A Customizable Multi-Agent System for Distributed Data Mining*

Giuseppe Di Fatta
School of Systems Engineering, University of Reading
Whiteknights, Reading RG6 6AY, U.K.
G.DiFatta@reading.ac.uk

Giancarlo Fortino
DEIS – Università della Calabria
Via P. Bucci cubo 41c, 87036 Rende (CS), Italy
g.fortino@unical.it

ABSTRACT
We present a general Multi-Agent System framework for distributed data mining based on a Peer-to-Peer model. Agent protocols are implemented through message-based asynchronous communication. The framework adopts a dynamic load balancing policy that is particularly suitable for irregular search algorithms. A modular design allows a separation of the general-purpose system protocols and software components from the specific data mining algorithm. The experimental evaluation has been carried out on a parallel frequent subgraph mining algorithm, which has shown good scalability performances.

Keywords
Multi-Agent Systems, Distributed Data Mining, Agent Oriented Software Engineering, Dynamic Load Balancing, Peer-To-Peer Computing.

1. INTRODUCTION
The last decade has seen an ever increasing availability of large amounts of data in many fields of science and in many IT applications. Data mining techniques have become popular techniques, which can reveal the valuable knowledge hidden in the rough data. However, these techniques often require high performance approaches in order to cope with the overwhelming amount of data and the complexity of algorithms. An effective approach is based on distributed and parallel computing. Clusters of Workstations, Grid computing infrastructures, massively parallel systems and multi-core technology make distributed and parallel data mining a very appealing solution in many application fields.

Many tools for data mining are available today (e.g. [14, 6, 12, 2]). They typically provide a large selection of algorithms for Association Rule Mining, Clustering, Classification, etc. However, none of them supports parallel and distributed approaches.

Distributed approaches can typically be adopted for two main reasons. First, data are often intrinsically distributed over several sites. Collecting them in a single processing node cannot be convenient for the communication costs, feasible for privacy policies, even useful for real time data streams. Second, the complexity of the task is often such that we need to partition the data to be able to process each data subset independently and finally combine the partial results.

Several parallel data mining algorithms have been proposed in the literature, e.g. for Association Rule Mining [18]. However, few of them have paid attention to scalability and heterogeneity of the computational infrastructure and have adopted dynamic load balancing policies.

In this context, we believe that a general Distributed Data Mining (DDM) framework can enable and accelerate the deployment of practical solutions.

Among the emergent paradigms for distributed computing, the Agent paradigm has demonstrated to be particularly suitable for supporting the construction of flexible and effective frameworks for distributed computation [11, 8]. According to the agent paradigm a distributed computation framework is designed in terms of a Multi-Agent System (MAS), i.e. a system composed of several agents, capable of reaching goals that are difficult to achieve by an individual system [15]. In addition, MASs can manifest self-organization and complex behaviours, even when the individual strategies of all their agents are simple.

Several efforts have been devoted to enable DDM through MASs. In [16] the authors present a MAS for context-based distributed data mining. The proposed MAS architecture is client/server and is basically composed of a Miner Agent at the server side which distributes mining tasks to Local Agents which, after task completion, send the results back to the Miner Agent. Due to this simple scheme, the Miner Agent represents a performance and scalability bottleneck. In [10] the authors review four well established agent-based DDM systems (BODHI, PADMA, JAM, Papyrus). The first three systems are based on a centralized architecture in which a facilitator (or coordinator) agent interacts with the mining agents in a way similar to the one proposed in [16]. Papyrus is conversely Peer-to-Peer (P2P) oriented.

In this paper we propose a customizable MAS for general-purpose distributed data mining which exploits P2P concepts to improve performance and scalability. Our system is designed by using the GAIA methodology [20] and implemented through a Java-based agent framework [8, 9]. In particular, the MAS architecture is organized as a flat P2P network of nodes, each of which is a MAS. Such an

(*) To appear in the Proceedings of ACM SAC, Technical Track on Agents, Interactions, Mobility, and Systems (AIMS), March 11-15, 2007, Seoul, Korea [in press].

organization supports an efficient dynamic load balancing particularly suitable for irregular search algorithm. Currently the MAS is customized for the distributed mining of molecular structures.

The rest of the paper is organized as follows. The next section introduces the architecture of the Distributed Data Mining framework based on a Multi-Agent System. Section 3 presents a test case, a frequent subgraph mining application, which has been implemented within the proposed framework. Section 4 presents some experimental results of the test case. Finally, section 5 provides conclusive remarks.

2. P2P-BASED ARCHITECTURE OF THE MULTI-AGENT SYSTEM

The overall MAS architecture is organized as a flat P2P network of nodes, each of which is a MAS. Two kinds of nodes have been defined: (i) the *processing node*, which executes the mining algorithm on local data and interacts with the other processing nodes to exchange jobs, data and partial results; (ii) the *coordinating node*, a particular processing node, which serves also as P2P network bootstrapper, coordinator of the processing nodes, collector of incoming statistics and application GUI.

In the following sections the MAS architecture and interaction protocols are analyzed and designed through the GAIA methodology [20]. GAIA can be thought of as a two-phase process of incremental development of detailed models of the system to be constructed. In the analysis phase GAIA produces the following work products: the Prototypical Roles Model, the Interactions Model and the Roles Model. In the design phase the analysis models are transformed into a design which is concerned with the cooperation of agents in a society and the services offered by each individual agent to achieve the system-level goals. The work products of this phase are the Agent Model, the Services Model, the Acquaintance Model and the Organization Model.

2.1 GAIA-based Design

In the *Prototypical Roles Model* the following key roles are identified:

- *Control Manager* (CM), which manages a processing node by allowing a processing node to join and synchronize with the P2P network;
- *Data Manager* (DM), which allows to manage the data local to each node and to distribute data on-demand to remote nodes;
- *Job Manager* (JM), which allows to manage the jobs intra- ad inter-node;
- *Worker* (W), which performs jobs according to a specific mining algorithm;
- *Coordinator* (C), which coordinates the P2P network formation, start-up, and computation.
- *Statistics Collector* (SC), which collects statistics about the execution of the distributed data mining algorithm.

The *Interactions Model* defines the interactions among the roles of the *Prototypical Roles Model*. An interaction is defined through the following tuple: *<InteractionName, InitiatorRole, ResponderRole, CommunicativeAct>*, where *InteractionName* is the name of the interaction, *InitiatorRole* is the role starting the interaction, *ResponderRole* is the role responding to the interaction, *CommunicativeAct* is the action performed by means of the interaction. In the following the defined interactions, which are of the control, job, data and statistics types, are described:

- <JoinRequest, CM, C, Service Request>. A CM requests C to join the P2P network;
- <Configuration, C, CM, Service Notification>. C replies to CM with the configuration file;
- <ExitNotification, CM, C, Operation Notification>. CM notifies C to exit from the P2P network;
- <ErrorNotification, CM, C, Operation Notification>. CM notifies C that an error is occurred;
- <BarrierRequest, CM, C, Service Request>. CM request to synchronize with C;
- <BarrierReleased, C, CM, Service Results>. C releases CM that means the processing node controlled by CM starts processing;
- <StartActivity, CM, (DM, JM, W), Operation Notification>. CM enables the local JM, DM and W;
- <KeepAlive, CM, C, State Notification>. CM sends a keep alive message to refresh the system status of C;
- <ReductionStart, C, W, Operation Notification>. C starts the reduction phase, i.e. the phase of assembling the partial results;
- <JobRequest, JM, JM, Task Request>. A JM requests a job to a remote JM; the selection criterion of the remote JM is not specified at this level;
- <Job, JM, JM, Task Assignment>. A JM assigns a job to a requesting JM;
- <JobSpawning, JM, W, Task Execution Request>. A JM assign a job to its local W;
- <JobResult, W, W, Result Notification>. A W sends the results of the execution of a job to another W to which is logically linked through a reduction relationship;
- <GetLocalData, W, DM, Data Request>. W asks for data to its local DM;
- <LocalData, DM, W, Data Assignment>. DM assigns the requested data to its local W;
- <GetRemotelData, DM, DM, Data Assignment>. DM asks for data to a remote DM;
- <RemoteData, DM, DM, Data Assignment>. DM sends the requested data to a remote DM;
- <JobCompletion, W, SC, Operation Notification>. A W sends SC the notification of job completion;
- <JobSplitting, W, SC, Operation Notification>. A W sends SC the notification of job splitting;
- <JobStats, SC, C, Data Notification>. SC sends statistics data about splitting and completion of jobs;
- <JobStatsRequest, C, SC, Data Request>. C requests SC statistics data about splitting and completion of jobs.

The *Roles Model* documents the Roles occurring in the system, their Responsibilities and Permissions, and the Protocols and Activities in which they participate. Responsibilities are the key properties associated with a role and are divided into two types: (i) Liveness Properties which describe those states of affairs that an agent must bring about, given certain environmental conditions; (ii) Safety Properties which are invariants and state that an acceptable state of affairs is maintained throughout all the states of execution. Permissions are the rights associated with a role and identify the resources that are available to that role in order to fulfill its responsibilities. Protocols define how a role can interact with other roles. Finally, Activities of a role are computations associated with the role that may be carried out by the agent without interacting with other agents. From a diagrammatic point of view, the *Roles Model* therefore consists of a set of Role Schemas. A Role Schema draws together the Protocols, Activities, Permissions and Responsibilities of a Role. Due to limited space, only the Role Schema of the CM is reported in Figure 1. The Role Schema presents the Protocols in which a CM is involved (NodeJoin, NodeBarrier, NodeExit, NodeError, StartActivity), and the following Permission with which it is provided: reading the configurationData contained in the configuration file sent from C.

Role Schema: CONTROLMANAGER
Description: *This role involves the control of a peer node*
Protocols and Activities: NodeJoin, NodeBarrier, NodeExit, NodeError, StartActivity
Permissions: reads supplied *configurationData*
Responsibilities Liveness: CONTROLMANAGER=NodeJoin · NodeBarrier · StartActivity · NodeExit Safety: - (*joined* = true) ∧ (*configurationData* not empty);

Figure 1. Role Schema of the Control Manager.

Regarding to the CM Responsibilities:

- the Liveness Properties are expressed by a Liveness Expression, formalized using the Fusion notation [20], which defines the "life-cycle" of the role. The Liveness Expression in Figure 1 indicates that a CM sequentially executes the protocols: NodeJoin, NodeBarrier, StartActivity, NodeExit.
- the Safety Properties ensure that a CM joined the P2P network and the configurationData are not empty.

The *Agent Model* (see Fig. 2) was obtained by aggregating the identified Roles into Agent Types. An Agent Type is represented by a tree, in which the leaf nodes correspond to roles and the other nodes correspond to agent types. The Agent Types identified were: Control Manager Agent (CMA), Job Manager Agent (JMA), Data Manager Agent (DMA), Worker Agent (WA), and Coordinator Agent (CA). The symbol "+" in the figure indicates multiple instantiation of an Agent Type.

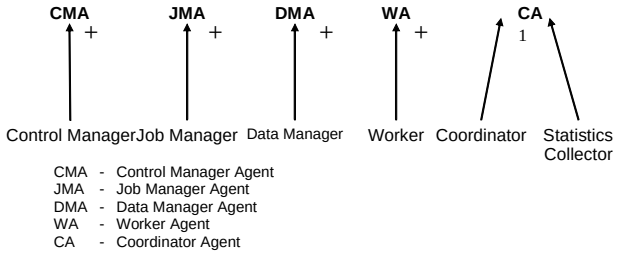

Figure 2. The Agent Model.

Each service associated with a role was documented in the *Services Model* by describing its inputs, outputs, pre-conditions, and post-conditions. The services provided were derived from the list of Protocols, Activities, Liveness and Safety Properties of the identified roles. For example, the Services associated with the Coordinator Role are shown in Table 1.

Table 1. The Services of the Coordinator Role

C ROLE				
SERVICE	**INPUTS**	**OUTPUTS**	**PRE-CONDITION**	**POST-CONDITION**
GroupFormation	*NodeID*	*DataConfiguration*	Started=false	Started=false
GroupSynchroStart		*StartSignal*	Started=false	Started=true
ReductionStart		*ReductionSignal*	Reducted=false	Reducted=true

The *Acquaintance Model* (see Fig. 3) derived from the *Roles*, *Interactions* and *Agent Model* is a directed graph, with nodes corresponding to Agent Types and arcs representing communication pathways.

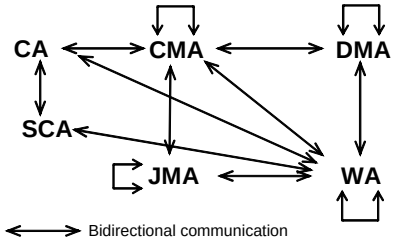

Figure 3. The Acquaintance Model.

Finally, the *Organizational Model* defines the structure of the overall MAS. According to a flat organization, the entire MAS is composed of a set of local MASs or processing nodes (or PNodes) and one coordinating/processing node (or BSNode). The organization structure is defined through the UML class diagram shown in Figure 4. It is worth noting that other kinds of organizations (e.g., hierarchical structure) can be easily defined.

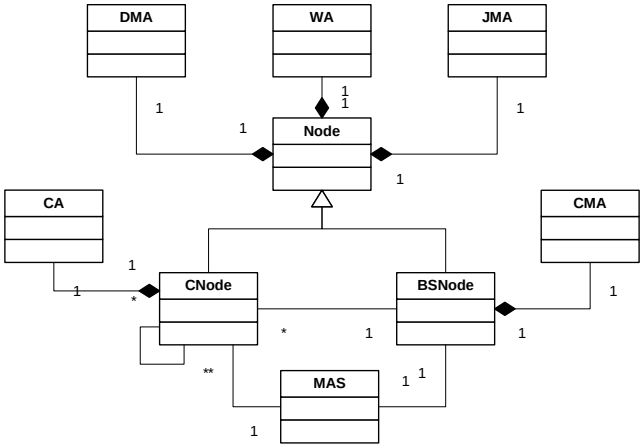

Figure 4. The Organization Model.

2.2 MAS Dynamics: Formation and Execution

The phases through which the MAS evolves are three:

- *Bootstrap* phase;
- *Computing* phase;
- *Reduction* phase.

In the *Bootstrap* phase the BSNode waits for the join requests from the PNodes. Once a join request is accepted a PNode receives a ConfigurationFile and, soon after, sends the BarrierRequest message to synchronize its activity start with respect to the other PNodes. The BarrierReleased messages are sent from the BSNode as soon as a condition at the BSNode is met according to a selectable policy, e.g. if the number of joining PNodes is equals to a given number.

In the *Computing* phase the BSNode creates the very first job from which the other jobs originate. The spawning of a new job is dynamically generated by the WA, according to its internal specific Miner component, when a job request is issued according to a receiver-driven policy, i.e. the WA asks for a new job when it is idle.

The Reduction phase, i.e. the phase in which the partial results are collected and merged, is triggered by the CA and involves all the WAs. At the end of this phase, the final result is kept into the BSNode and the PNodes can leave the P2P network.

3. A CASE STUDY: DISTRIBUTED MINING OF MOLECULAR STRUCTURES

A crucial step in the drug discovery process is the so-called High Throughput Screening and the subsequent analysis of the generated data. During this process, hundreds of thousands of potential drug candidates are automatically tested for a desired activity, such as blocking a specific binding site or attachment to a particular protein. This activity is believed to be connected to, for example, the inhibition of a specific disease. Once all these compounds have been screened automatically, it is necessary to select a small subset of promising candidates for further, more careful and cost-intensive analysis. A promising approach focuses on the analysis of the molecular structure and the extraction of relevant molecular fragments that may be correlated with activity. Such fragments can be used to directly identify groups of promising molecules (clustering). They can also be used to predict activity in other compounds (classification) and to guide the synthesis of new ones.

However, the computational complexity of the underlying problem and the large amount of data to be explored require the adoption of parallel and distributed computational environments [4, 5].

We have applied the general Agent-based Distributed Data Mining framework described in the previous section to a high performance distributed Frequent Subgraph Mining (FSM) approach to the molecular fragments discovery problem.

The distributed approach is based on three main components, namely a sequential algorithm [3], a dynamic partitioning of the search space [4], and a task distribution process based on a Peer-to-Peer (P2P) communication framework [5] embedded in the developed MAS.

3.1 The Molecular Fragment Miner

The problem of selecting molecular fragments in a set of molecules can be formulated in terms of FSM in a set of graphs, in analogy with the Association Rule Mining (ARM) problem [1, 19]. While in ARM the main structure of the data is a list of items and the basic operation is the subset test, FSM relies on graph and subgraph isomorphism.

Molecules are represented by attributed graphs, in which each vertex represents an atom and each edge a bond between atoms. Each vertex carries attributes that indicate the atom type (i.e., the chemical element), a possible charge, and whether it is part of an aromatic ring. Each edge carries an attribute that indicates the bond type (single, double, triple, or aromatic). Frequent molecular fragments are subgraphs that have a certain minimum support in a given set of graphs, i.e., are part of at least a certain percentage of the molecules. We assume that the molecular compounds in the dataset can be classified in two groups. We refer to the two classes of molecules as the focus set (active molecules) and its complement (non-active molecules).

The aim of the data mining process is to provide a list of molecular fragments that are frequent in the focus dataset and infrequent in the complement dataset. These topological fragments carry important information and may be representative of those components in the compounds that are responsible for a positive behavior. In this case, two parameters are required: a minimum support for the focus subset and a maximum support for the complement.

Existing methods attempt to implicitly organize the space of all possible subgraphs in a lattice, which models subgraph relationships. The search then reduces to traversing this lattice and reporting all subgraphs that fulfill the desired criteria. These methods conduct depth-first [3, 17] or breadth-first searches [7]. The parallel algorithm we have adopted is based on a sequential algorithm [3], which is based on a Depth First Search (DFS) strategy.

In [5] the analysis of the sequential algorithm has pointed out the highly irregular nature of this problem. An irregular problem is characterized by a highly dynamic or unpredictable domain. In this application, the complexity and the exploration time of the search tree and even of a single search tree node cannot be estimated nor bounded. The empirical probability distribution of the task complexity is characterized by a power-law distribution in a wide range. It is quite challenging to efficiently parallelize problems with such an unpredictable workload and, moreover, to provide scalability in terms of computing resources. The proposed MAS architecture in conjunction with the adopted Dynamic Load Balancing (DLB) policy is able to meet these goals, as shown in the experimental results.

3.2 Dynamic search space partitioning

A task partitioning policy has to provide a mechanism to fairly distribute the load among the processors using a small number of generated subtasks to reduce the communication cost and the computational overhead. In particular, in applications based on irregular search-trees, like the molecular fragment miner, the task partitioning policy has to carefully select (1) a suitable subtask donor among all the workers and (2) a non-trivial subtask among the search nodes in the local stack of the donor. The quality of both the selection of donors and the generation of new subtasks is fundamental for an effective and efficient computational load distribution. These two tasks are carried out, respectively, by a DLB algorithm and a Work Splitting mechanism.

Several DLB policies are provided in the proposed DDM system, such as Scheduler-based, Random Polling, Global Random Polling, Ranked-Random Polling [5]. The system can be easily extended with others.

On the contrary, the Work Splitting mechanism depends on the particular data mining applications and has to be embedded in the specific algorithm. This is the only effort required to the programmer in order to adopt a sequential algorithm in the DDM environment. In applications based on a search tree, the search nodes are typically stored in a stack and the work splitting mechanism corresponds to the selection of one or more elements of the stack. In this case, search nodes are often selected from the bottom of the stack in order to donate non trivial tasks. However, as already said, this policy intrinsically depends on the particular application and general rules cannot be provided.

An example of a search tree is depicted in figure 5, which also shows the partitioning strategy.

Each worker agent maintains only a local and partial list of results found during the execution of subtasks. Therefore, at the end of the search process, we perform a reduction operation. To this aim, workers are organized in a communication tree and the number of communication steps required is in the order of O(log N), where N is the number of agents.

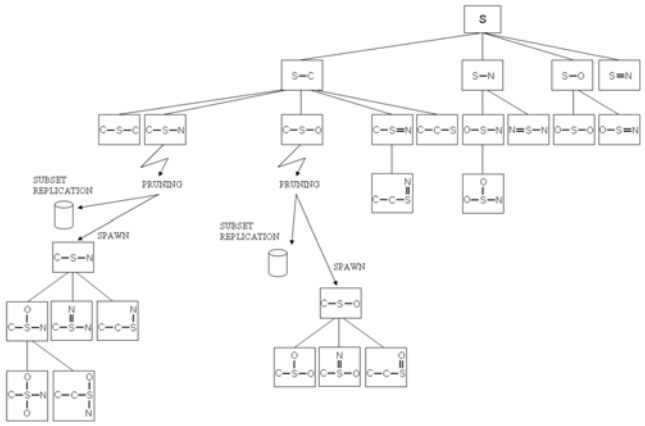

Figure 5. Search tree partitioning.

The proposed DDM environment provides the general framework to perform the partial result reduction step. However, as for the partitioning policy, the specific reduction operation depends on the specific application. For example, in molecular fragment mining we need to merge partial lists of fragments and discard duplicates.

4. Experimental results

The MAS has been tested for the analysis of a set of real molecular compounds – a well-known, publicly available dataset from the National Cancer Institute, the DTP AIDS Antiviral Screen dataset [13]. Experimental tests have been carried out on a network of workstations with different hardware and software configurations. The group of the eight highest performing machines is equipped with a CPU Intel Xeon 2.40GHz, 3GB RAM and run Linux 2.6.5-7.151 and Java SE 1.4.2_06. The software has been developed by using a Java-based agent framework [8, 9]; the communication among agents has been implemented through asynchronous messages and XML data format.

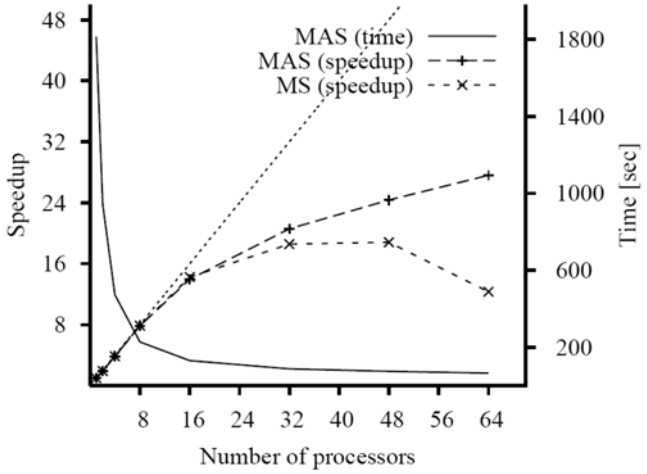

Figure 6. Running time and speedup.

Figure 6 shows a comparison of the speedup between the proposed MAS-based approach and a Master-Slave (MS) one. Both adopt similar heuristics in the donor selection mechanism of the DLB policy. However, the MAS-based system adopts distributed job pools and the MS system a centralized job pool.

In the first part of the chart the speedup is linear for both approaches and the running time decreases accordingly. For larger number of processors (greater than 32) the MS approach reveals the limitation of a centralized approach. The master node has become a bottleneck. On the contrary, the MAS-based system clearly shows good efficiency in the entire range.

For completeness, the running time for the MAS-based solution is also shown in Figure 6. More resources cannot further decrease the running time at the same rate because the amount of work is not significant enough and additional computational resources cannot be effectively exploited. Nevertheless, the running time of the MAS-based system does not increase when unnecessary resources are used as one might expect because of the additional communication and computation overheads. This provides evidence of the good scalability properties of the system.

5. CONCLUSIONS

In this paper we have presented a high performance distributed computing approach based on a multi-agent system. The proposed multi-agent system provides a general framework for distributed data mining applications. The effort to embed the logic of a specific domain has been minimized and is limited to the customization of the Worker Agent behaviour. The user has to provide application-specific methods for the execution of a sequential task, partition of a task in two subtasks, reduction of two partial results.

The proposed MAS has been applied to the frequent subgraph mining problem for the discovery of discriminative molecular fragments. Experimental tests on real molecular compounds confirmed its effectiveness. Very low communication and synchronization requirements, dynamic load balancing and high scalability of the communication model make the proposed agent-based distributed computing framework suitable for large-scale, non-dedicated, heterogeneous computational environments like Grids.

Future research efforts will focus on a more accurate formalization of the model and its simulation on very large-scale systems. Other research directions include the introduction of advanced and intelligent services based on the MAS potentiality and the adoption of the approach in other application domains to verify and extend its general applicability.

6. REFERENCES

[1] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, pages 207–216, May 26–28, 1993.

[2] M. R. Berthold, N. Cebron, F. Dill, G. Di Fatta, T. Gabriel, F. Georg, T. Meinl, P. Ohl, C. Sieb, B. Wiswedel. Knime: the Konstanz Information Miner. Proc. of the 4th Annual Industrial Simulation Conference (ISC), Workshop on Multi-Agent Systems and Simulation (MAS&S), Palermo, Italy, June 5-7, 2006.

[3] C. Borgelt and M. R. Berthold. Mining molecular fragments: Finding relevant substructures of molecules. IEEE International Conference on Data Mining (ICDM 2002), pages 51–58, Dec. 9–12, 2002.

[4] G. Di Fatta and M. R. Berthold. Distributed mining of molecular fragments. Proc. of the Workshop on Data Mining and the Grid (DM-Grid) of the IEEE International Conference on Data Mining (ICDM-04), Nov. 1–4, 2004.

[5] G. Di Fatta and M. R. Berthold. Dynamic Load Balancing in Distributed Mining of Molecular Compounds. IEEE Transactions on Parallel and Distributed Systems, Special Issue on High Performance Computational Biology, 17(8), August 2006, pages 773-785.

[6] Demsar J., Zupan B., Leban G. Orange: From Experimental Machine Learning to Interactive Data Mining. White Paper, Faculty of Computer and Information Science, University of Ljubljana, 2004.

[7] M. Deshpande, M. Kuramochi, and G. Karypis. Automated approaches for classifying structures. Proceedings of Workshop on Data Mining in Bioinformatics (BioKDD), pages 11–18, 2002.

[8] G. Fortino, F. Frattolillo, W. Russo, E. Zimeo. Mobile Active Objects for highly dynamic distributed computing. Proceedings of IEEE International Parallel and Distributed Processing Symposium (IPDPS), Workshop on Java for Parallel and Distributed Computing (JPDC'02), Fort Lauderdale (Florida), USA, pages 1-8, Apr. 15-19, 2002.

[9] G. Fortino, W. Russo. Multi-coordination of Mobile Agents: a Model and a Component-based Architecture. Proceedings of the 20th Annual ACM Symposium on Applied Computing (SAC'05), Special Track on Coordination Models, Languages and Applications, Santa Fe, New Mexico, USA, vol. 1, pages 443-450, Mar. 13-17, 2005.

[10] M. Klusch, S. Lodi, G. Moro. Agent-based Distributed Data Mining: The KDEC Scheme. Intelligent Information Agents - The AgentLink Perspective. Lecture Notes in Computer Science 2586 Springer 2003.

[11] Luck, M., McBurney, P., and Preist, C. Agent technology: enabling next generation computing: A roadmap for agent-based computing. AgentLink Report, 2003. Available from www.agentlink.org/roadmap.

[12] Mierswa, Ingo and Wurst, Michael and Klinkenberg, Ralf and Scholz, Martin and Euler, Timm. YALE: Rapid Prototyping for Complex Data Mining Tasks. Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-06), 2006.

[13] Weislow, O., Kiser, R., Fine, D., Bader, J., Shoemaker, R., Boyd, M. New soluble formazan assay for hiv-1 cytopathic effects: Application to high flux screening of synthetic and natural products for aids antiviral activity. Journal of the National Cancer Institute, University Press, Oxford, UK, 81 (1989), pages 577–586.

[14] Ian H. Witten and Eibe Frank. Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, June 2005.

[15] M. Wooldridge. An Introduction to Multi Agent Systems. John Wiley & Sons Ltd. 2002.

[16] Y. Xing, M.G. Madden, J. Duggan, G. Lyons. A Multi-Agent System for Context-based Distributed Data Mining. Technical Report Number NUIG-IT-170503, Department of Information Technology, NUI, Galway, 2003.

[17] X. Yan and J. Han. GSpan: Graph-based substructure pattern mining. Proceedings of the IEEE International Conference on Data Mining (ICDM'02), 2002.

[18] M.J. Zaki. Parallel and Distributed Association Mining: A Survey. IEEE Concurrency, vol. 7, no. 4, pages 14-25, 1999.

[19] M. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. Proceedings of 3rd Int. Conf. on Knowledge Discovery and Data Mining (KDD'97), pages 283–296, 1997.

[20] F. Zambonelli, N. R. Jennings, M. J. Wooldridge. Developing Multiagent Systems: the Gaia Methodology. ACM Transactions on Software Engineering and Methodology, Vol. 12, No. 3, July 2003.