset

As above figures shows, ODAM exchanges fewer messages among different sites to generate globally frequent itemsets. In all cases, ODAM reduces the communication cost by 50 to 80 percent compared to CDA. In each site, CDA exchanges messages with all other sites after every pass, and consequently the message exchange size increases when we increase the number of sites. ODAM reduces communication cost by 20 to 45 percent compared with FDM because FDM sends each support count to the polling site. After receiving a request, each polling site sends a polling request to all remote sites other than the originator site. Upon receiving the polling request from all other sites, the polling site computes whether that candidate itemset is globally frequent and broadcasts only

globally frequent itemsets to all other sites. Hence, it exchanges more messages because each polling site sends and receives support counts from remote sites. It also needs to send global support counts to all participating sites when a candidate itemset is heavy and subsequently increases the communication cost. Furthermore, each polling site receives polling requests only from one site. Therefore, without receiving the support counts of remote sites, we can't presume whether an itemset is heavy.

8. Discussion of Results

Performance of the algorithm depends on the number of nodes and number of transactions. The algorithms will take smaller execution time if we increase the number of nodes and also if the number of transactions are more.

Also, the algorithms will take longer execution time if we increase the number of nodes but the number of transactions is less.

Scalability is relative to support factors for a larger datasets. Performance increases for a smaller support factor if there is increase in the number of nodes for larger datasets.

Most DARM algorithms don't have an efficient message optimization technique, so they exchange numerous messages during the mining process.

The communication cost in DARM can be determined by the number of messages exchanged. ODAM exchanges fewer messages as compared to CDA and FDM. As we decrease the support count, the message size decreases as there are fewer candidates generated for higher support counts as compared to the lower counts. Also, the execution time taken by an algorithm increases if there more number of messages are passed in a distributed environment. ODAM algorithm can be further modified to decrease the communication cost further in a dynamic distributed environment and also can be extended to work on vertically partitioned datasets.

Researchers in this area should also focus more on developing algorithms and architectures that will be work on real data sets for Distributed Association Rule Mining. Future algorithms and methods should also consider the development of fault-tolerant and easily extendable systems in the area of distributed association rule mining. Such systems will greatly reduce communication and interpretation costs; improve efficiency and scalability of the DARM system, all of which are common issues with existing systems.

References

1. Park, Byung-Hoon, and Hillol Kargupta. "Distributed Data Mining: Algorithms, systems, and applications" (2002).
2. Rakesh Agrawal, John C. Shafer, “Parallel Mining of Association Rules”, IEEE Transactions on Knowledge and Data Engineering, Volume 8, no.6, December 1996.
3. David W. Cheung, Jiawei Han, Vincent T. Ng, Ada W. Fu and Yongjian Fu, “A Fast Distributed Algorithm for Mining Association Rules”, Fourth International Conference on Parallel and Distributed Information Systems, 1996.
4. Mafruz Zaman Ashrafi, David Taniar and Kate Smith, “ODAM: An Optimized Distributed Association Rule Mining”, IEEE Distributed SYSTEMS ONLINE, Volume 5, No.3, March 2004.
5. Bache, K. & Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.
6. Ogunde, A. O., et al. "A review of some issues and challenges in current agent based distributed association rule mining." Asian Journal of Information Technology 10.2, 2011.