The Design of an Adaptive Incremental Association Rule Mining System

Adewale O. Ogunde, *Member, IAENG*

Olusegun Folorunso, and Adesina S. Sodiya

***Abstract*— One major challenge still faced by most Distributed Association Rule Mining (DARM) systems is the inability of existing systems to adapt to constantly changing databases and mining environments. In this work, an Adaptive Incremental Mining Algorithm (AIMA) is therefore proposed to address these problems. AIMA was designed to adapt to changes in the distributed databases by mining only the incremental database updates and using this to update the existing rules in order to improve the overall response time of the DARM system. In this system, global association rules were integrated incrementally from one data site to another. The mining agents in AIMA were made adaptive by defining mining goals with reasoning and behavioral capabilities and protocols that enabled them to either maintain or change their goals. AIMA employed Java Agent Development Environment Extension for designing the internal agents' architecture. Results from experiments conducted on real datasets showed that the adaptive system, AIMA performed better than the non-adaptive systems.**

***Index Terms*— Adaptivity, Data Mining, Distributed Association Rule Mining, Incremental Mining, Mobile Agents.**

I. Introduction

MANY businesses have become reliant on data mining patterns and trends to make important decisions [1]. Association rule mining (ARM) is a very significant data mining method, which finds frequent patterns, associations, correlations, or casual structures sets of items or objects in these databases. Mining association rules has received a lot of attention to the data mining community [2]. Majority of modern organizations are now geographically distributed; hence most databases are now of distributed nature where each local database stores it's ever increasing amount of day-to-day data. Distributed Association Rule Mining (DARM) environments are constantly changing in real-life situations and there is a great need for systems that will efficiently respond to these real-time changes in the database and the entire system over time. This implies that both expected and unexpected events may occur in data mining environments and these may seriously affect the outcome or the overall performance or successful completion of mining tasks, most especially in distributed settings and databases. There is therefore an urgent need for distributed association rule mining systems that would adapt to both expected and unforeseen changes in the distributed databases and in the environment where mining agents will operate. Therefore, an adaptive incremental association rule mining system, which is improvement of our past works, was designed in this work to mine across distributed databases while adapting to expected and unexpected changes in the database and the entire system, using mobile agents.

Manuscript received March 10, 2015.

A. O. Ogunde is with the Department of Computer Science, Redeemer's University, P.M.B 230, Ede, Osun State, Nigeria, CO 80305 USA (+2348054579091; e-mail: ogundea@run.edu.ng).

O. Folorunso is with the Department of Computer Science, Federal University of Agriculture, Abeokuta, Nigeria. (e-mail: folorunsolusegun@yahoo.com).

A. S Sodiya is with the Department of Computer Science, Federal University of Agriculture, Abeokuta, Nigeria. (e-mail: sinaronke@yahoo.co.uk).

II. Literature Review

A. *Distributed Association Rule Mining (DARM)*

Data Mining (DM) is the process of extracting novel, hidden but useful hidden knowledge from data in databases [3]. DM techniques have shown very good results when applied successfully in many problem areas. DARM is the semi-automatic pattern extraction of frequent patterns and rules from distributed data sources [4]. In a DARM environment, often one needs to handle multiple large databases. No algorithm can claim to be the best in all circumstances out of all the algorithms already proposed in this area of study [5]. Sound solutions to the many multi-database mining problems can be obtained through the model of local pattern analysis [2].

ZigZag algorithm was proposed by Otey et al. [6]. In this algorithm, the data is initially distributed on different sites before the mining task. Schuster et al. [7] furthered their earlier work by proposing a distributed sampling algorithm called D-Sampling. D-Sampling assumed a centralized dataset and later distributes it during execution. In this work, each node is responsible for a set of items. Ariwa et al. [4] also proposed an optimized Incremental Knowledge Integration (OIKI) model. Here, , the client controls movement of the mining results among data servers for local integration before finally transferring the results to the client.

Silvestri and Orlando [8] proposed an algorithm called Distributed Approximate Mining of Frequent Patterns. This algorithm performs the distributed exact computation of locally frequent itemsets for inferring the local support of locally unfrequented itemsets. To further this work, [9] proposed a perspective on DDM algorithms in the context of multi-agent systems, discussing broadly the connection between DDM and MAS. Byrd and Franke [10] worked on the state of the art distributed clustering for DDM and frequent itemset mining. Furthermore, [11] proposed a distributed algorithm for frequent itemsets generation on heterogeneous clusters and grid environments called Distributed Frequent Itemsets Mining in Heterogeneous Platforms. The proposed approach uses a dynamic workload management through a block-based partitioning, and takes into account inherent characteristics of the Apriori algorithm related to the candidate sets generation.

Moreover, [12] proposed the New Fast UPdate algorithm (NFUP) for efficiently incrementally mining association rules from large transaction database. This method only requires scanning of the incremental database on the assumption that a lot of time is usually wasted in scanning the original database. In many situations, new information is more important than old information, such as in publication database, stock transactions, grocery markets, or web-log records [12]. Consequently, a frequent itemset in the incremental database is also important even if it is infrequent in the updated database. To mine new interesting rules in updated database, AIMA algorithm dynamically adapts to changes in the incremental database as opposed to NFUP which partitions the incremental database logically according to unit time interval. In real life situations this may not be appropriate as relatively large numbers of new transactions could be added to the database just in few days. The Adaptive Incremental Mining Algorithm (AIMA) proposed in this work was designed with great flexibility to allow users determine and choose when exactly to mine the updated database. For example, for each mining of the updated database, we may assume a dynamic incremental mining task for every seconds, or minutes or hours or days etc. Consequently, this may happen in a day, few days, weeks, months or years as the case may be, depending on how frequently new transactions are added to the database. For each case, the mined updated database eventually becomes the old database. The new and latest transactions then form the candidate for the new incremental database.

Meo et al. [13] presented a non-incremental algorithm called CARE and an unnamed constrained based incremental algorithm. CARE was specifically designed to deal with context dependent constraints on both the body and the head of association rules. CARE was implemented on a standalone database and rules mined were majorly constrained based as opposed to the method proposed in this work which is majorly on distributed data sites. Excessive constraints were avoided in this work as this could leave out some very important rules and create a robust system capable of accommodating unforeseen and real-life situations.

Badal and Tripathi [14] also proposed the VS_Apriori, which is an extension of the classical Apriori algorithm. The researchers claimed that VS_Apriori algorithm works faster than the classical and also scales well when the support threshold decreases. Rao and Vidyavathi [15] in their work employed game theory in data mining. Paul [16] also used XML data to work on DARM in conjunction with traditional global knowledge integration. Umarani and Punithavalli [17] also carried out an overview of sampling-based ARM, positing that sampling can speed-up the mining of association rules. Saravanan and Christopher [3] proposed a knowledge integration (KI) method in parallel and distributed environment with association rule mining using XML data. Details of the knowledge integration of the XML data done was not presented in the work.

Albashiri [18] worked on EMADS, an Extendible Multi-Agent Data mining System. The concept sees DARM as a community of multiple agents from several sources interacting to solve a common problem. EMADS concentrated on agent based data classification. In earlier works on EMADS [19], [20], a DM agent and task agents combines to solve the DARM task. There is a one-to-one relationship between a given data agent and a given data source. After some pre-processing was done, there were massive movements of either the entire data set or a subset of it while in the system AIMA proposed in this work, data agents do not have to forward their data at any point in time as this could be very costly for most real life applications. All sort of data movement were maximally eliminated in this work.

Albashiri [21] also described a Multi-Agent based approach to Data Mining using a Multi-Agent System (MADM) that comprises a collection of agents cooperating to solve given data mining tasks. Partitioning of datasets was used to achieve parallel/distributed ARM. A compressed set enumeration tree data structure called T-tree was used together with an associated ARM algorithm called Apriori-T. All the described methods presented DARM using various models all of which are still opened for further research.

B. JADEX Agents and Goal Revision

Agents are special software capable of carrying out autonomous action based on their design [22]. Java Agent Development Environment Extension (JADEX) is an integration of an agent middleware, with a reasoning engine in order to combine the advantages of both. Jadex allow the construction of rational agents in order to exhibit goal-directed behaviour. The Jadex Control Center (JCC) represents the main access point for all available Jadex runtime tools. The JCC itself provides its functionalities via plug-ins and is therefore quite easily extensible [23].

An agent is like a black box which receives and sends messages when viewed externally. The behaviour of a specific agent is therefore determined solely by its concrete beliefs, goals, and plans. An agent needs to adapt whenever its environment changes. Few researches have been done on agents' adaptability but there are none on the adaptivity of DARM agents [24]. Here are some of the few other researchers that have contributed in the area of agents' beliefs and goals changing [25], [26], [27], [28] and [29].

III. DESIGN METHODOLOGY

The design of AIMA was based on the definitions of the multi-agents and researchers earlier work defined in [29] and [30]. AIMA was equipped to dynamically mine the updated distributed databases based on flexible preset options. For each mining of the updated database, we assume dynamic mining task depending on the choice of the data miner even though there is a default setting for update search in the design. Consequently, updates may be searched from the distributed data sites in seconds, minutes, hours, days, weeks, months or years as the case may be, depending on how frequently new transactions are added to the database. For each case, the mined updated database eventually becomes the old database. The new and latest transactions then form the candidate for the new incremental database. The final set of frequent itemsets consists of the three following types. (1) α set: these are frequent itemsets in the updated database (DB) (2) β set: these are frequent itemsets in the old database (db) but infrequent in DB and (3) γ set: frequent itemsets in DB, but infrequent in db. It should be noted that there will be no need of representing the case where an itemset is infrequent in the DB and also infrequent in the db. Also note that *AIMA* keeps a record (date/time) and counter (*alpha*) for all the incremental mining tasks performed on all *db*. This is to know when exactly an itemset became frequent or infrequent in the DB.

AIMA scans the *db* to obtain the occurrence count of each *k-itemset*. Since the occurrence counts of F_k in DB are known in advance, the total occurrence count of arbitrary *X* is easily calculated if *X* is in one of the cases one to three described above. The frequent set of itemsets of DB is known in advance while *AIMA* scans the incremental database, that is, *db* for new frequent itemsets. This is then used to determine whether an itemset that had been frequent is still frequent or now infrequent. Also, this will also help to discover whether an initially infrequent itemset in DB is now frequent. For *db*, the process starts at 1-itemsets. Each candidate or frequent itemset has two attributes. That is (1) *X.count*: includes the occurrence count in db and (2) *X.type*: denotes one of the three types: α, β, and γ. The three sets are usually empty at the initial stage. After the *db* has been scanned, all frequent 1-itemsets are added into the α set. Each frequent 1-itemset is joined to form 2-itemset candidates. Each frequent 2-itemset is joined to form 3-itemset candidates and so on. In *db*, the process is performed like that of the *Apriori* algorithm and this is repeated until no more candidate k-itemsets can be generated in db. The occurrence count of each candidate in *db* is known after *db* is scanned. In *db*, *AIMA* determines which candidate k-itemset will become an element of α, β, or γ set. After *db* is scanned, the occurrence count is usually accumulated with that of DB. Note that *db.date* keeps the date of the system for the corresponding incremental database, db, when X becomes an element of frequent set.

A. The Design of AIMA Agents

The mining agent for the system, AIMA, in this work is referred to as Adaptive Mobile Agent Association Rule Miner (AMAARM). An adaptive MAARM, can be completely described at any point in time by the three tuple, which are its state, the adaptation plan, and the motivation degree function. The adaptive state of the AMAARM is thus the 3-tuple <MS, AS, ASS>, where MS is the Mechanism state, AS is the Adapting state, and AS is the Application-specific state for the AMAARM. The AMAARM model described here consists of two components; a *Mechanism* and an *Adapter* and all other parameters as described in [29]. The Adapter is the component that decides whether adaptation is necessary or not. If adaptation is necessary, the adapter determines how best to adapt to the current environment. The Mechanism senses the environment through the sensors, analyze them, and create a view of the environment called a *belief*. The *belief* is passed on to the adapter, which then decides whether adaptation is necessary or not. The Mechanism senses the environment through the sensors, analyses them, and creates a view of the environment called a *belief*. The *belief* is passed on to the adapter, which uses it to decide whether adaptation is necessary or not. If adaptation is needed, a new set of goals is passed on to the mechanism, which then transforms the set of goals into a set of actions to be carried out, and then carries out the actions. The *actors* are used to make any environment change specified in an action.

B. Adaptivity in AIMA

A task is executed by a mobile agent. A single mobile agent may execute multiple tasks or may clone itself to handoff some tasks to the cloned agents to be done in parallel. A mobile agent executing a mining task migrates to a data site that has the data to be mined, and performs the local mining task there. If the data is available and the environment is conducive for mining, the task is executed. Otherwise, the environment is sensed by the mechanism described in the previous sections and a *belief* of the situation is formed, which enables the mining agent to take the right decision. This was actually proven with the 99% task completion rate obtained during the experimentation and testing stage of this work. The various protocols required supporting the identified primary functions of AIMA agents and high-level interactions were defined for each agent.

IV. SYSTEM IMPLEMENTATION

The input to the system is an ARM task consisting of mining at either single or multiple data sites. Each individual data server has some specific data and resource requirements, all of which have to be satisfied before the task can be started. Association Rules are to be mined on distributed database servers, each of which has a (possibly different) set of data sizes. It is assumed that the set of database servers and the data sizes are static, and this information is available at all servers. However, the load conditions at the servers can vary with time. The aim of the system is to complete all the ARM tasks on the distributed network as fast as possible subject to dependency and resource constraints. An ARM task is fully executed by the agent - MAARM. A MAARM executing an ARM task migrates to a server that has the data that is required for the mining task, and tries to generate the frequent itemsets. If all the necessary resources at the data site are available and the environment is conducive, then the mining task is executed. Otherwise, the data server environment is sensed to get an idea of what the environmental change is and the MAARM may have to adapt to the changing environment using the hard-coded instructions programmed into it as described in section three.

A. Description of datasets

Datasets can thus be considered to be NxD tables where N is the number of columns and D is the number of records. Data used in this work were real datasets downloaded from the popular UCI Machine Learning Repository hosted by the Centre for Machine Learning and Intelligent Systems [31]. Four major benchmark data popularly used for distributed association rule experiments were downloaded and used for all the experiments. They are: Pima-Indians-Diabetes, Letter-Recognition, Connect-4 and Cover Type.

B. Virtual Implementation of the Distributed Data sites

A software tool called VMWare Workstation version 7.0.1 was used for creating the distributed environment where the datasets were stored. A total of four data sites were created on three virtual machines and the host system. Description of Datasite1 (Host System): Hard Disk - 400 GB, Memory - 3GB, Processors - 2 Core(s), Guest Operating System - Windows 7 Home Premium. Description of Datasite2: Hard Disk - 100 GB, Memory - 1 GB, Processors – 1, Guest Operating System - Windows 7 Ultimate Edition. Description of Datasite3: Hard Disk -100 GB, Memory - 1GB, Processors – 1, Guest Operating System - Windows 7 Ultimate Edition. Description of Datasite4: Hard Disk - 100 GB, Memory - 1 GB, Processors – 1, Guest Operating System - Windows 7 Ultimate Edition.

C. Experimentation and Analysis of Results

experiments were simulated on four virtual machines running on Intel (R) Core (TM) i5-2450M CPU @ 2.50GHz, 2501 MHz, 2 Core(s), 3 Logical Processor(s) Pentium(R) with 6GB of main memory running on Windows 7 Home Premium Edition. Datasets were thus distributed on four virtual machines at the most. The experiments were performed by varying the minimum support threshold between the ranges of 0% to 100% of total transactions depending on the particular dataset used. In order to evaluate AIMA, the experiments were also repeated by varying the sizes of the different datasets at various data sites in order to know how the algorithm scales in mining the incremental databases compared to existing methods that have to mine the entire databases all over. For users that selected DARM task from the ARM task page, they also have the additional option of choosing either (i) a manually executed AIMA, where user have to activate the mining agents, pick all the parameters, determine all or selected data sites to be mined or the global mining function, which is an adaptive AIMA where the entire distributed mining process is carried mainly by the agents using the default parameters and without the usual user interference or (ii) an adaptive incremental mining algorithm (AIMA) function that automatically adapts the system to changes found in any of the data sites, most especially data increments; where results are automatically processed and presented to the user by simply and quickly mining the distributed incremental database by updating the past results. For the purpose of experimentation, the minimum support of 20% and a minimum confidence of 80% were used as default values for AIMA. These values could be changed anytime by the user as the need arises.

D. Experiments of Adaptive Incremental Mining

AIMA offers the opportunity to dynamically the distributed data sites as the system adapts to any changes found in some or all of the data sites by automatically mining the concerning sites and presenting an up-to-date, real-time result to the data miner. This is done with details of significant changes that occurred in the concerned databases. The system immediately notifies the user whenever there is an update in any of the data sites and then go ahead with the mining process without the usual users' interference. The experiment was conducted on five different distributed databases named: DDB_0, DDB_1, DDB_2, DDB_3, and DDB_4 using the Pima Indian Diabetes dataset as the experimental data. Each of the distributed databases has four sites each. The first case (that is, DDB_0) is the case where no update is found in any of the four data sites. The second case (that is, DDB_1) is the case where an update is found the first data site but no update in the remaining sites and so on. A full description of the distributed databases used for the experiment with respect to whether a data site is updated or not is shown in Table 1. The normal DARM was performed with AIMA described in this work while the AIMA, the adaptive incremental mining algorithm aspect of our system was used for the incremental mining of the distributed databases. These results are automatically generated by the system and displayed with Jaspersoft iReport. The reports can be printed or saved in any format, for example, .doc, .pdf etc. as determined and chosen by the data miner.

TABLE I
DESCRIPTION OF DDB USED IN TESTING ADAPTIVE INCREMENTAL MINING

	Data site 1	Data site 2	Data site 3	Data site 4
DDB_0	No update found	No update found	No update found	No update found
DDB_1	Update found	No update found	No update found	No update found
DDB_2	Update found	Update found	No update found	No update found
DDB_3	Update found	Update found	Update found	No update found
DDB_4	Update found	Update found	Update found	Update found

Results obtained from the experiment showed that the adaptive incremental mining algorithm (AIMA) performed better than the normal DARM in mining the updated databases (fig. 1). Response time of the system actually increased for the two methods as the size of each of the distributed databases increased whenever new updates were found. It was also observed from the result that the rate of improvement of AIMA was far better than the normal method as the size of updated databases becomes bigger and bigger.

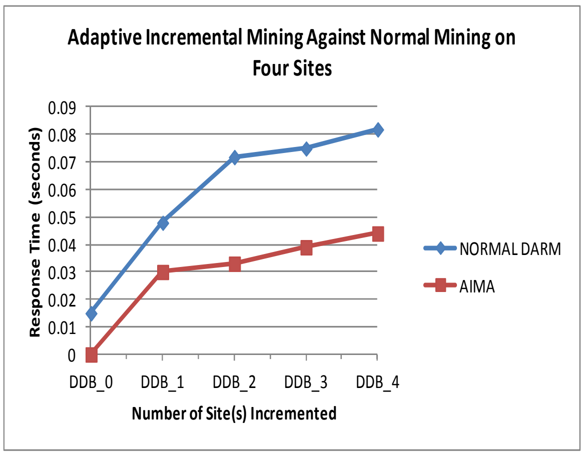

Fig. 1: Normal DARM compared with adaptive incremental DARM

E. *Performance Study on Mining Algorithm's Adaptivity*

A study was conducted to test how adaptive the mining agents, MAARM, used in the system are. For this experiment, adaptive MAARM is the mining agent hard-coded with the adaptive model described while the non-adaptive MAARM is the normal MAARM without the adaptive model. The experiment showed the number of times the mining process was completed with the adaptive MAARM and non-adaptive MAARM for all the datasets used in this work. The experiments were repeated twenty five times for each dataset, also covering each of the two options (adaptive MAARM and non-adaptive MAARM), and the number of times the DARM task completed were observed. The results showed that the adaptive MAARM performed better (99%) than the non-adaptive MAARM (94%) out of the 100 times the experiments were performed on all datasets (fig. 2).

V. Conclusion and Future Works

An important study on distributed association rule mining

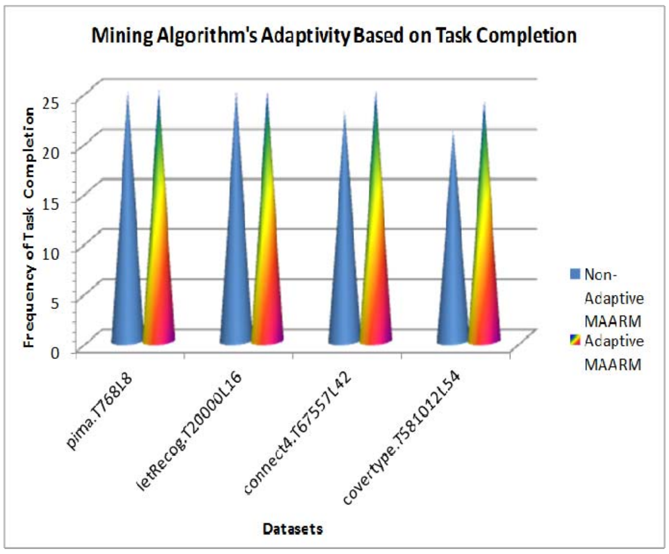

Fig. 1: Normal DARM compared with adaptive incremental DARM

area of the subject data mining was conducted in this work. The review conducted revealed that most of the current DARM algorithms are complex and also requires massive movement of data to a central memory, which is not feasible in many real life situations that has big data. Moreover, expected and unforeseen changes are bound to happen in the mining environment. For instance, most existing systems assume a static database while in real life transactional databases in science, engineering, business etc. change from time to time. This work therefore addressed a number of these challenges by tapping into the power of mobile agents, which was programmed and deployed, to mine these constantly changing distributed databases on behalf of the user. An improved system called AIMA was designed to adapt to both expected and unforeseen changes in the distributed databases and in the mining environment where the mining mobile agents will operate. The target of this design was to reduce the total response time and communication overhead usually incurred by existing DARM tasks. Adaptivity of mining agents were also hard-coded into the system.

The results of the distributed incremental association rule mining by AIMA proved that the proposed system can easily adapt to real-time changes in the database by adaptively mining the updated database and presenting detailed and interesting results to the data miner about the current situation of the updated database through the distributed incremental mining of transaction changes in these updated databases. The adaptive MAARM performed better (with 99% DARM task completion rate) than the non-adaptive MAARM (with 94% DARM task completion rate) out of the 100 times the experiments were performed on all datasets used in this work.

In addition, effective distribution and parallelization of DARM tasks was attained with great flexibility. Adaptivity was achieved both in changes occurring in the updated databases and also the mining environment and finally response time and communication overhead of distributed association rule mining tasks were generally reduced. The design, analysis and implementation of AIMA had really opened up a lot of some other explorable issues in the field of DARM; hence more efforts in the area of future works are required to fully develop the system to meet with future challenges. Future extensions of AIMA can also address more issues in the area of security and fault tolerance of the mining agents and the entire system. This could help to improve the reliability and trustworthiness of the system.

REFERENCES

[1] M.G. Kaosar, R. Paulet, X. Yi, Fully homomorphic encryption based two-party association rule mining. Elsevier Journals: Data & Knowledge Engineering. Volume 76–78 (2012) 1–15.

[2] A. Adhikari A., J. Adhikari, W. Pedrycz, Data Analysis and Pattern Recognition in Multiple Databases. Intelligent Systems Reference Library 61, @ Springer International Publishing Switzerland, (2014) 21-42.

[3] S. Saravanan, T. Christopher, A Study on Milestones of Association Rule Mining Algorithms in Large Databases. International Journal of Computer Applications, Volume 47, No.3 (2012) Pg 12-19.

[4] F.I. Ariwa, B.S. Mohamed, M.M. Mohamed, Informatization and E-Business Model Application for Distributed Data Mining Using Mobile Agents. International Conference WWW/Internet2003.

[5] K.A. Albashiri, EMADS: An Investigation into the Issues of Multi-Agent Data Mining. PhD Thesis, The University of Liverpool, Ashton Building, Ashton Street, Liverpool L69 3BX, United Kingdom, 2010.

[6] M.E. Otey, C. Wang, S. Parthasarathy, A. Veloso, J.W. Meira, Mining Frequent Itemsets in Distributed and Dynamic Databases. In ICDM 2003: Third IEEE International Conference on Data Mining (2003) 617 - 620.

[7] A. Schuster, R. Wolff, D. Trock, A high-Performance Distributed Algorithm for Mining Association Rules. In TCDM '03: Proceedings of the Third IEEE International Conference on Data Mining, Washington, DC, USA (2003) page 291.

[8] C. Silvestri, S. Orlando, Distributed Approximate Mining of Frequent Patterns. ACM Symposium on Applied Computing, Italy (2005).

[9] C. Silvestri, Distributed and Stream Data Mining Algorithms for Frequent Pattern Discovery. Ph.D. Thesis: TD-2006-4, Universit'a Ca' Foscari di Venezia.

[10] M. Byrd, C. Franke, The State of Distributed Data Mining. ECS265 Project Report UC Davis, Davis CA 95616, USA (2007).

[11] N.A.L. Khac, L.M Aouad, M. Kechadi, Distributed Knowledge Map for Mining Data on Grid Platforms. International Journal of Computer Science and Network Security, Vol.7 No.10 (2007).

[12] C. Chang, Y. Li, J. Lee, An Efficient Algorithm for the Incremental Mining of Association Rules. Proceedings of the International Workshop on Research Issues in Data Engineering: Stream Data Mining and Applications, (2005) 1-8.

[13] R. Meo, M. Botta, R. Esposito, A. Gallo, A Novel Incremental Approach to Association Rules Mining in Inductive Databases. Constraint-Based Mining and Inductive Databases, Springer Berlin Heidelberg, Lecture Notes in Computer Science Volume 3848 (2006) 267-294.

[14] N. Badal, S. Tripathi, Frequent Data Itemset Mining Using VS_Apriori Algorithms. International Journal on Computer Science and Engineering. Vol. 02, No. 04 (2010) 1111-1118.

[15] V.S. Rao, S. Vidyavathi, Distributed Data Mining and Mining Multi-Agent Data. International Journal on Computer Science and Engineering Vol. 02, No. 04 (2010) 1237-1244.

[16] S. Paul, An optimized distributed association rule mining algorithm in parallel and distributed data mining with xml data for improved response time. International Journal of Computer Science and Information Technology, Volume 2, Number 2 (2010).

[17] V. Umarani, M. Punithavalli, Sampling based Association Rules Mining - A Recent Overview. International Journal on Computer Science and Engineering Vol. 02, No. 02 (2010) 314-318.

[18] K.A Albashiri, Data Partitioning and Association Rule Mining Using a Multi-Agent System. International Journal of Engineering Science and Innovative Technology (IJESIT), Volume 2, Issue 5 (2013) 161-169.

[19] K.A. Albashiri, F. Coenen, P. Leng, EMADS: An Extendible Multi-Agent Data Miner. Knowledge-Based Systems (KBS) Journal, Volume 22, Issue 7, (2009) 523-528.

[20] K.A. Albashiri, F. Coenen, P. Leng, An investigation into the issues of Multi-Agent Data Mining. In Bouca, D. and Gafagnao, A. (Eds), Agent Based Computing, Nova Science Publishers, ISBN: 978-1-60876-684-0, 2010.

[21] K.A. Albashiri, Agent Based Data Distribution for Parallel Association Rule Mining. International Journal of Computers Volume 8 (2014) 24-32.

[22] M. Wooldridge, An Introduction to Multi-Agent Systems. John Wiley and Sons, Chichester, United Kingdom, 2009.

[23] A. Pokahr, L. Braubach, Jadex User Guide. Release 0.96, Distributed Systems Group, University of Hamburg, Germany. http://vsis-www.informatik.uni-hamburg.de/projects/jadex/, 2007.

[24] F.M.T. Brazier, N.J.E Wijngaards, Automated servicing of agents. AISB journal, Vol.1, No.1 (2001) 5-20.

[25] S. Ranjan, A. Gupta, A. Basu, A. Meka, A. Chaturvedi, Adaptive mobile agents: Modeling and a case study. 2nd Workshop on Distributed Computing "IEEE md CFP : WDC (2000).

[26] L.H. Tamargo, A.J. Garcia, M.A. Falappa, G.R. Simari, Modeling knowledge dynamics in multi-agent systems based on informants. The Knowledge Engineering Review, Cambridge University Press, DOI: 10.1017/S0000000000000000, Printed in the United Kingdom, vol. 00:0, (2010) pp. 1-31.

[27] S.M. Khan, Y. Lespérance, A Logical Framework for Prioritized Goal Change. Proceedings of the 9th Int. Conf. on Autonomous Agents and Multi-agent Systems (AAMAS 2010), Toronto, Canada, (2010) 283-290.

[28] M.B. Riemsdijk, M. Dastani, M. Winikoff, Goals in Agent Systems: A Unifying Framework. Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Sys (AAMAS 2008), May, 12-16, Estoril, Portugal, (2008) 713-720.

[29] A.O. Ogunde, O. Folorunso, A.S. Sodiya, On the Adaptivity of Distributed Association Rule Mining Agents. The Fourth International Conference on Adaptive and Self-Adaptive Systems and Applications, Adaptive 2012 Conference, IARIA July 22nd – 27th , Nice, France, www.iaria.org/conferences (2012) 69-74.

[30] A.O. Ogunde, O. Folorunso, A.S. Sodiya, J.A. Oguntuase, Towards an adaptive multi-agent architecture for association rule mining in distributed databases. In IEEE Xplore database Adaptive Science and Technology (ICAST), 2011, 3rd IEEE International Conference on Adaptive Science and Technology, Pg 31 – 36.

[31] A. Frank, A. Asuncion, UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, 2010 .