On the Adaptivity of Distributed Association Rule Mining Agents

Adewale Opeoluwa Ogunde
Dept. of Mathematical Sciences
Redeemer's University (RUN),
Redemption Camp, Nigeria
ogundea@run.edu.ng

Olusegun Folorunso
Dept. of Computer Science
Federal University of Agriculture,
Abeokuta, Nigeria
folorunsolusegun@yahoo.com

Adesina Simon Sodiya
Dept. of Computer Science
Federal University of Agriculture,
Abeokuta, Nigeria
sina_ronke@yahoo.co.uk

***Abstract*—Current and real association rule mining tasks can only be successfully done in a distributed setting where transaction data sites are mined dynamically and appropriately as they are updated. Mobile agents' paradigms are now used to mine association rules in such circumstances. As these mobile agents travel in the distributed association rules mining environment, they are liable to unforeseen changes, circumstances and faults that may arise in these environments. Few researches had been carried out on the adaptivity of mobile agents, but the adaptivity of distributed association rule mining agents is yet to be explored. Therefore, this work examines an adaptive architectural framework that mines association rules across multiple data sites, and more importantly the architecture adapts to changes in the updated database and the mining environment giving special considerations to the incremental database. This system was made adaptive both at the algorithm level and the mining agent level. Adaptation at the mobile agent level uses sensors to sense environmental changes, creates a percept of the environment and sends it to the adapter which adapts to the environmental changes by dynamically changing the goals of the mining agents or maintaining the original goals. The system promises to efficiently generate new and up-to-date rules while also adapting to faults and other unforeseen circumstances in the distributed association rules mining environment without the usual user's interference. The model presented here provided the background ideas needed for the development of adaptive distributed association rule mining agents.**

Keywords-adaptive agents; distributed association rule mining; distributed databases; knowledge integration; mobile agents

I. INTRODUCTION

Association rule mining (ARM) finds frequent patterns, associations, correlations, or casual structures sets of items or objects from large databases [1]. The idea is to find out the relation or dependency of occurrence of one item based on occurrence of other items. Distributed Association Rule Mining is the process of mining association rules and patterns from distributed data sources. Mobile agents are any relatively autonomous entity able to perform actions in an environment perceived by it. Mobile agents' paradigm [7] has several advantages among which are: conservation of bandwidth and reducing latencies while also complex, efficient and robust behaviors can be realized with surprisingly little code. Performance of these mining agents may be hampered in the distributed association rule environments due to faults and other unforeseen circumstances. Therefore, in this research, we capitalized on the power of agents to introduce an adaptive distributed association rule mining agents that mines across distributed databases while adapting to unforeseen changes in the entire system. The organization of the rest of this paper is as follows. Section II provides a review of some existing and related works. Section III describes the design details of the adaptive distributed association rule mining agents. Finally, section IV contains some concluding remarks and scope for future work.

II. LITERATURE REVIEW

This section reviews existing work on distributed association rule mining, agents and adaptive systems.

A. Association Rule Mining

Association Rule Mining (ARM) is one of the most popular tasks of Data Mining (DM). Data mining is a powerful new technology with great potential to help companies focus on the most important information in the data they have collected about the behavior of their customers and potential customers [2]. It finds patterns in data that show associations between domain elements. DM is generally focused on transactional data, such as a database of purchases at a store. This task is known as Association Rule Mining (ARM), and was first introduced in Agrawal et al. [1]. An association rule is of the form X → Y, where X and

 ISBN: 978-1-61208-219-6

Y are disjoint conjunctions of attribute-value pairs. The most commonly used mechanism for determining the relevance of identified ARs is the support and confidence framework. The confidence of the rule is the conditional probability of Y given X, that is Pr(Y|X). The support of the rule is the prior probability of X and Y, that is Pr(X and Y). Distributed association rule mining (DARM) refers to the mining of association rules from distributed data sets. The data sets are stored in local databases hosted by local computers which are connected through a computer network [3]. Typical DARM algorithms involve local data analysis from which a global knowledge can be extracted using knowledge integration techniques [4]. A review of current distributed association rule mining methods was presented by Ogunde et al. [5]. Albashiri [6] gave some key issues to be addressed for distributed data mining tasks dwelling so much on the extendability of the system but the adaptivity has so far not been addressed by researchers.

B. Agents and Multi-Agent Systems (MAS)

Agents are defined by Wooldridge [7] as computer software that are situated in some environment and are capable of autonomous action in this environment in order to meet their design objectives. Agents are active, task-oriented, modeled to perform specific tasks and capable of decision making. By combining multiple agents, in one system, to solve a problem, the resultant system is a MAS. From the literature, well documented advantages of MAS includes: Decentralized control, Robustness, Simple extendability, Sharing of expertise and Sharing of resources [7]. According to Wooldridge [8], the cognitive functions of a rational agent are categorized into the following three modalities. First, beliefs are facts which the agent holds, which represent the properties about the agent's environment. Ideally, the agent's current belief set should be consistent. Second is that desires are the agent's long term goals. There is no requirement that the agent's desires should be consistent. Third modality is that intentions represent a staging post between beliefs and desires, in that they represent goals or sub-goals that the agent intends to actually bring about.

C. Adaptivity in Multi-Agent Systems

Agents typically operate in dynamic environments. Agents come and go, objects appear and disappear, and cultures and conventions change. Whenever an environment of an agent changes to the extent that an agent is unable to cope with (part of) the environment, an agent needs to adapt. Changes in the social environment of an agent, for example, may require modifications to existing agents [9]. The ability to adapt to dynamic environment and unexpected events is a key issue for mobile agents [10]. These inherent changes are dynamic in nature and demands that multi-agent systems should be adaptive and flexible. Therefore, for a multi-agent system, adaptation represents the ability of the multi-agent system to recognize and response to unanticipated internal and external change. Few researches have been done on agents' adaptability but there are none on the adaptivity of DARM agents [9]. Most especially, adaptive agents proposed in this work were based on the foundation laid by Ranjan et al. [10] for adaptive mobile agents.

Lacey and Hexmoor [11] addressed the question of assigning social norms to agents that will eventually act in complex dynamic environments and also treated the possibility of allowing the agents to adapt to new situations as they arise, and choose their norms accordingly. The researchers argued that adaptation is preferable to prescription, in that agents should be allowed to revise their norms when there is a need to adapt to new situations by revising their norms as appropriate. They also argued that their approach is better than prescribing norm adherence at design time. In Lacey and Hexmoor [11], a system was constructed in which the performance of multiple agents operating in the same environment were assessed and experimental results showed that in some circumstances adaptive norm revision strategies performed better than prescriptive norm assignment at design time.

In Tamargo et al. [12], knowledge dynamics in agents' belief based on a collaborative multi-agent system was examined. Four change operators were introduced: expansion, contraction, prioritized revision, and non-prioritized revision. For all of them, both constructive definitions and an axiomatic characterization by representation theorems were given. Minimal change, consistency maintenance, and non-prioritization principles were formally justified by the researchers. Khan and Lespérance [13] and Riemsdijk et al. [14] also contributed in the area of agents' beliefs and goals changing. Khan and Lespérance [13] in their work ensured that the agent's chosen goals/intentions were consistent with each other and with the agent's knowledge. When the environments change, the agents recomputed their chosen goals and some inactive goals may become active again. This ensured that the agent maximized utility. Riemsdijk et al. [14] gave a formal and generic operationalization of goals by defining an abstract goal architecture, which described the adoption, pursuit, and dropping of goals in a generic way.

In our work, we considered that adaptation could be provided at both the agent level and the algorithm level; but, in this paper, emphasis was placed on adaptivity of the mining agents.

III. SYSTEM DESIGN

This design of the proposed adaptive mining agent architecture is presented in this section.

A. The Proposed Adaptive Architecture

The adaptive architecture described in this work was based on the earlier Distributed Association Rule Mining (DARM) architecture AMAARMD presented by Ogunde et al. [15]. The architecture was characterized by a given distributed data mining task being executed in its entirety using the mobile agents. In general, this was expressed as m mobile agents traversing n data sources (where m >n).

B. The Adaptive Algorithm

In our architecture, the very first mining by the system is based on the traditional Apriori algorithm (if the initial

dataset is very small) and the Partition Enhanced Algorithm (PEA), which is an improved version of the state-of-the-art Apriori algorithm contributed by the researchers. PEA partitions the large dataset into smaller partitions, while mining each partition (as it easily fits into the memory) to generate local patterns, which was integrated to generate global frequent patterns for a particular large data site in the DARM architecture. The partition sizes were chosen such that each partition can be accommodated in the main memory, so that the partitions are read only once in each phase. In this work, the mining agent examines the system to obtain the current total available memory space and then use this information to divide DB into the several partitions. This is to ensure that each partition fits into the main memory during the first phase of the mining. Subsequent mining of the incremental database is done with the Adaptive Incremental Mining (AIM) algorithm also contributed by the researchers. Details of *PEA* and *AIM* algorithms were not presented as this particular work is focused on the adaptivity of DARM agents. *AIM* mines only the incremental database dynamically whenever there is a pre-defined increase in the total transactions inside the database. It stores the previously frequent and non-frequent itemsets to be able to determine whether an itemset is still frequent in the updated database or it is no more frequent, taking note of the specific time and periods when these changes occurred for proper management decisions by data miners.

C. Description of the Adaptive DARM Agents

In this section, the different types of agents and users in the architecture are described. Agent types: User Agent (UA), Association Rule Mining Coordinating Agent (ARMCA), Data Source Agent (DSA), Mobile Agent Based Association Rule Miner (MAARM), Mobile Agent- Based Result Reporter (MARR), Results Integration Coordinating Agent (RICA), Task Agent (TA) and Registration Agent (RA). All agents are created and resident in the DARM server. The UA and DSA are interface agents because they all provide "interfaces" to either users, or data sources. UA provides the interface between the architecture, users and the rest of the architecture; while DSA provides the interface between input data and the rest of the architecture. MAARM agents are processing agents because they carry out the required ARM at the data sites automatically or in response to user requests, and possibly, to pre-process data within the system. A description of the various agents in the system described and their interactions are summarized in Figure 1.

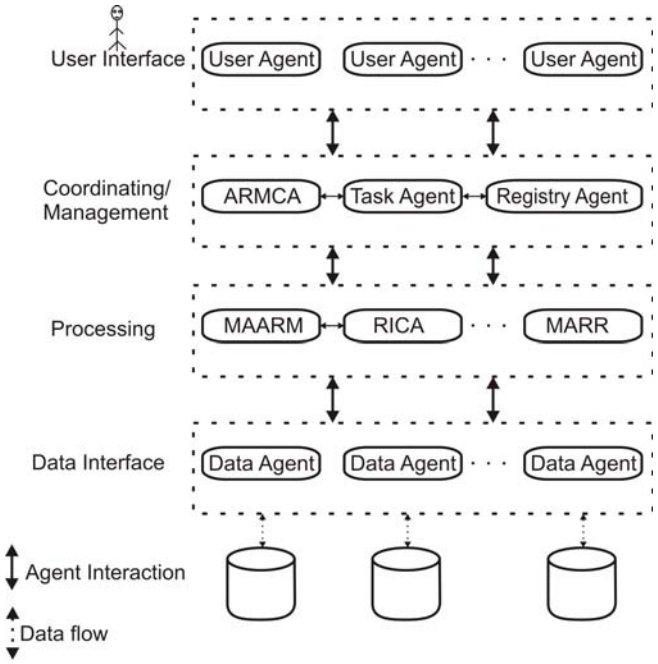

Figure 1: Agent Architecture for the System.

According to Figure 1, the task agent receives a DARM request and asks ARMCA to check all available databases (DA) and MAARM agents to find: (i) which data to use, and (ii) which data mining algorithms held by ARM agents are appropriate. The RA informs all appropriate agents that are already in the system and interested of a new agent arrival. When any new agent is introduced into the system, the TA then passes the DARM task to MAARM agents, which clones itself into multiple copies depending on the number of available data sites, and then travel to each data site in the DARM task. Each data site must have an interface agent – data agent (DA) to check the database for a matching schema and then report back to the MAARM agent. Distributed association rule mining at each data source is performed by the ARM agents – MAARMs. The return of results information at each data site is carried out by the MARRs. The agent RICA integrates the various local results to get a global rule from the DARM sites. Final results or knowledge are passed from the RICA to the TA and then to the UA all through the DARM server.

D. Adaptive Mobile Agent Association Rule Miner (AMAARM)

An agent can be viewed as software satisfying an ordered set of goals to achieve some overall objective. The agent takes a sequence of action in order to satisfy the next goal in the set. Adaptation can be viewed as changing the goal set. The effect of the change can be a new set of actions to achieve the same overall objective as before, or it may even result in a new overall objective if the original objective cannot be achieved anymore in the current environment. The model described here consists of two components: a

 ISBN: 978-1-61208-219-6

Mechanism and an Adapter. The Mechanism is the interface of the AMAARM to the environment. It contains sensors that periodically sense the DARM environment parameters and report their findings to the Mechanism.

For instance, if a particular data server is down out of five data sites or there is a sort of interference to the mining process, the mechanism of the mining agent senses this and reports it to the Adapter, which decides a waiting period for the agent in order to make another attempt to complete the mining process or in the worst case excludes the result of that particular site from the global knowledge integration performed by RICA after a number of preset trial-times. This means that if the set goal for RICA was to integrate local ARM results from five data sites, it will now change the set goal to integrate only the four available results, which are returned as the global knowledge.

It also contains actors that can take actions to change the environment the mining agent is in. 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 percept. The percept is passed on to the adapter, which then decides whether adaptation is necessary or not. Another instance of an unforeseen problem that can arise here are the possibilities of collision of the mining agent with either other mining agents or agents carrying out some other tasks within the same environment. As a matter of fact, all these agents could be possibly competing for the same resources and these could hamper the performance of the association rule mining agent in such environments, hence there is a need for adaption on the fly by these agents. Therefore, 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. Figure 2 shows the basic structure of the AMAARM components and their interactions as explained above.

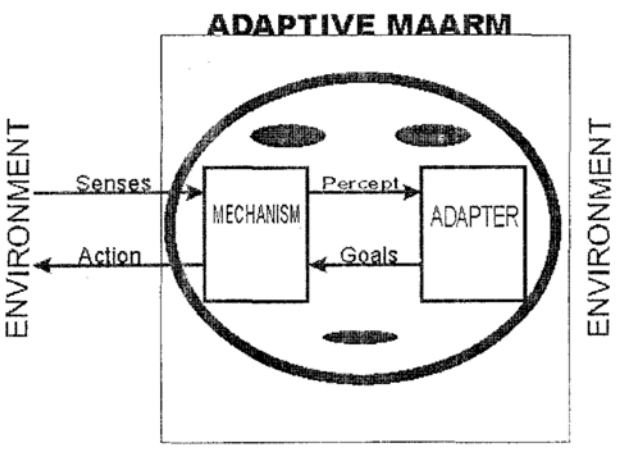

Figure 2: Components of AMAARM

In figure 2, the mechanism actually senses the DARM environment, creates a percept for the adapter, which decides the action, that is, whether initial the goals of the mining agent should be maintained or changed.

E. Description of the AMAARM's Mechanism

The state of the AMAARM's Mechanism is represented by the 3-tuple <S, L, T>, where S represent the behavioral state of the Mechanism which identifies what the Mechanism is doing currently. *S* could therefore be the state where the Mechanism senses the environment for changes or the state at which action is taken to change the agent's goal. *L* is the current location of the agent, and *T* is the time the Mechanism had spent in its current state. The state variable *S* can take one of three possible values: extractGoal, executeCommand and senseEnvironment. In the extractGoal behavioral state, the Mechanism picks up the current mining goals to be executed, and generates the set of commands for it. In the executeCommand behavioral state, the generated commands are carried out. In the senseEnvironment behavioral state, the Mechanism senses the environment and forms a current view of the environment, and then passes it to the Adapter. Figure 3 is a state diagram showing the different states that the AMAARM's Mechanism can be at any point in time and the possible state transitions.

According to Figure 3, the Mechanism can always be in any of these three states: the FirstNormalState = <extractGoal, L, T> or SecondNormalState = <executeCommand, L, T> or SenseEnvironmentState = <senseEnvironment, L, T>, where *L* and *T* retains their predefined definitions as *L* contains the current location of the agent, and *T* is the time the MAARM had been in that state, which is usually reset to 0 every time a state transition occurs. Initially, the mechanism enters the FirstNormalState on receiving an ordered set of goals from the adapter. The commands for the next goal in the ordered list are generated and a transition to state SecondNormalState occurs.

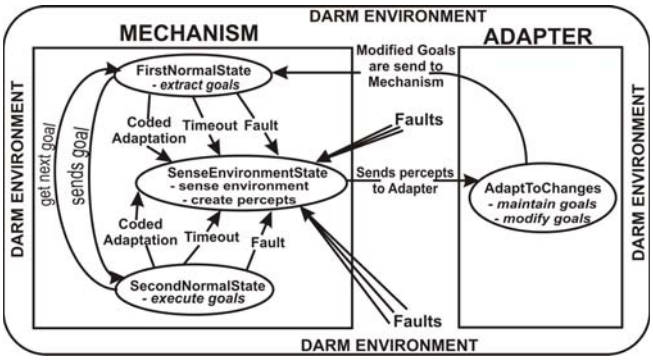

Figure 3: State Diagram of AMAARM

In the SecondNormalState, the commands are executed, and then, the transition goes back to the FirstNormalState in order to generate the commands for the next goal in the list of goals. This process continues until all the goals in the list are executed by the Mechanism. However, the Mechanism may go to the third state, that is, SenseEnvironmentState = <senseEnvironment, L, T> from either the FirstNormalState

or the SecondNormalState if any of the following happens: a timeout, a fault, or a coded adaptation (an explicit command in the application code itself to sense the environment for adaptation reasons), a collision, an attempt to corrupt the mining agent, etc. The *T* component of the Mechanism state detects a timeout if an action is not carried out within a specified time in the SecondNormalState. This may indicate changed environment and may force AMAARM to sense the environment and determine whether adaptation is necessary.

In the SenseEnvironmentState, the environment is usually sensed for any sort of change earlier mentioned and a percept is sent to the Adapter to see if any adaptation is necessary. Normally, the environment can also be sensed in the FirstNormalState and the SecondNormalState, occasionally as the case may be; but, in these cases there is usually no interaction with the Adapter, therefore the values sensed are usually used internally by those states of the AMAARM.

Given a DARM environment, there may be different ways the mining agent can adapt. Thus some type of ranking of the adaptation methods in the adaptation policy is necessary. This is achieved by a motivation degree function. Motivation is any desire or preference that that can lead to the generation and adoption of goals and which affects the outcome of the reasoning or behavioral task intended to satisfy the goal [16]. A motivation degree is therefore associated with each adaptation method, which is the probability of success in achieving the final goal if the set of goals corresponding to the adaptation method is selected as the current set of goals. The Adapter then selects the adaptation method with the highest motivation degree corresponding to the current environment. The set of adaptation methods and the motivation degree function can be hard-coded or learnt dynamically from history or a combination of both where the user specifies an adaptation policy and a motivation degree function, which then can be modified dynamically as well. For the purpose of this work, the adaptation policies for AMAARM are hard-coded.

The adaptive state of the AMAARM is thus described by the 3-tuple <MS, AS, AS>, where MS is the Mechanism state, AS is the Adapting state, and AS is the Application-specific state for the AMAARM. On receiving a percept from the Mechanism, the Adapter goes through the set of adaptation methods, looking for the ones that match the percept. The one with the highest motivation degree is then chosen, and the current set of ARM goals are modified to be the one corresponding to that adaptation method. The new ARM goal set is passed to the Mechanism, which then generates and executes commands for the set of goals. If no adaptation method matches the current environment, adaptation is deemed unnecessary and no change to the goal set occurs. Thus, no adaptation can also be viewed as a special adaptation method.

F. Description of the AMAARM's Adapter

The Adapter state consists of the two tuple <S, T>, where S is the behavioral state of the Adapter, which can only adapt by either maintaining the mining goals, if a change of goal is not necessitated by the percepts received from the mechanism, or modify the mining goals if the percepts received from the mechanism is significant for modifying the original goals. *T* is the time spent in the *AdaptToChange* state. An *attribute* is a perceivable feature of the DARM environment, e.g., a fault in the DARM environment, a time-out, agent collision or an attack or violation of the integrity of the mining agents. A *percept* is a set of attributes, that is, a view of the DARM environment. An *adaptation method* is a single mapping from a percept to a set of mining goals. An *adaptation policy* is a set of adaptation methods. Thus, in this case, the adaptation policy specifies the possible ways in which the AMAARM adapts to different DARM environments.

IV. CONCLUSION AND FUTURE WORK

An adaptive distributed association rule mining architecture with adaptive mining agents was presented. The system described here promises to guarantee the completion of major DARM tasks even in the face of unforeseen circumstances and faults. Each individual data server has some specific data and resource requirements, all of which have to be satisfied before the task can be started. An adaptive mining agent AMAARM executing a task migrates to a data server from the DARM server, 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 about the time the adaptive mining agent may have to wait to perform the mining task. The MAARM relies on the coded adaptation to make this adaptation decision. Future work will consider the set of adaptation methods and policies in DARM that will be a combination of hard-coded policies and also dynamic learning of earlier mining agents' adaptation from history. Implementation of the system using synthetic and real life datasets in order to test the performance of this method will also be done as a future work.

REFERENCES

[1] R. Agrawal, T. lmielinski, and A. Swami, "Mining Association Rules between Sets of Items in Large Databases," In Proceedings of the ACM SIGMOD Conference on Management of Data, Washington, D.C, 1993, pp. 207-216.

[2] V. S. Rao and S. Vidyavathi, "Distributed Data Mining And Mining Multi-Agent Data," (IJCSE) International Journal on Computer Science and Engineering vol. 02, no. 04, 2010, pp. 1237-1244.

[3] M. Z. Ashrafi., D. Taniar, and K. Smith, Monash University "ODAM. An Optimized Distributed Association Rule Mining Algorithm", IEEE distributed systems online, pp. 1541-4922 © 2004 published by the IEEE computer society vol. 5, no. 3; march 2004

[4] H. Kargupta, H. Ilker, and S. Brian, "Scalable, istributed data mining-an agent architecture," In Heckerman et al. [8], pp. 211.

[5] A. O. Ogunde, O. Folorunso, A. S. Sodiya, and G. O. Ogunleye, "A Review of Some Issues and Challenges in Current Agent-Based Distributed Association Rule Mining," Asian Journal of Information Technology, vol. 10, no. 02, 2011, pp. 84-95.

[6] K. A. Albashiri, "EMADS: An Investigation into the Issues of Multi-Agent Data Mining," PhD Thesis, The University of Liverpool, Ashton Building, United Kingdom, 2010. www.csc.liv.ac.uk/research/techreports/tr2010/ulcs-10-004.pdf [retrieved: June, 2012]

[7] M. Wooldridge, “An Introduction to Multi-Agent Systems,” John Wiley and Sons (Chichester. United Kingdom), 2009.

[8] M. Wooldridge, “Reasoning About Rational Agents,” Cambridge, MA: MIT Press, 2000.

[9] F. M. T. Brazier., and N. J. E. Wijngaards, “Automated servicing of agents,” AISB Journal, vol.1, no.1, pp. 5-20, 2001.

[10] S. Ranjan, A. Gupta, A. Basu, A. Meka, and A.Chaturvedi, “Adaptive mobile agents: Modeling and a case study”. 2nd Workshop on Distributed Computing “IEEE md CFP : WDC’2000’.

[11] N. Lacey and H.Hexmoor, “Norm Adaptation and Revision in a Multi-Agent System”, American Association of Artificial Intelligence, FLAIRS 2003, pp. 27-31.

[12] L. H. Tamargo, A. J. Garcia, M. A. Falappa, and G. R. Simari, “Modeling knowledge dynamics in multi-agent systems based on informants,” The Knowledge Engineering Review, Cambridge University Press, DOI: 10.1017/S000000000000000, Printed in the United Kingdom, vol. 00:0, pp. 1-31, 2010.

[13] S. M. Khan and Y. Lespérance, “A Logical Framework for Prioritized Goal Change,” Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010), May, 10–14, 2010, Toronto, Canada, pp. 283-290.

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

[15] A. O. Ogunde, O. Folorunso, A. S. Sodiya, and J. A. Oguntuase, “Towards an adaptive multi-agent architecture for association rule mining in distributed databases,” Adaptive Science and Technology (ICAST), 2011 3rd IEEE International Conference on 24-26 Nov. 2011, pp. 31 – 36 from IEEE Xplore.

[16] Z. Kunda, “The case for motivated reasoning,” Psychological Bulletin, 108 (3), pp. 480-498, 1990.

 ISBN: 978-1-61208-219-6