Bulletin of Electrical Engineering and Informatics
Vol. 10, No. 5, October 2021, pp. 2759~2770
ISSN: 2302-9285, DOI: 10.11591/eei.v10i5.2841

Framework proposal for adaptive mobile intelligent agents

Elena Fabiola Ruiz-Ledesma[1], Rosaura Palma-Orozco[2], Elizabeth Acosta-Gonzaga[3]
[1,2]Instituto Politécnico Nacional, Escuela Superior de Cómputo, CDMX, México
[3]Instituto Politécnico Nacional, Unidad Interdisciplinaria de Ingeniería Ciencias Sociales y Administrativas, CDMX, México

Article Info

Article history:

Received Jan 29, 2021
Revised Apr 30, 2021
Accepted Jul 8, 2021

Keywords:

Artificial intelligence
Intelligents agents
Mobile agents
Multi-agent systems

ABSTRACT

Intelligent agents are computational entities which have elements that provide them with the ability to perceive and manipulate their environment: sensors and actuators. These are characterized by displaying various properties that adapt and achieve their objectives. Autonomy, learning, collaboration and reasoning are examples of these properties which together make them intelligent artificial entities. This article shows the development of a framework that has made it possible to speed-up the construction of a system of adaptive mobile intelligent agents, called SySAge. The system agents have integrated search and learning techniques for the execution of automated processes focused on solving search, classification and optimization problems. It has been found that through learning, the agents were able to estimate input parameters and apply them in estimating the shortest route in a graph, considering cost and penalty aspects. To determine the choice of search technique, a probabilistic selection was used. The autonomous behavior of each agent was appreciated through the various attempts to solve the search problem and not to focus the information acquired individually on a single agent.

Corresponding Author:

Elena Fabiola Ruiz Ledesma
Instituto Politécnico Nacional
Escuela Superior de Cómputoy
Av. Juan de Bátiz sn Col. La Escalera, Mexico City, 07320, Mexico
Email: eruizl@ipn.mx

1. INTRODUCTION

Artificial intelligence (AI) is a worldwide high-impact computational discipline. From its beginnings to the present, it has had as its aim intelligent behavior in artificial entities. This involves acting and communicating in complex environments. The study of various skills such as reasoning, learning or perception, has allowed AI to address various alternatives to solve problems in scientific and engineering fields [1].

The present work emphasizes the integration of intelligent agents, computational entities used in various contexts such as simulation, robotics, probabilistic classification, aeronautics, medicine and other areas [2]. Intelligent agents allow the resolution of various problems using advanced simulation, coordination, planning and collaboration techniques. They also require the integration of a second computational paradigm. Mobile agents are entities capable of moving around, and of executing specific previously programmed tasks in a distributed environment made up of interconnected physical devices [3].

The architecture of an agent incorporates sensors and actuators, elements that make it an entity capable of perceiving and manipulating its environment [2]. A reactive agent is an entity that only has these elements. It is also called a stimulus-response agent (Agent S-R) and is capable of solving simple tasks such

as avoiding obstacles or following lines. On the other hand, an intelligent agent is capable of communicating and collaborating with other agents; this allows it to carry out a specific task through various planning and coordination mechanisms [1]. When a system integrates more than one agent, it is called a multi-agent system (MAS) [4]. Together, the previous schemes are consolidated as advanced artificial intelligence techniques that try to solve problems that involve search, optimization, classification, coordination and planning situations [5].

For the resolution of automated processes, the construction of a model of intelligent agents is required to optimize the development of MAS. The problem of this article focuses on the fact that there are frameworks for MAS that provide various tools that allow the development of multi-agent environments. However, they do not have their own simulation environment; they are proprietary, have insufficient documentation, they do not have a mechanism for extension or persistence, and work with specific technologies such as Java or Python [6]. This situation makes it impossible for the integration and interoperability with secondary systems that require the use of intelligent agents. This results in the construction of multi-agent environments from scratch, which results in a considerable investment over development time [7].

It is considered that the construction of a model of adaptive mobile intelligent agents reduces the development time of MAS, by optimizing interoperability with systems that integrate intelligent agents, and by covering specific characteristics such as its own simulation environment. This is an extensibility mechanism and independence of the programming language, aspects that some frameworks for current MAS, lack. Due to this aspect, a framework is built supported by a model of intelligent agents (IA) that give life to a system called SySAge, which is also described in this work.

The article is divided into four sections, the first gives an account of the study problems, the second provides the state-of-the-art where a comparison of different frameworks is made according to 8 established metrics. The third section describes the design and development of the system framework called SySAge and shows the agent modules. In the fifth section, the results are presented through the tests, allowing the search and learning operations of the agents to be carried out, verifying their autonomy and classification characteristics. Finally, the conclusions are given.

2. THEORETICAL ASPECTS

In general, the area of MAS is expanding rapidly. The expansion is even more accentuated in the programming of MAS environments, using specific languages, platforms and tools. In an effort to try to find the best way to program these highly complex, but potentially useful systems, a wide variety of frameworks have been developed [8]. A framework for MAS is a set of models that define the infrastructure of a cooperative environment. Some of these models are abstract which allows the design to be reusable. It integrates concepts, criteria and practices for the resolution of a particular problem [9].

The application programming interface of the framework creates, manages, operates and maintains the structure of an IA and the communication and collaboration mechanisms that enable group behavior. To validate the adequate functioning of the framework, applications were implemented that attempt to solve search and optimization aspects, such as air traffic, inverse kinematics of mechanical manipulators and navigation of mobile robots.

2.1. Examples of frameworks for the development of multi-agent systems

Some of the most common MAS development frameworks used in industry and academia are described below.

2.1.1. Java agent development enviroment

It is possibly the most widely used agent-oriented distributed system. It was developed by the Research and Development department of Telecom in Italy. It is currently a free project and is distributed under the LGPL license. It has an infrastructure that extends its functionality through the integration of new modules.

The development of applications based entirely on agents is greatly facilitated because it has a runtime environment that supports the life cycle, and the logical components required by the agents. It is programmed in Java, which allows the benefit of various tools, libraries or plug-ins implemented in this language [10]. The framework was used by Pólakow for the development of a proportional-derivative (PD) control mechanism composed of a network of sensors. The precision of its model increases as the number of sensors increases. So, the main task of each agent consists of the simultaneous control of each component, participating in the dynamic reconfiguration of the sensor network. Pólakow demonstrates the efficiency of

multi-agent coordination. Futhermore, it has found a bottleneck when using the centralized coordination scheme [11].

In recent years there have been initiatives to adapt java agent development enviroment (JADE) in mobile environments. Such is the case of the JADE add-on for Android project developed by Bergenti, which basically consists of a module for the platform, allowing the deployment of agents on Android devices [12]. Bergenti experimented with the integration of this module into the agent-based multi-user social environment (AMUSE) project, an environment based on agents for social games designed to solve common problems such as advanced game, turn and player management [13].

2.1.2. JASON

It consists of an interpreter for the extended version of AgentSpeak, a language permtis specifying mobile agents and defining communication rules between agents. It implements the operational semantics of languages, and provides a platform for the development of MAS with a considerable number of user-customizable features. Stabile [14] points out one of the disadvantages of this framework by showing the fact that Jason does not guarantee that the agent finishes its processing within the established time. The execution time grows considerably when more complex reasoning algorithms are implemented.

2.1.3. Agent sheets.

It is a programming environment based on visual agents, specifically designed to support the learning of computational reasoning while building games and simulations. This environment lets users do most of the programming of the agents by interacting with a graphical user interface. To specify the rules of agent behavior, users perform specific actions in the interface such as clicking, dragging and dropping elements, pressing keys, and capturing data [15].

2.1.4. Common-pool resources and multi-agent systems (CORMAS) framework

It is being developed by the center of cooperation internationale in recherche agronomique pour le development (CIRAD) in France. The main objective of the framework is not focused on making predictions about the behavior of complex systems, but rather, it focuses on providing a set of tools to help users to develop applications based on agents, by building and testing various models. It is implemented in Smalltalk using VisualWorks as the programming environment [16].

2.1.5. Ingenias

It counts with an agent development kit (ADK) for the construction and production of automated code with which it is possible to generate test sequences that in turn can be executed on the Jade platform. Through the use of data mining tools, Ingenias processes the information acquired from the experiments that are executed with MAS. The tool has been validated and evolved in a wide variety of applications based on agents [17].

Table 1 concentrates the metrics considered based on [18] for the evaluation of ten frameworks commonly used in research and industry areas. They were selected that met the highest number of metrics compared to the others evaluated. The characteristics of each framework are grouped in Table 2. The lack of persistence mechanisms and the complete dependence on the programming language in which the framework was implemented can be seen. The SySage framework is located in the last line of Table 2. According to the manner it was built, it complies with the 8 metrics displayed in Table 1. The following section shows the design and development of the framework for the SySAge multiagent system. According to the manner it was built, it complies with the 8 metrics displayed in Table 1. The following section shows the design and development of the framework for the SySAge multiagent system.

Table 1. Metrics considered for the comparison of frameworks

Metrics	Description
M1	Full documentation
M2	Java as a programming language
M3	Free license and source code available
M4	Own simulation environment
M5	Version maintenance
M6	Mechanisms of extensibility
M7	Independence of the programming language
M8	Persistence mechanism

Table 2. Comparison between frameworks

Framework	M1	M2	M3	M4	M5	M6	M7	M8
Jade [11]	x		x	x	x			
Jason [14]	x		x	x	x			
Agentsheets [15]	x			x	x	x		
Cormas [16]		x	x	x	x	x		
Ingenias [17]	x		x	x	x	x		
Repast [19]	x		x	x	x			
Madkit [20]			x		x			
Netlogo [21]	x	x		x				
Simpack [22]								
Starlogo [23]	x	x		x		x		
Framework de SySAge	x	x	x	x	x	x	x	x

3. DESIGN AND DEVELOPMENT OF THE SYSAGE FRAMEWORK

3.1. Description of the framework modules

The framework for the SysAge multi-agent system is composed of four modules, which are shown in Figure 1. The modules that integrate the SysAge framework are: Users, environments, activity logs and backups that are called logs and agents. Within the system, each intelligent agent has the same composition. The structure incorporates communication mechanisms with the environment, and with other agents through interfaces; both interfaces have input and output elements. These elements constitute the sensors (input) and the actuators (output).

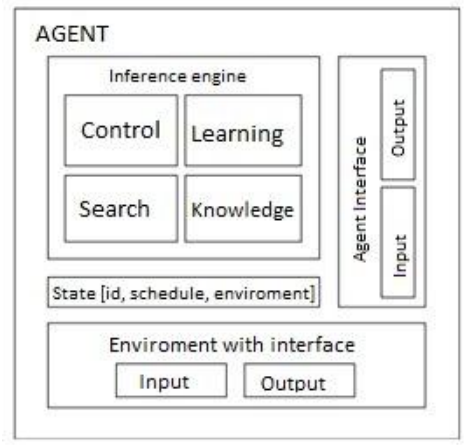

Figure 1. Graphic representation of the framework corresponding to the MAS

3.1.1. Users

Through this module it is possible to create, edit, delete user accounts; manage profiles and roles. Managing a role consists of assigning the activities allowed for the user to which a role is associated. There are various roles: administrator, manager, researcher, manipulator and tester. This module is responsible for controlling restricted access to the system.

The user with the administrator role is the only one who has access to this module. This manages secondary modules, specific components, the search engine and inference. It enables the creation of accounts with higher privilege roles and activity monitoring. It also enables the administrator to schedule frequent tasks.

3.1.2. Environments

This module was created with the objective of managing distributed environments grouped in clusters of mobile or stationary devices. It also allows configuring an identifier for each node. Its intenet protocol (IP) address, whether it would be a slave or a master, and the speed of execution of commands from the central node. It provides the environment manager the ability to monitor active nodes. Communication between nodes is carried out asynchronously by sending messages to REST web services. The agents move

between each device serializing its current state and accessing the remote computer by sending its state by asynchronous messages. Through this module the visualization and execution of MAS is processed.

3.1.3. Activity logs and backups (Logs)

This module consists of tools that allow the export of records stored in the database through the programming of total or partial back-ups in plain text with SQL format. By default, the administrator user has access to the back-up module. However, it is possible to enable the export of activity records and evaluation test results to other users in excel format. The module has tools that help with the scheduling of repetitive tasks such as the generation of continuous back-ups every time.

3.1.4. Agents

An agent can be mathematically modeled as a function that relates the set of perceptions with its respective sequence of actions [24]. Therefore, the agent can be seen as $A: P^* \rightarrow A_C$, where A is the agent function, P^* the set of perceptions and A_C the sequence of actions. Based on the above, an MAS can be modeled as it appears as shown in 1:

$$S_A = \sum_{j \in N(i)} Ai(P_{j,} A_{Cj}, N) \quad (1)$$

For $j = 1,2, \ldots., n$ and where S_A is the MAS and N is the set of objectives pursued by the agents.

The environment is the place where the agents live, and the direct communication with it allows the acquisition of the data structure that describes it. This could be a graph, a tree or a linked list. The main characteristic of this structure is the definition of nodes and their associations. The user is the one who determines the composition of the environment, indicating the number of associations that each node would have. Figure 2, shows each module that constitutes the agent structure.

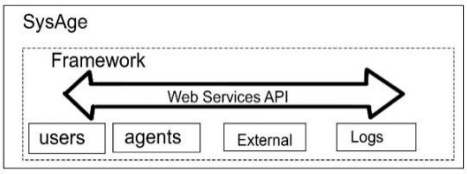

Figure 2. Graphical representation of the intelligent agent

The social behavior of an intelligent agent is strongly associated with the direct communication between each member of the community. Each agent communicates with its similar through an interface defined in its structure. This interface provides the mechanisms for sending and receiving messages. The language spoken by the agents allows the sending of data structures.

The intelligent agent state has a collection of agents defined in the agents attribute: Array:agent. This collection allows each agent the ability to communicate with other agents. If the current agent needs to send its knowledge of the field (stored in a graph), it will only be necessary to make a call to the method sendGraphtoAll (g: Graph). This method implements a cycle that runs through the entire collection and delegates in each iteration the sending of the graph through the method setSubgraph(g:Graph) of the i-agent instance.

The model incorporates a component called *state* and stores the identifier (unique name of the agent) that facilitates the identification of the entity in the environment and in the system. The itinerary consists of a list of identifiers corresponding to the mobile devices that the agent must visit; the environment where it stores a reference to the environment in which it operates; and a collection called agents, which contains the list of all the entities that exist in the environment. The storage is temporary while the agent is running, and if the persistent:true directive is facilitated in the file settings, persistent storage in the database is enabled. In particular, the state persists in the field of the same name within the agent's table.

3.2. Environment graphical representation

The environment is the place where an agent or a group of agents live. This has the ability to integrate and provide a series of routes through which each agent can travel. The environment can be instantiated locally, or it can be distributed on a set of mobile devices. Each device constitutes a node. By having a series of nodes available to the environment, it becomes a distributed environment.

More than one agent can co-exist in each node and each of them has the ability (allowed by the environment) to be able to move from one node to another, since each node is a completely different physical

device from the other nodes. This characteristic highlights one of the fundamental properties of the agent, mobility. Figure 3 shows the distributed environment model. For an agent to be able to move through the nodes of the environment, it must be serialized, that is, it must be able to acquire a form that allows it to move from one node to another, or from one mobile device to another. This is translated through a conversion of the agent state to a JSON-formatted character string. When this string reaches the receiving node, it is possible to instantiate the agent from its serialized state.

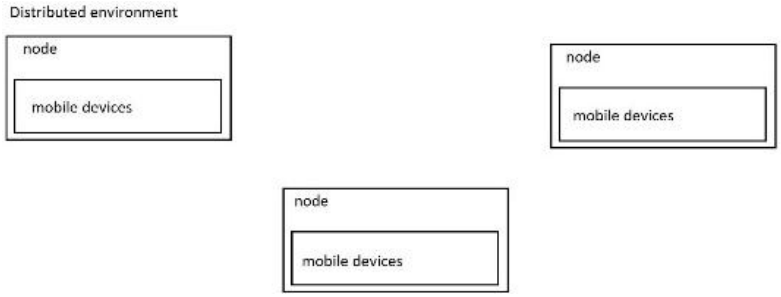

Figure 3. Graphic representation of distributed environment

3.3. Implementation

The structure of the framework is multilayer. Therefore, it is basically supported by three layers: the data layer, which consists of a set of modules that enable persistence; the service layer that integrates the API of all available web services, and on which the full functionality of the system resides. The application layer that incorporates the applications that implement the mechanisms required for the MAS design. Figure 4 shows the multilayer distribution of the framework.

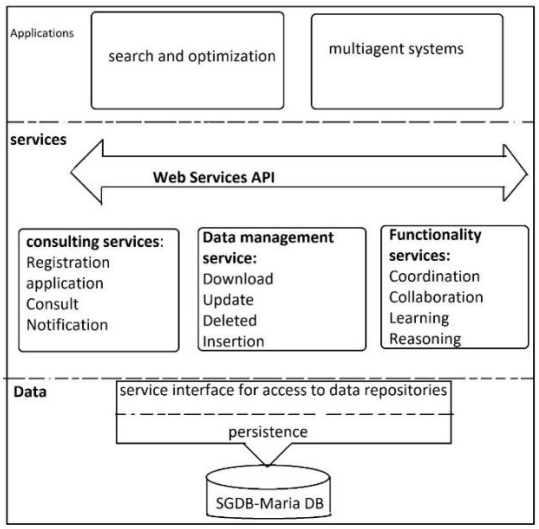

Figure 4. Layered architecture of the framework [25]

3.4. SysAge multiagent system

SysAge covers the aspects required for the creation of the following MAS: A model of a system of adaptive mobile intelligent agents that integrates in multi-agent environments for the resolution of automated processes, such as search, learning and optimization, incorporating in each agent a sufficient structure to support classification and learning techniques. This facilitates the extension of new search and control algorithms, and provides a communication interface with the environment and with other agents. To provide a mechanism that permits the mobility of agents, the environment has the facility to be distributed in various physical devices. The system supports the persistence of the knowledge acquired by each agent and the

configuration of scenarios that allow simulating, studying or analyzing the problems that involve the resolution of automated processes.

To create a project in SysAge, the user must have a user account to access the system. The user has an indeterminate number of projects. In particular, one project is a MAS. With the purpose to create a MAS it is necessary to respect a structure that has a local identifier, that is, a string made up of a set of characters that user knows which MAS is being referred to within all their available projects. Internally, the framework is responsible for generating a unique and unrepeatable *id* to identify the MAS. This *id* is made up of the object type and of a random character string encrypted in md5 to avoid duplication. The MAS must also have, as shown in Figure 5, a simulation identifier, a vector of agents, an environment identifier, and in some specific cases, the identifier of a concentrating agent and the identifier of the global knowledge acquired by the set of agents available in the MAS during the run of the resolution of a specific problem.

```
{
idProyecto:Dtr,
idSimul:Str,
Agentes:[],
AgenteConcentrador:Agente,
Conocimiento:Nodos[]
}
```

Figure 5. Structure in JSON for an MAS project

Figure 6 shows the direct association that exists between users and MAS. It can be seen that a user has a direct association by aggregation with the object of the MAS type. This is because the user can add unlimited amount of that is why multiplicity is one to many (1- *). In the case of the association of the MAS with the environment, the multiplicity is also one to many, and the type of association is by aggregation. This means that an indeterminate number of environments can be added directly to the desired MAS. This type of association reuses an environment to place it in another MAS. For the case of the association of the environment with the graph, it is important to note that the association is by composition, since one requires the other, but there is a strong dependence, that is, the environment depends completely on the graph.

This type of association is called strongly coupled, and determines that an environment cannot exist without the graph, because it is precisely in the graph, where the agents are going to move in the set of nodes that make up the graph, either to perform a search, or to determine the shortest vias from the routes. It depends mainly on the problem to be solved. In the case of the environment-agent association, it can be seen that it is by aggregation with multiplicity from one to many. It is possible to incorporate more than one agent to an environment, and it can be done from the service assign agent to environment.

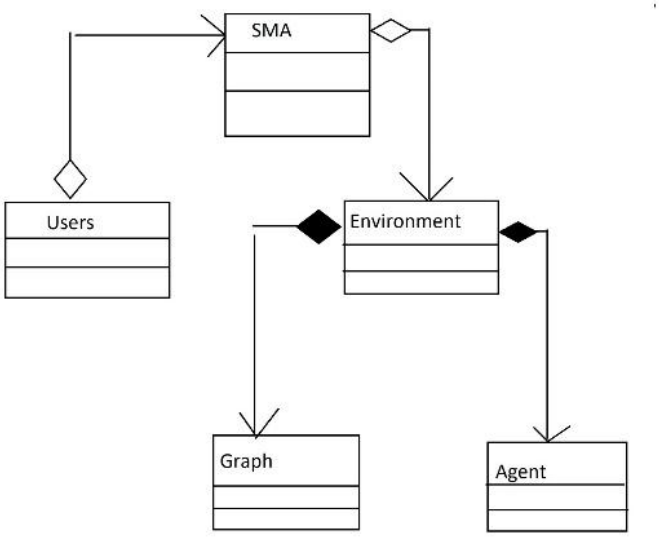

Figure 6. Models diagram for the associations between environments, agents and users

SysAge was designed from a service-oriented architecture (SOA). This avoids dependency on specific programming languages and facilitates component reuse, making it possible to extend and maintain the system efficiently. Web services are based on RESTful, a technology that uses GET, POST, PUT and DELETE requests through the HTTP protocol [26], [27]. Graphical user interfaces (GUIs) make up the front-end layer of SysAge. The interfaces have been developed with *client-side* technologies such as HTML5, Javascript, and CSS3. AJAX is used for asynchronous communication with the server. The capture forms and requests have been validated to avoid setbacks with user interaction issues.

On the server side, for the Front-end layer, PHP is used as the main language and MySQL as the database (DB) server. The web server is Apache. All the technologies in this area are run on an HP ProLiant computer, with a 2.3 Ghz 4-core Xeon processor. It has 8 MB of RAM and a 500 GB hard drive. The operating system is GNU Linux Debian 7. An environment with the above configuration is known as LAMP.

4. RESULTS

As a result, this section shows the validation of the functionality of the SySAge multiagent system framework, through some tests that were carried out for the execution of automated processes focused on solving search, classification and optimization problems.

4.1. Optimal search test

With relation to the test that was petformed on optimal search, each agent managed to find the optimal route in a weighted graph that integrated a set of penalties, where each agent (A_n) had a sensor that had been implemented as a method that receives its environment as a parameter, and an actuator that manipulates the environment through a sequence of private methods. The sensor provides each agent with the ability to know the search space, through an input interface establishing an association with the graph G, the interface is defined as a set of methods or functions. An election criterion was established based on a random decision making to specify the behavior of each agent. Two search algorithms were integrated: depth first search (DFS) and breadth first search (BFS). Each agent had to determine by their own means the algorithm to choose, considering the respective penalties and relying on the experience acquired with the passage of each iteration. Agents were employed $A_g = \{A_1, A_2, A_3\}$. Each would try to solve the problem, and the information or knowledge generated by each agent would be used autonomously, that is, the agents would not try to coordinate and collaborate to solve the search problem as a group.

For this particular case, a visual representation was associated for each agent. The associations were represented by $A_g = \{A_1 = \text{cyan}", \text{A}_2 = "green, \text{A}_3 "red"\}$. Each agent would employ different strategies and would deliver (through their respective actuator) a linked list with the corresponding nodes that make up the optimal route. Each node has a reference to the corresponding prime node.

A search space S is considered as a graph where each node represents a state with the following characteristics: an initial state $s_0 \in S$, a solution state or a set of target states $S_G \subseteq S$, a set of actions A(s)⊆ A that act on each state $S \in S$, a transition function and $f(s, a) =$ costs of carrying out actions $c(a; s) > 0$ which are treated as penalty mechanisms. The set of actions a_i, with $i = 0, \dots, n$, that reaches the solution state $s \subseteq S_G$ and that is considered as a solution to the problem. Total costs $\sum_{i=0}^{n} c(a, s_i)$, are minimized by optimal solutions.

For the search in extension or width, BFS was used, which basically consists of an algorithm that systematically examines all the nodes of a tree or graph. To perform the width route, a first in first out (FIFO) structure was used that stores all the available nodes at the i-level of the tree. During execution, it is required to store for each node, the reference of the prime node to which it is associated. Once the algorithm finds the solution node, it returns the shortest path as a subgraph containing the set of connected nodes to follow from the initial state to the final state. BFS has a time complexity of $O(|E| = O(bd)$ and a spatial complexity of $O(|V| = O(bd)$ so a considerable amount of memory is consumed if the search space is very long.

A test with 5 iterations has been carried out for a particular case where 3 agents tried to find the shortest path in a random graph of 500 nodes. The graph has certain restrictions that penalize the agent. Figure 7 shows the accumulated penalty factor (PF) against the number of iterations I(t).

In each iteration, the agents find a sub-optimal solution, remembering the route found and the penalty factor associated with that route. In the next iteration, a different search algorithm was chosen with which they may or may not improve the sub-optimal solution, remembering the best option. This behavior tends to decrease the penalty factor with the passage of iterations.

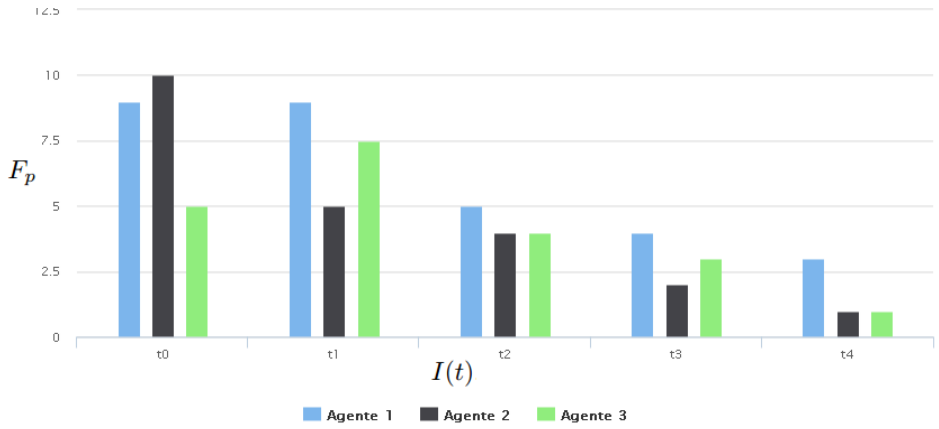

Figure 7. Reduction of the penalty factor for each agent

4.2. Ranking search test

For the classification test, the problem was to classify the resources in a distributed environment. Each node had various resources available to the agent. The entity must be able to discriminate those resources that are not of interest to it. To achieve this behavior, the agent's inference engine implemented Bayesian networks and decision trees.

For this particular case, the MAS was configured to operate with the second model implementing ID3. Algorithm 1 shows a proposed adaptation of the ID3 algorithm for solving the problem. Figure 8 shows 100 randomly dispersed resources in a simulated distributed environment. The agent was trained with a sample of 30 previously classified specimens. The colored regions show the classification made by the agent.

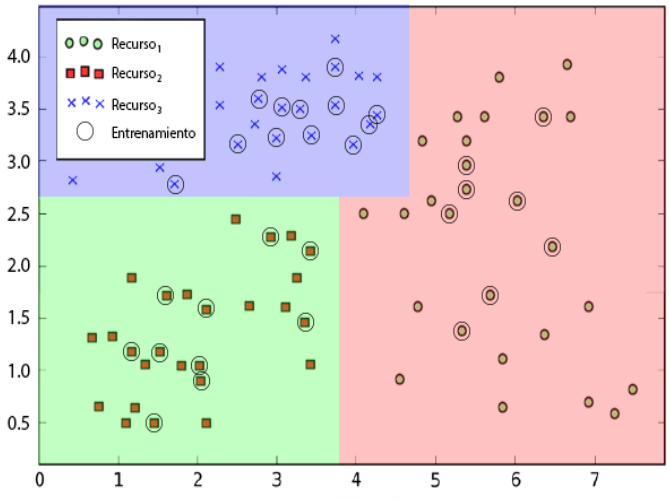

Figure 8. Classification of resources with agents that integrate Bayesian networks and decision trees as inference engine

4.3. Optimization test

For the optimization test, a problem was used that consisted in determining the interval $x, y \in [-1,1]$ the global maximum value of the function:

$$f(x,y) = sinc\left(\frac{(190x)^2}{\pi^4} + \frac{(190y)^2}{\pi^4}\right) \quad (2)$$

Where:

$$sinc(\varphi) = \begin{cases} 1, for\ \varphi = 0 \\ \frac{\sin(\varphi)}{\varphi}, for\ \varphi \neq 0 \end{cases} \quad (3)$$

To solve this problem, a MAS with 20 agents was configured and one agent was designated as the data concentrator. Each agent was given a set of 500 random values in the interval $x, y \in [-1,1]$ considering $x, y \in [-1,1]$ accuracy of 0.0001. Through Bayesian inference, the highest conditional probability of the set of values evaluated in the objective function was determined.

```
Algoritm 1 ID3
In: V ← {registros}, A ← {atributos}
Out: Tree.
1: if A = 0 or SAME_CLASS(V) then
2: C ← MAYORITYCLASS(V)
3: N ← CREATE_NODE_LEAF(C)
4: yes no
5: a_max ← MAX(∀a ∈ A GI(V, a)
6: N ← CREATE_NODE(a_max)
7: for Val_i ∈ VALUES (a_max) make
8: RegV_i ← {Val_i = a_max}
9: ADD_SON(N, ID3(RegV_I, A − a_max))
10: end for
11. end if
12 return N
```

Subsequently, the value obtained was stored in its bank of records (experience). At the end of each iteration, the agents communicate with the hub and provide it with the obtained values. The concentrating agent takes the best evaluated value in the function. After 30 iterations a global maximum was found at f (0.0054; 0.0015)=0.9982. Figure 9 shows the distribution of the agents with the best value obtained in the last iteration. The resolution of distributed optimization problems from a multiagent system notably speeds-up the process required to find the maximum or minimum point of the objective function.

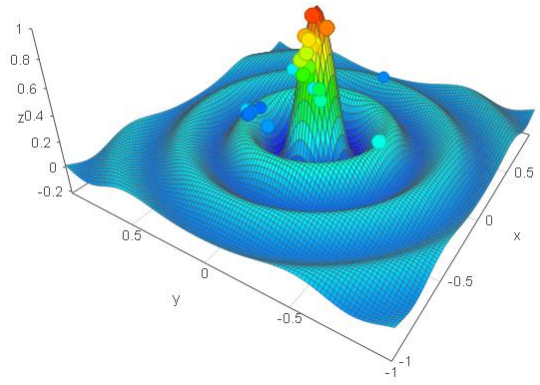

Figure 9. Agents trying to find the maximum of the objective function

5. ANALYSIS

Using probabilistic classifiers for the implementation of inference engines in a multiagent system, gives a multivariate representation of the data to be treated, which in turn shows a description of complex relationships of certain elements, also considering the handling of uncertainty.

The autonomous behavior of each agent is appreciated through the various attempts to solve the search problem, and not by focusing the information acquired individually on a single agent. Because agents employ BFS, DFS as search mechanisms, must convert the graph into a tree whose root will be the initial node, and the final node will be at some level of the tree. This implies that the agent knows a priori the graph to be worked on.

Agents display conditional behavior. Depending on the experience acquired in previous executions, agents are able to decide which search technique will be used to avoid falling into any penalty. If collaboration mechanisms are integrated, the agents will be able to share this acquired experience. If the

automated learning mechanisms are used, the values acquired in past experiences (previous iterations) can be remembered, with which it is possible to try to solve function optimization problems.

6. CONCLUSION

The model of a system of adaptive mobile intelligent agents that integrates in multi-agent environments for the resolution of automated processes, such as search, learning and optimization, must allow the incorporation, in each agent, of a structure sufficient to support technical classification and learning, which facilitates the extension of new search and control algorithms. This provides a communication interface with the environment and with other agents. To provide a mechanism permiting the mobility of the agents, the environment must have the facility to be distributed in various physical devices. The system must support the persistence of the knowledge acquired by each agent and the configuration of scenarios that simulates, studying or analyzing the problems that involve the resolution of automated processes. SysAge was presented as the proposal implemented in this research that covers these aspects required for the MAS design. System implementation and the API definition established in comparison with other frameworks, appreciates the contrast between the main aspects associated with the analyzed metrics. The available documentation, the extensibility and scalability mechanism, the availability of the source code, the tools for building and executing MAS, and the persistence mechanism that enables the reuse of environments, agents and MAS, facilitate the creation of MAS for the user. The integration of SOA as the central architecture allows heterogeneity and decoupling of modules and subsystems. Because direct dependency on a specific programming language is avoided, ease of use is reduced to consuming the services available in the system to initialize the implementation of a MAS project.

The local and distributed environments used by SysAge do not change over time; they are static. This makes it impossible to simulate or study systems that require dynamic environments. The main disadvantage lies in the fact that the agent is required to have the ability to predict the behavior of other agents. If this characteristic is implemented, it opens the way to a further stage, artificial consciousness, which involves the knowledge of their existence. The generation of decision trees, as a support for artificial learning, may become an expensive task, so the integration of an algorithm that optimizes this activity is beneficial for decision-making, in execution time or in real time, of an agent. The ID3 tree-building algorithm has runtime advantages. Moreover, the construction cost increases considerably by incorporating more than two values for a target attribute of a data collection. Through the use of algorithms based on conditional probability such as NB, the resolution of problems that require classifying a set of data is optimized, considering various values for an objective attribute. The resolution of distributed optimization problems from a multi-agent system notably speeds-up the process required to find the maximum or minimum point of the objective function. For the tests carried out in this research, each agent used the descending gradient. However, depending on the type of problem to be considered, evolutionary or swarm algorithms can be implemented; it is enough to incorporate the definition of the algorithm in the core of the agent. The framework allows this type of extensions.

ACKNOWLEDGEMENTS

We are grateful with the Instituto Politécnico Nacional, the SIP-IPN, EDD and COFAA for their support.

REFERENCES

[1] S. Buitrago, and M. Sánchez, "VMAS-Modeller: Una aplicación visual para el modelado de sistemas multi-agentes guiado por la metodología MASINA," *ReVeCom*, vol. 4, no. 1, pp. 47-58, 2017.
[2] L. A. Eras, A. Gómez, and J. Aguilar, "Sistemas multiagentes para la gestión de recursos y actividades en un aula inteligente," *IEEE Latin America Transactions*, vol. 19, no. 9, pp. 1511-1519, 2021.
[3] D. Ye, M. Zhang and A. V. Vasilakos, "A Survey of Self-Organization Mechanisms in Multiagent Systems," in *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 3, pp. 441-461, March 2017, doi: 10.1109/TSMC.2015.2504350.
[4] Y. Demazeau, "Advances in practical applications of agents, multi-agent systems, and sustainability," *Springer*, Heidelberge, 2015.
[5] S. Knorn, Z. Chen and R. H. Middleton, "Overview: Collective Control of Multiagent Systems," in *IEEE Transactions on Control of Network Systems*, vol. 3, no. 4, pp. 334-347, Dec. 2016, doi: 10.1109/TCNS.2015.2468991.
[6] F. Derakhshan and S. Yousefi, "A review on the applications of multiagent systems in wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 15, no. 5, 2019, doi: 10.1177/1550147719850767.
[7] J. Liu and J. Wu, "Multiagent robotic systems," *CRC press*, 1st Edition, p. 328, 2018, doi: 10.1201/9781315220406.

[8] J. Xie and C. C. Liu, "Multi-agent systems and their applications," *Journal of International Council on Electrical Engineering*, vol. 7, no. 1, pp. 188-197, 2017, doi: 10.1080/22348972.2017.1348890.
[9] Fei Chen; Wei Ren, *On the Control of Multi-Agent Systems: A Survey*, now, 2019.
[10] F. Michel, J. Ferber, and A. Drogoul, "Multi-agent systems and simulation: A survey from the agent commu-nity's perspective," *Multi-Agent Systems: Simulation and Application*, pp. 17-66, 2018, doi: 10.1201/9781420070248.pt1.
[11] G. Polaków, "JADE environment performance evaluation for agent-based continuous process control algorithm," *2016 21st International Conference on Methods and Models in Automation and Robotics (MMAR)*, 2016, pp. 571-576, doi: 10.1109/MMAR.2016.7575199.
[12] A. Byrski, R. Dreżewski, L. Siwik, and M. K.-Dorohinicki, "Evolutionary multi-agent systems," *The Knowledge Engineering Review*, vol. 30, no. 2, pp. 171-186, 2015, doi: 10.1017/S0269888914000289.
[13] O. Boissier, R. H. Bordini, J. F. Hübner, and A. Ricci, "Dimensions in programming multi-agent systems," *The Knowledge Engineering Review*, vol. 34, 2019, doi: 10.1017/S026988891800005X.
[14] M. F. Stabile and J. S. Sichman, "Evaluating Perception Filters in BDI Jason Agents," *2015 Brazilian Conference on Intelligent Systems (BRACIS)*, 2015, pp. 116-121, doi: 10.1109/BRACIS.2015.18.
[15] R. Calegari, G. Ciatto, V. Mascardi, and A. Omicini, "Logic-based technologies for multi-agent systems: A systematic literature review," *Autonomous Agents and Multi-Agent Systems*, vol. 35, no. 1, pp. 1-67, 2021, doi: 10.1007/s10458-020-09478-3.
[16] M. Herrera, M. P.-Hernández, A. K.-Parlikad, & J. Izquierdo," "Multi-agent systems and complex networks: Review and applications in systems engineering," *Processes*, vol. 8, no. 3, p. 312, 2020, doi: 10.3390/pr8030312.
[17] D. Weinbaum and V. Veitas, "Open ended intelligence: the individuation of intelligent agents," *Journal of Experimental & Theoretical Artificial Intelligence*, vol, 29, no. 2, pp. 371-396, 2017, doi: 10.1080/0952813X.2016.1185748.
[18] M. Jacobs and D. Selva, "A CubeSat catalog design tool for a multi-agent architecture development framework," *2015 IEEE Aerospace Conference*, 2015, pp. 1-10, doi: 10.1109/AERO.2015.7119240.
[19] D. Monett and J. E. N.-Barrientos, "Simulating the fractional reserve banking using agent-based modelling with NetLogo," *2016 Federated Conference on Computer Science and Information Systems (FedCSIS)*, 2016, pp. 1467-1470.
[20] D. M. R.-Albarrán, N. G. U.-Erazo, L. R. Guerra, B. Arellano, and S. M. Arciniegas, "Agente adaptativo para la enseñanza en ambientes inteligentes," *Revista Ibérica de Sistemas e Tecnologias de Informação*, vol. 19, pp. 694-707, 2019.
[21] G. J. Díaz-García, "Diseño e implementación de trayectorias para sistemas multi-agentes," (Bachelor's thesis, Uniandes), 2017.
[22] J. Saldanha, N. Bastos, G. Dimuro, C. Billa, and D. Adamatti, "Modelagem da teoria da identidade social em sistemas multiagente," in *WESAAC. Anais do X Workshop-Escola de Sistemas de Agentes, seus Ambientes e apliCacoes WESAAC*, 2016, pp. 187-192.
[23] J. A. C. Garofalo, J. A. V. Salazar, C. A. S. Guevara, and A. G. R. Chisag, "Inteligencia artificial, sistemas inteligentes, agentes inteligentes," *RECIMUNDO: Revista Científica de la Investigación y el Conocimiento*, vol. 4, no. 2, pp. 16-30, 2020.
[24] A. Cruzado, R. Bruges, J. Dávila, Y. Mendoza, and P. S. Sánchez, "Objeto virtual de aprendizaje para el estudio de algoritmos de búsquedas de agentes," *Investigación y Desarrollo en TIC*, vol. 6, no.1, pp. 34-39, 2015.
[25] R. Salas-Partida, E. Rodriguez-Avila, A, Ramirez-Arellano, and E, Acosta-Gonzaga., "Software Architecture Using A Decoupling Layered and Web Restful Services Approach," *Journal of Theoretical and Applied Information Technology*, vol. 98, no. 7, 2020.
[26] D. Kumaran, D. Hassabis, and J. L. McClelland, "What learning systems do intelligent agents need? Complementary learning systems theory updated," *Trends in cognitive sciences*, vol. 20, no. 7, pp. 512-534, 2016, doi: 10.1016/j.tics.2016.05.004.
[27] J. A. Iglesias-Martínez, "Modelado automático del comportamiento de agentes inteligentes," PhD in *Computer ScienceAdvisor: Araceli Sanchis and Agapito Ledezma*, Universidad Carlos III de Madrid, España, 2010.