International Journal of Science and Engineering Applications
Volume 2 Issue 5, 2013, ISSN-2319-7560 (Online)

Agent-Driven Distributed Data Mining

Rohini . P
Department of CSE
Malla Reddy College of Engineering
Andhra Pradesh, India.

Sree Lakshmi.P
Department of CSE
Malla Reddy College of Engineering
Andhra Pradesh, India.

Abstract: Multi-Agent systems (Autonomous agents or agents) and knowledge discovery (or data mining) are two active areas in information technology. A profound insight of bringing these two communities together has unveiled a tremendous potential for new opportunities and wider applications through the synergy of agents and data mining. Multi-agent systems (MAS) often deal with complex applications that require distributed problem solving. In many applications the individual and collective behavior of the agents depends on the observed data from distributed data sources. Data mining technology has emerged, for identifying patterns and trends from large quantities of data. The increasing demand to scale up to massive data sets inherently distributed over a network with limited band width and computational resources available motivated the development of distributed data mining (DDM).Distributed data mining is originated from the need of mining over decentralized data sources. DDM is expected to perform partial analysis of data at individual sites and then to send the outcome as partial result to other sites where it sometimes required to be aggregated to the global result.

Keywords: Distributed Data Mining, Multi-Agent Systems, Multi Agent Data Mining, Multi-Agent Based Distributed Data Mining.

1. INTRODUCTION

Data Mining (DM), originated from knowledge discovery from databases (KDD), the large variety of DM techniques which have been developed over the past decade includes methods for pattern-based similarity search, cluster analysis, decision-tree based classification, generalization taking the data cube or attribute-oriented induction approach, and mining of association rules [7]. DDM is a branch of the field of data mining that offers a framework to distributed data paying careful attention to the distributed data and computing resources. Distributed data mining (DDM) mines data from data sources regardless of their physical locations. The need for such characteristic arises from the fact that data produced locally at each site may not often be transferred across the network due to the Excessive amount of data and security issues. Recently, DDM has become a critical component of knowledge based systems because its decentralized architecture reaches every network such as weather databases, financial data portals, or emerging disease information systems has been recognized by industrial companies as an opportunity of major revenues from applications such as warehousing, process control, and customer services, where large amounts of data are stored.

In the DDM literature, one of two assumptions is commonly adopted as to how data is distributed across sites: homogeneously (horizontally partitioned) and heterogeneously (vertically partitioned). Both viewpoints adopt the conceptual viewpoint that the data tables at each site are partitions of a single global table.

Data Mining still poses many challenges to the research community. The main challenges in data mining are: 1) Data mining has to deal with huge amounts of data located at different physical locations. 2) Data mining is computationally intensive process involving very large data i.e. more than terabytes. So, it is necessary to partition and distribute the data for parallel processing to achieve acceptable time and space performance. 3) The data stored for particular domain the Input data changes rapidly. In these cases, knowledge has to be mined fast and efficiently in order to be usable and updated.

2. NEED OF MULTI-AGENTS

Autonomous agents are computational systems that inhibit some complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are designed. Agents are reactive i.e., they perceive their environment and respond in a timely fashion to changes that occur.

Multi-Agent systems are used for all types of system composed of multiple autonomous components showing the following characteristics:

- each agent has incomplete capabilities to solve a problem
- there is no global system control
- data is decentralized
- computation is asynchronous

Multi-Agent has the following features

- Dividing functionality among many agents provides modularity, flexibility, modifiability, and extensibility.
- Knowledge that is spread over various sources

(agents) can be integrated for a more complete view when needed.
- Applications requiring distributed computing are better supported by MAS.
- Agent technology supports distributed component technology.

In a typical distributed environment analyzing distributed data is a non-trivial problem because of many constraints such as limited bandwidth (e.g. wireless networks), privacy sensitive data, distributed compute nodes, only to mention a few. The field of Distributed Data Mining (DDM) deals with these challenges in analyzing distributed data and offers many algorithmic solutions to perform different data analysis and mining operations in a fundamentally distributed manner that pays careful attention to the resource constraints.

Since MAS are also distributed systems, combining DDM with MAS for data intensive applications is appealing. DDM is expected to perform partial analysis of data at individual sites and then to send the outcome as partial result to other sites where it is sometimes required to be aggregated to the global result. Quite a number of DDM solutions are available using various techniques such as distributed association rules, distributed clustering, Bayesian learning, classification (regression), and compression, but only a few of them make use of intelligent agents at all. The main problems any approach to DDM is challenged issues of autonomy and privacy. For example, when data can be viewed at the data warehouse from many different perspectives and at different levels of abstraction, it may threaten the goal of protecting individual data and guarding against invasion of privacy. These issues of privacy and autonomy become particularly important in business application scenarios where, for example, different (often competing) companies may want to collaborate for fraud detection but without sharing their individual customers' data or disclosing it to third parties. DDM is a complex system focusing on the distribution of data resources over the network as well as extraction of data from those resources. The very core of DDM systems is the scalability as the system configuration may be altered time to time, therefore designing DDM systems deals with great details of software engineer issues, such reusability, extensibility, compatibility, flexibility and robustness. For these reasons, agents' characteristics are desirable for DDM systems.

Autonomy of the system: A DM agent here is considered as a modular extension of a data management system to deliberatively handle the access to the data source in agreement with constraints on the required autonomy of the system, data and model. This is in full compliance with the paradigm of cooperative information systems [6].

Multi-strategy DDM: For some complex application settings an appropriate combination of multiple data mining techniques may be more beneficial than applying just one particular one. DM agents may choose depending on the type of data retrieved from different sites and mining tasks to be done. The learning of multi-strategy selection of DM methods is similar to the adaptive selection of coordination strategies in a multi-agent system as proposed.

Collaborative DM: DM agents may operate independently on data they have gathered at local repositories, and then combine their respective patterns or they may agree to share potential knowledge as it is discovered.

Scalability of DM to massive distributed data: To reduce network and DM application server load, DM agents migrate to each of the local data sites in a DDM system on which they may perform mining tasks locally, and then either return with or send relevant pre-selected patterns to their originating server for further processing. Experiments in using mobile information filtering agents in distributed data environments are encouraging [9].

Security: Any agent-based DDM system has to cope with the problem of ensuring data security and privacy. Any mining operation performed by agents of a DDM system lacking sound security architecture could be subject to eavesdropping, data tampering, or denial of service attacks. Agent code and data integrity is a crucial issue in secure DDM: Subverting or hijacking a DM agent places a trusted piece of (mobile) software. In addition, data integration or aggregation in a DDM process introduces concern regarding inference attacks as a potential security threat. However, any failure to implement least privilege at a data source, that means endowing subjects with only enough permissions to discharge their duties, could give any mining agent unsolicited access to sensitive data. Finally, selective agent replications may help to prevent malicious hosts from simply blocking or destroying the temporarily residing DM agents.

Trustworthiness: Data mining agents may infer sensitive information even from partial integration to a certain extent and with some probability. This problem, known as the so called inference problem, occurs especially in settings where agents may access data sources across trust boundaries which enable them to integrate implicit knowledge from different sources using commonly held rules of thumb.

Furthermore, the decentralization property seems to fit best with the DDM requirement in order to avoid security treats. At each data repository, mining strategy is deployed specifically for the certain domain of data.

3. OPEN PROBLEMS STRATEGY

Several systems have been developed for distributed data mining. These systems can be classified according to their strategy to three types; central learning, meta-learning, and hybrid learning.

3.1 Central learning strategy

is when all the data can be gathered at a central site and a single model can be build. The only requirement is to be able to move the data to a central location in order to merge them and then apply sequential DM algorithms. This strategy is used when the geographically distributed data is small. The strategy is generally very expensive but also more accurate [10]. The process of gathering data in general is not simply a merging step; it depends on the original distribution. Agent technology is not very preferred in such strategy.

3.2 Meta-learning strategy

offers a way to mine

classifiers from homogeneously distributed data. Meta-learning follows three main steps. 1) To generate base classifiers at each site using a classifier learning algorithms. 2) To collect the base classifiers at a central site, and produce meta-level data from a separate validation set and predictions generated by the base classifier on it. 3) To generate the final classifier (meta-classifier) from meta-level data via a combiner or an arbiter. Copies of classifier agent will exist or deployed on nodes in the network being used. Perhaps the most mature systems of agent-based meta-learning systems are: JAM system [11], and BODHI [11].

3.3 Hybrid learning strategy is a technique that combines local and centralized learning for model building [15]; for example, Papyrus [12] is designed to support both learning strategies. In contrast to JAM and BODHI, Papyrus can not only move models from site to site, but can also move data when that strategy is desired. Papyrus is a specialized system which is designed for clusters while JAM and BODHI are designed for data classification. The major criticism of such systems is that it is not always possible to obtain an exact final result, i.e. the global knowledge model obtained may be different from the one obtained by applying the one model approach (if possible) to the same data. Approximated results are not always a major concern, but it is important to be aware of that. Moreover, in these systems hardware resource usage is not optimized. If the heavy computational part is always executed locally to data, when the same data is accessed concurrently, the benefits coming from the distributed environment might vanish due to the possible strong performance degradation. Another drawback is that occasionally, these models are induced from databases that have different schemas and hence are incompatible.

Autonomous agent can be treated as a computing unit that performs multiple tasks based on a dynamic configuration. The agent interprets the configuration and generates an execution plan to complete multiple tasks. [7], [14], [8], [6], and [9] discuss the benefits of deploying agents in DDM systems. Nature of MAS is decentralization and therefore each agent has only limited view to the system. The limitation somehow allows better security as agents do not need to observe other irrelevant surroundings. Agents, in this way, can be programmed as compact as possible, in which light-weight agents can be transmitted across the network rather than the data which can be more bulky. Being able to transmit agents from one to another host allows dynamic organization of the system. For example, mining agent *ma*, located at repository r1, posses algorithm *alg1*. Data mining task t1 at repository *r*2 is instructed to mine the data using a*lg*1. In this setting, transmitting *alg*1 to *r*2 is a probable way rather than transfer all data from *r*2 to r1 where *alg*1 is available.

4. AGENT-BASED DISTRIBUTED DATA MINING (ADDM)

ADDM is a novel data mining technique that inherits all powerful properties of agents and, as a result, yields desirable characteristics. In general, constructing an ADDM system concerns with three key characteristics: interoperability, dynamic system configuration, and performance. Interoperability concerns with collaboration of agents in the system, and external interaction which allow new agents to enter the system seamlessly. The architecture of the system must be open and flexible so that it can support the interaction including communication protocol, integration policy, and service directory. Communication protocol covers message encoding, encryption, and transportation between agents. Integration policy specifies how a system behaves when an external component, such as an agent or a data site, requests to enter or leave. Dynamic system configuration, handles the dynamic configuration of the system, and is a challenge issue due to the complexity of the planning and mining algorithms. A mining task may involve several agents and data sources, in which agents are configured to equip with an algorithm and deal with given data sets. In distributed environment, tasks can be executed in parallel, in exchange, concurrency issues arise. Quality of service control in performance of data mining and system perspectives is desired; however it can be derived from both data mining and agent's fields.

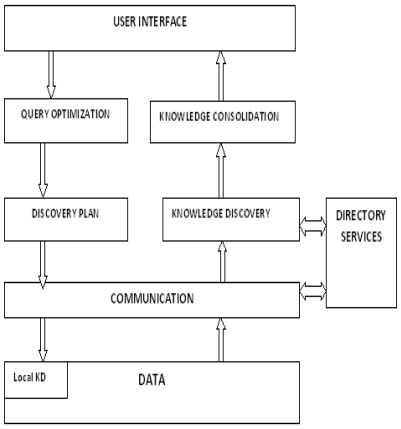

Fig.4.1: ADDM System Architecture

An ADDM system can be generalized into a set of components and viewed as depicted in figure 4.1.We may generalize activities of the system into request and response, each of which involves a different set of components. Basic components of an ADDM system are as follows.

Data: Data is the foundation layer of the architecture. In distributed environment, data can be hosted in various forms, such as online relational databases, data stream, web pages, etc., in which purpose of the data might be varied.
Communication: The system chooses the related resources from the directory service, which maintains a list of data sources, mining algorithms, data schemas, data types, etc. The communication protocols may vary depending on implementation of the system, such as client-server, peer-

to-peer etc.

User Interface: The user interface (UI) interacts with the user as to receive and respond to the user. The interface simplifies complex distributed systems into user-friendly message such as network diagrams, visual reporting tools, etc. On the other hand, when a user requests for data mining through the UI, the following components are involved.

Query optimization: A query optimizer analyses the request as to determine type of mining tasks and chooses proper resources for the request. It also determines whether it is possible to parallelize the tasks, since the data is distributed and can be mined in parallel.

Discovery Plan: A planner allocates sub-tasks with related resources. At this stage, mediating agents play important roles as to coordinate multiple computing units since mining sub-tasks performed asynchronously as well as results from those tasks. On the other hand, when a mining task is done, the following components are taken place,

Local Knowledge Discovery (KD): In order to transform data into patterns which adequately represent the data and reasonable to be transferred over the network, at each data site, mining process may take place locally depending on the individual implementation.

Knowledge Discovery: Also known as mining, it executes the algorithm as required by the task to obtain knowledge from the specified data source.

Knowledge Consolidation: In order to present to the user with a compact and Meaningful mining result, it is necessary to normalize the knowledge obtained from various sources. The component involves complex methodologies to combine knowledge/ patterns from distributed sites. Consolidating homogeneous knowledge/patterns is promising and yet difficult for heterogeneous case.

5. PROPOSED SCHEMA

Building and managing of large-scale distributed systems is becoming an increasingly challenging task. Continuous intervention by user administrators is generally limited in large-scale distributed environments. System support is also needed for configuration and reorganization when systems evolve with the addition of new resources. The primary goal of the management of distributed systems is to ensure efficient use of resources and provide timely service to users. Most of the distributed system management techniques still follow the centralized model that is based on the client-server model. Centralization have presented some problems, such as: 1) it could cause a traffic overload and processing at the manager node may affect its performance;2) it does not present scalability in the increase of the complexity of the network; 3) the fault in the central manager node can leave the system without a manager.

One model is the distributed management where management tasks are spread across the managed infrastructure and are carried out at managed resources. The goal is to minimize the network traffic related to management and to speed up management tasks by distributing operations across resources. The new trend in distributed system management involves using multi-agents to manage the resources of distributed systems. Agents have the capability to autonomously travel (execution state and code) among different data repositories to complete their task. The route may be predetermined or chosen dynamically depending on the results at each local data repository.

The concept of multi-agents promises new ways of designing applications that better use the resources and services of computer systems and networks. For example, moving a program (e.g., search engine) to a resource (e.g., database) can save a lot of bandwidth and can be an enabling factor for applications that otherwise would not be practical due to network latency.

Conceptually, a multi-agent can migrate its whole virtual machine from host to host; it owns the code, not the resources. Multi-agents are the basis of an emerging technology that promises to make it very much easier to design, implement, and maintain distributed systems. We have found that multi-agents reduce network traffic, provide an effective means of overcoming network latency, and, perhaps most importantly, through their ability to operate asynchronously and autonomously of the process that created them help us to construct more robust and fault tolerant systems. The purpose of the proposed multi-agent system is to locate, monitor, and manage resources in distributed systems. The system consists of a set of static and mobile agents. Some of them reside in each node or element in the distributed system. There are two multi-agents named delegated and collector agents that can move through the distributed system. The role of each agent in the multi-agent system, the interaction between agents, and the operation of the system

5.1 Structure of Multi-Agent Systems

The multi-agent system structure assumes that each node in the system will have a set of agents residing and running on that node [1]. These agent types are the following:

Client agent (CA) percepts service requests, initiated by the user, from the system. The CA may receive the request from the local user directly. In the other case, it will receive the request from the exporter agent coming from another node.

Service list agent (SLA) has a list of the resource agents in the system. This agent will receive the request from the CA and send it to the resource availability agent. If the reply indicates that the requested resource is local then the service list agent will deliver the request to the categorizer agent. Otherwise, it will return the request to the CA.

Resource availability agent (RAA) indicates whether the requested resource is free and available for use or not. It also indicates whether the requested resource is local or remote. It receives the request from the service list agent and checks the status of the requested resource through the access of the MIB. The agent then constructs the reply depending on the retrieved information from the database.

Resource agent (RSA) is responsible for the operation and

control of the resource. This agent executes the on the resource. Each node may have zero or more RSAs.

Router agent (RA) provides the path of the requested resource on the network in case of accessing remote resources. Before being dispatched, the exporter agent will ask the router agent for the path of the requested resource. This in turn delivers it to the exporter agent.

Categorizer agent (CZA) allocates a suitable resource agent to perform the user request. This agent percepts inputs coming from the service list agent. It then tries to find a suitable free resource agent to perform the requested service.

Exporter agent (EA) is a mobile agent that can carry the user request through the path identified by the RA to reach the node that has the required resource. It passes the requested resource *id* to the RA and then receives the reply. If the router agent has no information about the requested resource, the EA will try to locate the resource in the system. There are also two additional mobile agent types exist in the system.

Delegated agent (DA) is a mobile agent that is launched in each sub network. It is responsible for traversing sub network nodes instead of theexporter agent to do the required task and carry results back to the exporter agent.

Collector agent (CTA) is a mobile agent that is launched from the last sub network visited by the exporter agent. It is launched when results from that sub network become available. This agent goes through the reversed itinerary of the exporter agent trip. The CTA collects results from the delegated agents and carries it to the source node. All mobile agents used here are of interrupt driven type.

5.2 Functionality of the System

The activity cycle of the multi-agent system resides in a local data repository. The client agent receives the service requests either from the user or from an exporter agent. The client agent then asks the service list agent for the existence of a resource agent that can perform the request. The service list agent checks the availability of the required resource agent by consulting a resource availability agent to perform the requested service. The reply of the resource availability agent describes whether or not the resource is locally available and whether or not there is a resource agent that can perform the requested service. If the resource availability agent accepts the request, the service list agent will ask the categorizer agent to allocate a suitable resource agent to the requested service and the resource agent will perform the requested service. Otherwise, the service list agent informs the client agent with the rejection and is passed to the exporter agent. The exporter agent asks the router agent for the path of the required resource agent. Once the path is determined, the exporter agent will be dispatched through the network channel to the destination node identified by that path. If the router agent has no information about the location of the required resource agent, the exporter agent will search the distributed system to find the location of the required resource agent and assign the required task to it.

As shown in Fig. 5.2.2, the exporter agent traverses the sub networks of the distributed system through its trip. At each sub network, a delegated agent is launched to traverse the local nodes of that sub network doing the required task and carrying results of that task.

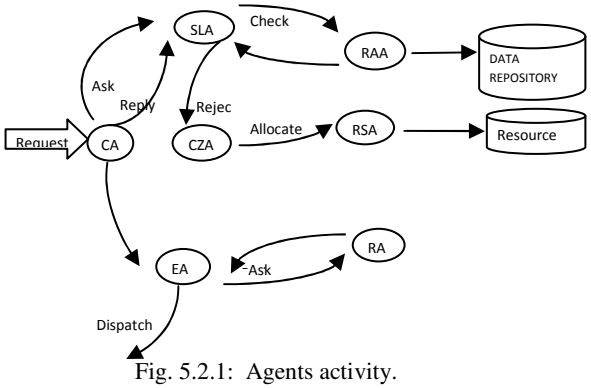

Fig. 5.2.1: Agents activity.

The agents of the social interface described in Fig. 5.2.1 are implemented at each node in the system. There are two approaches to collect results of the required task and send these results back to the source.

In traditional agent-based management systems that use mobile agent, the exporter agent will wait at each visited sub network until the delegated agent finishes its work and obtains results. Then, the exporter agent will take these results and go to the next sub network in its itinerary. The exporter agent will return to its home sub network after visiting all the sub networks determined in the itinerary. The home sub network of the exporter agent is the sub network from which it was initially dispatched. The waiting of the exporter agent prevents execution of tasks to be started in the other sub networks. This approach is used in most of the previously developed management systems in which operation is based on mobile agents. In the proposed multi-agent management system, the exporter agent does not wait for results from each sub network. It resumes its trip visiting other sub networks, and at each sub network, another delegated mobile agent is launched to carry out management tasks instead of the exporter agent. The exporter agent will be killed at the last visited sub network in its itinerary. When results from the last visited sub network become available, another mobile agent called collector agent is launched or dispatched from this sub network to collect results from it and other sub networks. The collector agent goes through the reversed itinerary of the exporter agent trip carrying results to the home sub network. In this manner, operations can be done in a parallel fashion at different sub networks because there is no delay of the task submission to local data repositories of these sub networks.

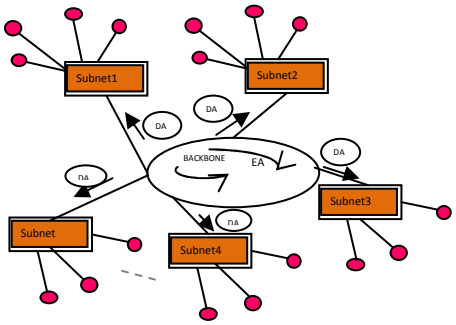

Fig. 5.2.2: Network architecture of ADDM.

6. CONCLUSION

Distributed management for distributed systems is becoming a reality due to the rapid growing trend in internetworking and the rapid expanding connectivity. This paper describes a new multi-agent system for the management of distributed systems. The system is proposed to optimize the execution of management functions in distributed systems. The proposed system can locate, monitor, and manage resources in the system. The new technique in that system allows management tasks to be submitted to sub networks of the distributed system and executed in a parallel fashion. The proposed system uses two multi- agents. The first is used to submit tasks to the sub networks of the distributed system and the other collects results from these sub networks. The proposed system is compared against traditional management techniques in terms of response time, speedup, and efficiency. A prototype has been implemented using performance management as the case study. The performance results indicate a significant improvement in response time, speedup, efficiency, and scalability compared to traditional techniques. The use of JVM in the implementation of the proposed system gives the system a certain type of portability. Therefore, it is desirable to use the proposed system in the management of distributed systems. The proposed system is limited to be applied to high-speed networks that have bandwidth 100 Mb/s or more. Also, the system cannot work when a failure occurs.

Future research will be related to the security of mobile agents and of hosts that receive them in the context of public networks. Mobile agents should be protected against potentially malicious hosts. The hosts should also be protected against malicious actions that may be performed by the mobile code they receive and execute. So, a detailed design and implementation of the whole secure system should be considered as a future work. Also, the high complexity of distributed systems could increase the potential for system faults. Most of the existing management systems assume that there is no fault in the system. It would be interesting to develop a fault tolerant management system that introduces safety in the system and attempts to maximize the system reliability without extra hardware cost.

7. REFERENCES:

[1] T.C. Du, E.Y. Li, and A. Chang,—Mobile agents in distributed network management, Commun. ACM, vol. 46, no. 7, pp. 127–132, July 2003.

[2] H. Ku, G.W.R. Ludere, and B.Subbiah, —An intelligent mobile agent framework for distributed network management, in Proc. Globecom'97Phoenix, pp. 160–164.

[3] N. R. Jennings and S. Bussmann, —Agent-based control systems—Why are they suited to engineering complex systems? IEEE Control Syst.Mag., vol. 23, no. 3, pp. 61–73, Jun. 2003.

[4] R. B. Patel, Neeraj Goel, —Mobile Agents in Heterogeneous Networks: A Look on Performance, Journal of Computer Science,2(11): 824-834, 2006.

[5] O'Hare G.M.P., Marsh D., Ruzzelli A., R. Tynan,—Agents for Wireless Sensor Network Power Management, in Proceedings of International Workshop on Wireless and Sensor Networks (WSNET-05), Oslo, Norway IEEE Press, 2005.

[6] S. Bailey, R. Grossman, H. Sivakumar, and A. Turinsky. Papyrus: a system for data mining over local and wide area clusters and super-clusters. In Supercomputing '99: Proceedings of the 1999 ACM/IEEE conference on Supercomputing (CDROM), page 63, New York, NY, USA, 1999. ACM

[7] R. J. Bayardo, W. Bohrer, R. Brice, A.Cichocki, J. Fowler, A. Helal, V. Kashyap, T. Ksiezyk, G. Martin, M. Nodine, and Others.InfoSleuth: agent-based semantic integration of information in open and dynamic environments. ACM SIGMOD Record, 26(2):195–206, 1997.

[8] F. Bergenti, M.P. Gleizes, and F.Zambonelli. Methodologies And Software Engineering For Agent Systems: The Agent oriented Software Engineering Handbook. Kluwer Academic Publishers, 2004.

[9] R. Bose and V. Sugumaran. IDM: an intelligent software agent based data mining environment. 1998 IEEE International Conference on Systems, Man, and Cybernetics, 3, 1998.

[10] J. Dasilva, C. Giannella, R. Bhargava, H.Kargupta, and M. Klusch. Distributed datamining and agents. Engineering Applicationsof Artificial Intelligence, 18(7):791–807,

October 2005.

[11] S. Datta, K. Bhaduri, C. Giannella, R. Wolff,and H. Kargupta. Distributed data mining in peer-to-peer networks. Internet Computing,IEEE, 10(4):18–26, 2006.

[12] U. Fayyad, R. Uthurusamy, and Others. Data mining and knowledge discovery in databases.
Communications of the ACM,39(11):24–26, 1996.

[13] Vladimir Gorodetsky, Oleg Karsaev, and Vladimir Samoilov. Multi-agent technology for distributed data mining and classification.In IAT, pages 438–441. IEEE Computer
Society, 2003.

[14] Sven A. Brueckner H. Van Dyke Parunak. Engineering swarming systems.Methodologies and Software Engineering for Agent Systems, pages 341–376, 2004.

[15] W. Davies and P. Edwards. Distributed Learning: An Agent-Based Approach to Data-Mining. In *Proceedings of MachineLearning 95 Workshop on Agents that Learn from Other Agents*, 1995.