A Multi-Agent Architecture for Distributed Data Mining Systems

Gustavo H. B. S. Oliveira[1], Josenildo C. da Silva[1], Omar A. C. Cortes[1], Luciano R. Coutinho[2]

[1]Instituto Federal do Maranhão (IFMA) – Av. Getúlio Vargas, 04, Monte Castelo
CEP 65030-005 – São Luís, MA, Brazil

[2]Universidade Federal do Maranhão (UFMA)
Av. dos Portugueses – Bacanga – São Luis – MA – Brazil

{gustavo.oliveira, jcsilva, omar}@ifma.edu.br, luciano.rc@ufma.br

Abstract. *The agent-based approach is appealing to Distributed Data Mining (DDM) systems since the concept of agency offers some relevant features, such as scalability, flexibility, robustness, and modularity. This paper investigates whether and how the multi-agent system metaphor might be used for Distributed Data Mining Systems. We proposed and implemented a multi-agent architecture called SeAMS, which is capable of mining patterns efficiently using the DPD-TS algorithm. The system was developed in JADE and designed to be easily extensible and protect any local datasets' privacy. Results show that the agent-based approach was able to identify patterns efficiently using three different popular scientific datasets in a distributed time series: Sunspot, Power, and TEK.*

1. Introduction

Distributed Data mining (DDM) has faced two main challenges in the past years. Firstly, the amounts of data are generated too fast to be processed even for single supercomputers [Zeng et al. 2012]. Secondly, massive data can be stored at multiple locations, and it becomes increasingly expensive to centralize it in one place [Zeng et al. 2012]. In this context, clusters of workstations, grid computing infrastructures, multicore, many-core technologies, and cloud computing make distributed and parallel data mining appealing in many application fields. Therefore, it is natural to move from centralizing services to cloud computing, which usually offers cheaper storage data services.

Taking this scenario into account, a practical approach to implementing a DDM system ends up being based on both distributed and parallel computing [Di Fatta and Fortino 2007]. One could follow the client-server model in which a central entity starts and coordinates the mining session while the other entities act as data clients. On the other hand, one could implement a set of web-based services to build a network of data sources and data mining algorithms. Regardless of the number of parallel and distributed technologies, real-world scenarios can involve different companies or institutions. Thus, the critical questions appear: who will coordinate the mining session? Who will combine individual partial results into a valid global model? What happens if one of the parties goes off-line during the mining session? Moreover, in a distributed scenario, each party might need to follow local privacy policies regulating how data can

be used beyond its original purpose. Additionally, each party might want to contribute by using new approaches and algorithms that require flexible and extensible distributed mining applications. Therefore, traditional methods seem not to be suitable for implementing a DDM system.

Thus, our contribution is proposing and implementing a distributed agent-based architecture called SeAMS (Secure Agented-Based Distributed Data Mining System), a system capable of mining patterns in temporal series. Consequently, we also propose how to solve the referred questions. The principle is that in the Distributed Artificial Intelligence (DAI) field, distributed problems are solved by a community of independent entities called agents, valuable abstractions in complex scenarios with no central authorities. Moreover, the agent-based approach is appealing since the concept of agency offers many essential features to DDM, including scalability, flexibility, robustness, and modularity [da Silva et al. 2005].

This paper is divided as follows: Section 2 introduces the related works in the field of DMM and DAI; Section 3 presents the SeAMS architecture, communication protocols between the agents, its functionalities, and some applications in three different scenarios; finally, Section 4 presents the conclusions of this work and future work.

2. Related Work

Among the emergent paradigms for distributed computing, the Agent model has been demonstrated to be particularly suitable for supporting the construction of flexible and effective frameworks for distributed computation [Di Fatta and Fortino 2007]. Several works have been created using agents in DDM as follows:

- EMADS (Extendible Multi-Agent Data Mining System) [Albashiri et al. 2009] is a hybrid peer-to-peer agent-based system for distributed data mining. It was implemented in JADE and includes data mining agents, data agents, task agents, user agents, and JADE-specific agents for mediation and coordination. EMADS considers three mining tasks: classification, clustering, and association rules mining, though it is not bound to any specific algorithm;
- MADM [Chaimontree et al. 2010] was designed as a generic data mining framework allowing the inclusion of new algorithms as needed. It introduced an extendable ontology to describe message semantics exchanged among the agents. They demonstrate the usefulness of MADM framework by building a distributed clustering application. The system was implemented in JADE;
- BNNMAS [Hafezi et al. 2015] proposed a multi-agent-based system for stock price prediction. The design is tailored to the financial market, and no general application is discussed. No details on implementation are given;
- EMAS [Golzadeh et al. 2018] present a multi-agent system for prediction problems. It is based on the ensemble of experts and was implemented in Java with JADE framework. EMAS includes agents for data pre-processing, data mining, and presentation of results.

Several agent-based systems we reviewed have been implemented in JADE framework [Bellifemine et al. 2007]. JADE (Java Agent DEvelopment framework) is a middleware that enables developers to implement multi-agent systems in various settings. It handles several agent-related tasks such as creation, communication, complex behavior, and

protocols. JADE adopts a peer-to-peer approach and supports both desktops and mobile devices.

The main difference between those approaches and our architecture lies in the concept of peers. Each peer has access to a local dataset only, thus, avoiding that other peers could see the real data. Then, a single peer assumes the responsibility of aggregating the mining sections and providing the final results. This concept will be explained in detail in Section 3.

3. SeAMS: Secure Agent-Based Distributed Data Mining System

SeAMS is an agent-based system for mining patterns in distributed time series. It has been implemented in Java 1.8 with JADE Framework [Bellifemine et al. 2007]. It is easily extendible to offer new mining algorithms by registering new agents to the system. It is important to remark that each peer will enforce a local privacy policy. This is dependent on the mining task but, in general, means that a given peer will not take part in a mining session if certain parameters are not satisfied. Moreover, raw data will not be sent over the network but only models or statistical descriptions of the dataset.

Currently, SeAMS supports *pattern discovering* in time series and uses the DPD-TS algorithm [da Silva et al. 2012]. However, new mining tasks and algorithms can be added to the system easily because a given mining algorithm is coded into the system as a group of agents. In this context, although it is necessary to implement the specific behavior of particular mining agents to include new mining tasks in the system architecture presented in Figure 1.

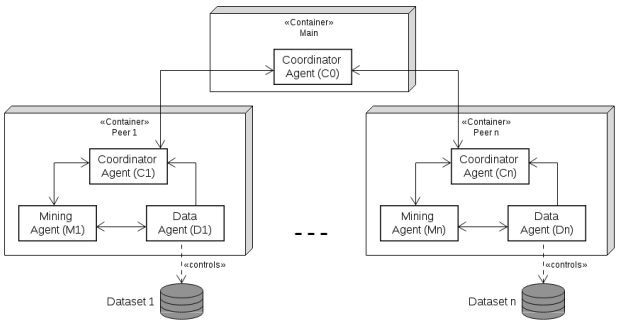


Figure 1. SeAMS components architecture

The architecture is devised by three types of agents: *coordinators*, *data agents*, and *mining agents*. All of them run in JADE *containers* called *peers*. **Coordinators** are agents responsible for holding global information, such as the number and localization of active agents, and mediating the interaction among the other agents when required. There are two kinds of coordinators: a *central coordinator* and several *Local coordinators*. The central coordinator works as a global reference providing essential information for the system's overall functioning. It is responsible for peer registration and intermediates communication among local coordinators during the mining sessions. Local coordinators are the information providers and message routers inside the peers. There is only one coordinator whose primary role is to facilitate the interactions between intra and inter peers within a peer.

Mining agents are responsible for the execution of data mining sessions. They start generating local results and finish mining sessions, performing the main global mining tasks expected from the system.

The **Data agent** manages the databases. These are the only agents that have direct access to the local data. Their role is to ensure that any other agent in the system does not access the local dataset. All agents are composed of one or more *behaviours*. Some behaviors represent steps of the mining algorithm to be performed. Other behaviors enable the agents to send and receive messages following specific protocols.

As a conceptualization, we can state that protocols drive the actions and interactions between agents. In general, coordinators directly communicate among themselves. Within a peer, the local coordinator can receive messages from all other agents. However, it is able to send messages only to other coordinators and mining agents. The dataset agent can send messages to the mining agent or the coordinator and receive messages only from the mining agent. Finally, the mining agent establishes bidirectional connections with all other agents, including mining agents located in different peers. The next subsection details the protocols.

3.1. Protocols

A protocol is a sequence of actions to be done, including message exchanges, to accomplish one or more steps in distributed data mining. They define what basic behaviors should be performed and the order for a given joint action to be executed. Messages received by the agents activate these behaviors. Thus, there are several kinds of messages, each one implying a different behavior. Specifically, the message names indicate the exact response to be executed by the message recipient.

The main protocols are: *register peer*, *announce session*, *negotiate parameter*, *performing mining activity*, and *cancel mining*. The first one is the addition of new peers to the system, each peer running in a distributed node containing one or more datasets. The other use case is the execution of mining sessions involving coordinators, mining agents, and dataset agents; these last two implements a specific mining algorithm. In the sequel, the main protocols are discussed in more detail.

Register peer. It is the protocol that should be executed every time a new peer is instantiated. For every new peer, a registration request is sent by the local coordinating agent to the SeAMS central coordinating agent, which receives and stores the new peer's address. After this, if the peer is not registered yet, a confirmation message named *accept registration* is sent to the local coordinator agent.

Announce session. This protocol is used to publish a new data mining session. It is always initiated by a mining agent that implements a given mining task/algorithm chosen by the SeAMS user. It begins with the mining agent sending a *propose mining session* message to the local peer coordinator. Composing the message, the initiator agent appends a set of mining parameters configured by the user. After receiving the message, the local coordinator forwards it to all known mining agents and the other known coordinators. Thus, the *propose mining session* containing the mining parameters is propagated throughout the peers. When receiving the announcement of a new session, the mining agents send a *verify local dataset message* to their respective local dataset agents. After this, a response to the request is sent to the local mining agent, stating whether there is a

local database satisfying the parameters. If there is a database that matches the parameters, the local mining agent replies to the session initiator with an *accept mining invitation* message; otherwise, it refuses the mining invitation. The session initiator registers all the mining agents that have accepted the invitation as members of the newly created data mining group, which can be located in the same or different peers.

Negotiate parameters. This protocol is similar to the final steps of the previous protocol. It extends and refines the process of reaching a mutual agreement among the session participants concerning the mining parameters. The protocol starts the session initiator mining agent, which sends a *propose parameters* message for all the current mining session members. After receiving the message, each member sends a new *verify local dataset* to its associated dataset agent, containing a more detailed set of parameters. Then, the dataset agents verify the parameters by comparing their local privacy policies related to the databases and reply to the request by issuing a *response to dataset request* message to the mining agents. If the answer is no, the local mining agent sends to the session initiator a *refuse parameters* message along with a different but possible parameter list according to the dataset characteristics and rules of the local dataset agents. Upon receiving these refuses, the initiator can modify parameters and resend the *propose parameters* message. This cycle is maintained until a portion or, at best, all participants agree on the parameters. When this happens, *accept parameters* messages are sent to the session initiator with the mining agent in agreement, which, finally, consolidates the actual mining group.

Perform mining activity. This protocol is initiated when it is time to calculate the local mining task and then send the result to the next member of the mining group. The protocol's first message is *request local model*, which is sent from the local mining agent to the local dataset agent. Upon receiving the message, the dataset agent, using the agreed session parameters, calculates the local model and sends it to the mining agent via the message *continue mining activity*. In the sequence, the local *mining agent* updates the partial model with data from the local model and sends the updated partial model to the next member of the data mining session. Suppose the next mining agent is not the initiator. In that case, the message cycle is repeated in order to calculate the local model, update the partial model with the local model and send the result to the next member. Otherwise, the session initiator agent sends the *announce global model* message with the final global model to all members of the data mining session.

Cancel mining. Finally, there is the *cancel mining* protocol. It is a protocol that can be started by any agent in the data mining session. Firstly, the mining agent sends a *cancel mining* message to the session initiator agent. After that, the session initiator sends a *wait for the mining conclusion* message to the agent requesting the cancellation to ensure that it really wants to leave the session before being completed. Upon receiving this message, the agent reevaluates whether or not it wants to leave the mining process. If it reconsiders the request and decides not to leave, the message *announce withdrawl* is sent to all agents participating in the session so that they can still count on the agent for the data mining task.

3.2. SeAMS Functionality

The SeAMS application is divided into four modules: ***datasets***, ***data mining sessions***, ***peers online***, and ***user seetings***. In ***datasets*** module, the user can see all data sets he

can work with, locally. Furthermore, the user can delete, edit, and include new databases. Each data set must be configured according to a list of basic configurations, which indicate privacy parameters and a description of the data set. The module ***peers online*** informs which agents are connected to the platform. That is necessary because all agents must be running to start the process of mining a time series.

The ***data mining sessions*** module is responsible for creating, managing, starting, and visualizing all results from a data mining process. Finally, the user module controls who can use the system, controlling access to the system through encrypted passwords. Five states compose a data mining session: created, announced, ongoing, canceled, and finalized. When a mining session is created, it receives the "created" status. After that, the agent that started the mining section has to announce itself to all other mining agents in the system; thus, its status changes to "announced". Then, when the mining session starts, the status is modified to "ongoing". If the user wants to cancel the session, then a message is sent to all mining agents, and its status changes to "canceled". On the other hand, if the mining process has finished, the status is modified to "finalized".

It is crucial to notice that when a session is created, the user can set all parameters, which includes what he wants to mine and the algorithms available to the task. Once the status changes to "announced", those settings cannot be changed. When the task is completed, the user can see the results graphically.

3.3. Application Scenario: distributed pattern discovery in time series

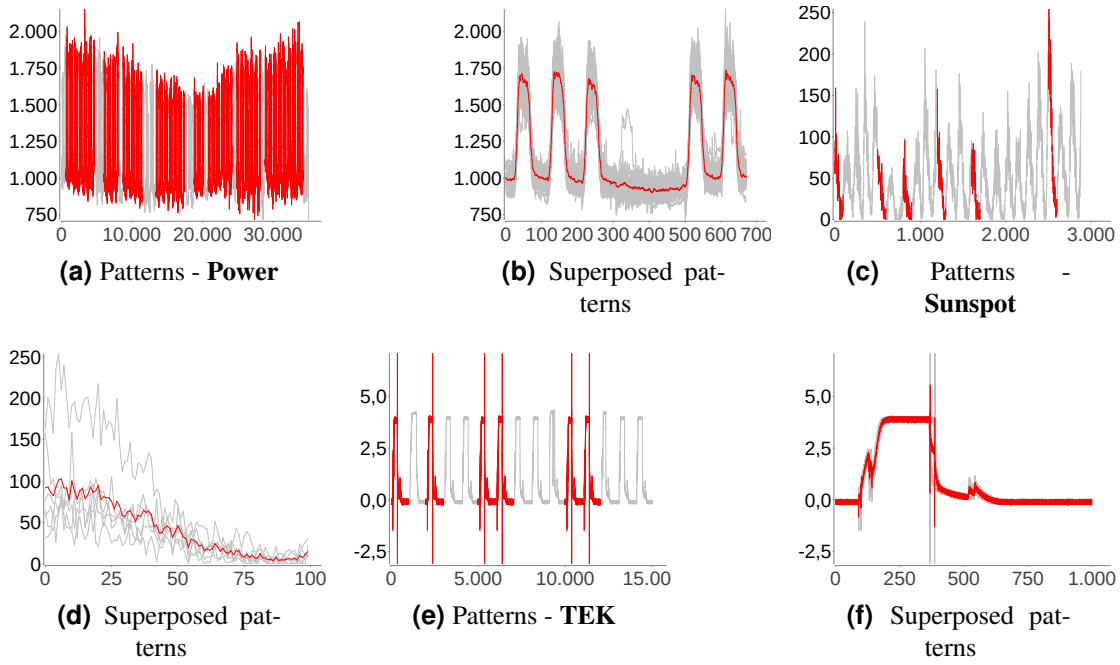
In this section, we show an experimental evaluation of SeAMS to find patterns using the algorithm DPD-TS in the following time-series datasets: **Sunspot dataset -** This dataset records the monthly average number of sunspots from January 1749 until 1993. There are $2\,880$ data points in this dataset*. This dataset was downloaded from UCR Time Series Data Homepage [Keogh et al. 2011]. The **Power dataset** presents the electricity consumption from the Netherlands Energy Research Foundation (ECN) for one year, recorded every 15 minutes. There are $35\,040$ data points corresponding to the year 1997. This dataset was downloaded from UCR Time Series Data Homepage † [Keogh et al. 2011]. Finally, the **TEK dataset -** is a time series dataset records solenoid current measurements on a Marotta MPV-41 series valve as the valve is cycled on and off under various test conditions in a laboratory. The valves are used to control fuel flow on the NASA Space Shuttle ‡. All experiments were run on a computer with *Intel Core* i7 (2.40 GHz), with 8 GB RAM and *Windows* 8.1 64 bits. The DPD-TS algorithm [da Silva et al. 2012] works as follows. As the mining session starts, local data agents compute local density estimates. Results presented in Figure 2 are related to all datasets.

Figure 2 shows patterns found in several datasets, highlighted in red in all subfigures. In power dataset (Figure 2 a and b), the x-axis represents time and y-axis represents power consumption in KWh. The most frequent pattern corresponds to a typical week with five days of high consumption, followed by two low consumption days. Notice that the first day of the power dataset was a Wednesday; therefore, the two low consumption days are in the middle section of the pattern. This behavior is a particularity of DPD-TS

*For up to date sunspot data visit `http://sidc.oma.be/`

†https://www.cs.ucr.edu/ eamonn/time_series_data_2018/

‡`http://cs.fit.edu/~pkc/nasa/data/`



(a) Patterns - **Power**

(b) Superposed patterns

(c) Patterns - **Sunspot**

(d) Superposed patterns

(e) Patterns - **TEK**

(f) Superposed patterns

Figure 2. Patterns found on datasets.

algorithm.

Figure 2 (c and d) shows the results for *sunspot* dataset. This dataset represents a natural phenomenon without a fixed pattern size. Therefore, fewer instances were found, and the superposition of patterns shows a wider variability than in the power dataset.

Finally, Figure 2 (e and f) illustrates the results for TEK dataset. Patterns on this dataset have a very similar profile, with almost the same size. Therefore, all pattern instances are identified.

4. Conclusions

This paper presented an agent-based architecture for mining patterns in distributed time series. The proposed architecture lay in peers' concepts and was designed to be extensible, providing privacy to local datasets. Through experimental evaluation, we demonstrated that the proposed architecture could correctly identify patterns in three different time-series datasets: Sunspot, Power, and TEK. We consider adding new algorithms to the current implementation in future work, such as distributed data clustering and developing a web-based interface to the distributed system.

References

Albashiri, K. A., Coenen, F., and Leng, P. (2009). Emads: An extendible multi-agent data miner. *Knowledge-Based Systems*, 22(7):523–528.

Bellifemine, F. L., Caire, G., and Greenwood, D. (2007). *Developing multi-agent systems with JADE*. John Wiley & Sons.

Chaimontree, S., Atkinson, K., and Coenen, F. (2010). Multi-agent based clustering: Towards generic multi-agent data mining. In *Industrial Conference on Data Mining*, pages 115–127. Springer.

da Silva, J. C., Cortes, O. A. C., Oliveira, G. H. B., and Klusch, M. (2012). Density-based pattern discovery in distributed time series. In *Proc. of the 21st Brazilian Symp. on Artificial Intelligence (SBIA)*, pages 62–71. Springer.

da Silva, J. C., Giannella, C., Bhargava, R., Kargupta, H., and Klusch, M. (2005). Distributed data mining and agents. *Engineering Applications of Artifical Intelligence*, 4(18):791–807.

Di Fatta, G. and Fortino, G. (2007). A customizable multi-agent system for distributed data mining. In *Proceedings of the 2007 ACM Symposium on Applied Computing*, SAC '07, pages 42–47, New York, NY, USA. ACM.

Golzadeh, M., Hadavandi, E., and Chehreh Chelgani, S. (2018). A new ensemble based multi-agent system for prediction problems: Case study of modeling coal free swelling index. *Applied Soft Computing*, 64:109–125.

Hafezi, R., Shahrabi, J., and Hadavandi, E. (2015). A bat-neural network multi-agent system (bnnmas) for stock price prediction. *Appl. Soft Comput.*, 29(C):196–210.

Keogh, E., Zhu, Q., Hu, B., Y., H., Xi, X., Wei, L., and Ratanamahatana, C. A. (2011). The ucr time series classification/clustering homepage. Availabe at www.cs.ucr.edu/∼eamonn/time_series_data/.

Zeng, L., Li, L., Duan, L., Lu, K., Shi, Z., Wang, M., Wu, W., and Luo, P. (2012). Distributed data mining: a survey. *Information Technology and Management*, 13(4):403–409.