Self-Adaptive Aided Decision-Making using a Multi-Agent System

Nicolas Brax[1], Eric Andonoff[2], Marie-Pierre Gleizes[1] and Pierre Glize[1]
[1]*IRIT, Université Paul Sabatier, 118 route de Narbonne, 31062 Toulouse, France*
[2]*IRIT, Université Toulouse 1 - Capitole, 2 rue du Doyen Gabriel Marty, 31042 Toulouse, France*
{*brax, gleizes,glize*}*@irit.fr, andonoff@univ-tlse1.fr*

Keywords: Multi-Agent System, Learning, Self-Adaptation, Decision Making

Abstract: Information required for decision-making in complex applications, such as flood forecast or maritime surveillance, can be represented using a mathematical function. However, due to the complexity of the considered applications and their dynamics, the parameters involved in the mathematical function can be hard to value *a priori*. This paper first introduces the mathematical function representing decision-making situation. It then presents a Multi-Agent System, called PaMAS (Parameter Multi-Agent System) that is able to learn values of the parameters involved in the considered situation, on the fly, autonomously, cooperatively and by self-adaptation. More precisely, it defines the behaviour of PaMAS agents and their learning strategy by self-adaptation of parameters values.

1 INTRODUCTION

Nowadays in most of complex applications, for instance, flood forecast (Georgé et al., 2003), crisis management (Bénaben et al., 2008) or maritime surveillance (Mano et al., 2010), designing software systems for assisting human operators in their decision-making activity is difficult. This difficulty is mainly due to the dynamics the considered system has to cope with: it is open, evolving and a lot of parameters have to be taken into account. In a nutshell, decision-making is dependent of the current situation which is a combination of states at a given time, *i.e.* in a given context. To enumerate and to evaluate all possible situations is very hard because there is a huge amount of combinations of states and because the context evolves during the system life. As a consequence, it is necessary to provide human operators with a system able to learn situations evaluation according to the context in which they appear in order to really help the decision-making. This paper deals with such a problematic. It advocates to use a cooperative and self-adaptive Multi-Agent System, called PaMAS (Parameter Adjustment Multi-Agent System), to learn situations evaluation according to the context. The learning is realized using a feedback coming from human operators involved in decision-making without stopping the system: the system adjusts its behaviour at runtime and the learning is realized on the fly. This paper particularly focuses on PaMAS and its learning strategy. This paper is organised as follows. Section 2 introduces the requirements and the problematic of the learning of situations evaluation and explains how this learning can be generalized to mathematical function values learning. Section 3 presents the system PaMAS we have designed and built for such a learning. More precisely it defines the behaviour of PaMAS agents and introduces its learning strategy. Section 4 shortly introduces related works while section 5 concludes the paper and gives direction for our future works.

2 LEARNING REQUIREMENTS

To be efficient for decision-making, the system must provide pertinent information enabling human operators to take decisions. For example, in the context of maritime surveillance (Mano et al., 2010), the system must indicate to decision-makers ships which behaviours are suspicious, triggering them alerts. For example, if a ship stopped several times not very far from the coast, if it is a tanker and if the weather and the sea are fine, the system can guess a transshipment (drug, gun or weapon traffic, illegal immigration) from the tanker into fast boats. As a consequence, the ship is considered suspicious by the system and it triggers an alert to the decision-maker.

These different information correspond to the states and the context of the considered situation:

stops and nature of the ship define the states of the situation while weather and sea conditions define the context. In order to identify the need to trigger an alert, the system cumulates for all states the number of occurrences of a state multiplied by the importance of it. The result is compared to a threshold defined by a decision-maker to conclude about the suspicion or not of the ship and the need to trigger an alert or not. Received alerts are then evaluated by decision-makers who feedback the result of this evaluation to the system. This feedback enables the system to learn if it did mistakes, *i.e.* it triggered useless alerts or didn't trigger useful alerts. The result the system has to find can be formulated by the equation 1 because the only information known about the states and context is that they are linked in a mathematical way.

$$\sum_{i=1}^{n} a_i^t \times x_i = r^t \quad (1)$$

In the above equation, x_i is the importance, represented as a value, of a state (or context) i the system has to learn, the coefficient a_i^t is the number of occurrences of a state (or context) i at a time t and the result r^t is the result to provide at time t. This equation is generalized as follows: the different x_i are the values to be learnt, the a_i^t are the known coefficients and r^t is the provided result.

Besides the parameters that are unknown, they are some constraints that make this problem hard to solve using regular means (neural networks, bayesian networks,...). Indeed, the number of parameters is not known beforehand and this number can change during the system execution. As a consequence, the system must be able to learn the values of the parameters on the fly.

3 PARAMETER ADJUSTMENT MULTI-AGENT SYSTEM MODELLING

In this section, the multi-agent system PaMAS designed to learn such a mathematical function is presented. First the agents in the system, their attributes, their interactions and their functions are described. Then we focus on the environment of the Multi-Agent System and its interaction with the agents.

3.1 The agents and their environment

In order to tackle the current problem, we have designed *parameter-agents* in order to learn the values of the unknown function. As the name suggests, these agents are built to manage one parameter each. Then the multi-agent system is able to compute the result of the function based on the values given by the parameter-agents. The parameter-agents were given attributes, abilities and knowledge described below.

Regarding attributes of parameter-agents, first, they know the search space within which is the value to find, this value being a real number. Second, each parameter-agent knows its associated coefficient a_i^t in the function to learn at the current time t. This value provides an efficient way to evaluate the participation of the agent in the current situation compared to the other agents. This allows the agent to determine the importance of itself in the computation of the final result. Third, a parameter-agent knows all the agents that are involved in the learning function at the current time. Fourth, the parameter-agents store a numerical value representing the learning performed during their past actions and called *histOfEvolution*, and accordingly this numerical value, they decide to adapt the value of their parameter using an Adaptive Value Tracker (AVT). Thus the value stored in *histOfEvolution* indicates the evolution of the changes made by the AVT.

Besides this attributes, the parameter-agents are given some abilities in order to modify the value of the parameter in an efficient way. Besides the common abilities of a MAS (communication skills,...), we added two skills dedicated to the learning of a function. On the one hand, the parameter-agents have the ability to use an Adaptive Value Tracker (AVT)(Lemouzy et al., 2011). This tool is a tracker that can be used by an agent in order to search a real value in a given search space. Thereby, each parameter-agent is provided with an AVT allowing him to search the value of the parameter it manages. On the other hand, the parameter-agents have the ability to compute the numerical value *histOfevolution* based on the formula 2.

$$\begin{aligned} histOfEvolution_t = Feedback \times SysDyn \times PartRes \\ + (1 - SysDyn) \times histOfEvolution_{t-1} \end{aligned} \quad (2)$$

This formula is used by the parameter-agent each time it has to modify the value of its parameter. The parameter *Feedback* is an integer with two possible values: -1 when the feedback from the environment indicates that the proposed value is too high and 1 when it indicates the value is too low. The parameter *PartRest* represents the importance of the parameter-agent. And the last parameter, *SysDyn* is an indicator of the dynamic of the system represented by a real number between 0 and 1, from an almost motionless system to a highly dynamic one. All these variables are updated in real-time while the system runs and

receives feedbacks. Finally, using this function, the parameter-agents are able to represent the latest actions they have made in a certain time window (expressed by the parameter *SysDyn*).

Finally, regarding the knowledge handled by parameter-agents, the latter know a set of *Non Cooperative Situations* in which the tuning of their values is problematic. These non cooperative situation are defined according to the AMAS Theory (Capera et al., 2003) and we consider two kinds of such situations: the conflict of the agent with itself –with its past actions and knowledge– and the inability of the agent to perform an action. These *Non Cooperative Situations* and their handling are described in section 3.2.

Given these elements, the parameter-agents are set in a common environment with which they will have interactions. The environment of PaMAS consists of the result of a mathematical function to learn, given by the general formula 1. Using the values given by the parameter-agents, PaMAS is able to compute the result of the function and proposes it to its environment. This value is then evaluated by human operators involved in surveillance, and compared with the value of the environment. If not equal, a feedback is sent from the environment to PaMAS, indicating if the value computed is superior or inferior to the wanted value. So the environment of the PaMAS can be seen as a supervisor for the learning, sending feedbacks and evaluating the resulting value computed by the parameter-agents when they are needed. This environment is dynamic and the number of parameters can change over time. Each time the function changes, PaMAS is informed of the new set of parameters involved in the current function (see equation 1). In order to communicate with the environment, there is a non-agent component within PaMAS dedicated to this task whose role is to retrieve the values estimated by each parameter-agent and compute the result of the function. Then this component sends this result and waits for possible feedbacks from the environment, and transmits it to the concerned parameter-agents. Basically, it aggregates values and transmits information. This component is called *agent-manager*.

3.2 PaMAS Functioning

The parameter-agents use the feedbacks from the environment and the background of their past actions to decide if they have to modify the value of their own parameter in the current situation and adapt themselves to their environment. Here is the algorithm of the parameter-agent during the learning process.

```
Begin
  Compute histOfEvolution_{t}
  If(sign(histOfEvolution) = sign(Feedback)) Then
    CallAVT(Feedback)
  End If
End
```

As we can see in the algorithm of the parameter-agents, the computation of the value *histOfEvolution* (see section 3.1) also allows the agent to decide if it calls its AVT to modify its parameter or not. This is the will of the agent to follow the current feedback. If the latest actions of the agent all lean to raise its value, the agent won't want to comply with a feedback indicating that the global value is too high and should be lower. Indeed, the agent interprets it as an action that is against its past situations and thus that the value which would be reached is obviously an incorrect one. This is the first Non Cooperative Situation identified and solved by the parameter-agents. Indeed, each parameter-agent tries to avoid conflict. It as to be noted that the system is dynamic and the past actions taken into account are only memorized during a certain amount of time. This way, an agent that will not comply to a feedback against its knowledge and still receiving such feedbacks will be able to change its decision and modify the value of its parameter according to the current feedback. When all the agents have taken a decision and at least one of them have made a modification of its parameter, the agent-manager of the multi-agent system computes the resulting value according to the current function to learn and propose it to its environment.

However, if none of the parameter-agents has made a modification, the agent-manager informs all of them of the situation: a feedback has been received and nothing was done. As it is a cooperative MAS, the agents within must act in order to improve the whole system. Thereby, the parameter-agents try to solve the second Non Cooperative Situation identified before, said the inability of the agents to comply with the feedback received. In order to improve the state of the system each agent cooperates with its neighbours to determine which one is the less critical, meaning the agent whose modification of the parameter is the less disturbing for the system. This can be represented by the value of *histOfEvolution*. Indeed, the more this value is close to 0, the less the agent is certain of its past actions; plus, if the agent chooses to modify its parameter, it won't go too much against them.

This way, at least one of the agent modifies the value of its parameter and the system always complies with the feedback received. The agents are thus able to identify and solve *Non Cooperative Situations* preventing the PaMAS to reach its goal, designed according to the theory of the *Adaptive Multi-Agent System*. This allows the multi-agent system to organize

itself in order to adapt itself to its environment.

4 RELATED WORKS

This section briefly describes the related works considering machine learning. When it comes to automatic learning, it exists several means like the supervised learning, the unsupervised learning and the reinforcement learning (Alpaydin, 2004).

For the supervised learning, the system is based on a knowledge database constructed by an expert in the considered domain, in order to associate the data in input with one of the classes defined in the database. If the system can not find a corresponding class, it associates these new data to a class where the data are qualified as unknown. This is a technology widely used in the domain of pattern recognition.

Concerning the unsupervised learning, the system gradually discovers the classes by himself, *i.e.* without the help of an expert, based on the data provided in input. The system analyses the data received and more precisely the groups of attributes and primitives that it is able to observer (Barlow, 1989). The user of the knowledge database has then to analyse the results in order to determine the kind of each class previously identified.

Finally, we can distinguish the reinforcement learning where the effects of an action on the environment are taken into account. Using feedbacks from the environment, the knowledge database is able to reorganize itself according to these actions and is able to modify the established classification (Sutton and Barto, 1998). This kind of learning is often applied using agent-based systems. The agents can take decision in accordance with their objectives, both collective and individual, and they can observe the direct impact on their environment. According to the benefits or the drawbacks they make out of it, each agent can they choose to modify its own classification or not.

5 CONCLUSION

The paper has presented PaMAS, a cooperative Multi-Agent System (MAS) that is able to learn by self-adaptation the parameters of a mathematical function which models the states and the context of a decision-making situation as parameters. PaMAS is able to learn the parameters values according to feedbacks of decision makers. This paper defends the idea that a mathematical function can represent the decision-making environment in most of complex systems. In this context, we have explained the running of PaMAS and how it learns, using a reinforcement learning, the values of the parameters representing the states and the context of decision-making situations.

We have experimented this claim and implemented the PaMAS multi-agent system in the context of maritime surveillance but it is not reported in this paper /cite(Brax2012). In order to fully validate our proposition, we have to run PaMAS in a large scale real environment considering hundred of parameters. We also have to establish accurate comparisons with existing learning algorithms. Our first focus are the algorithms based on neural networks due to the similarities with multi-agent systems. The comparisons will be on the number of feedbacks, the time taken and the reliability of the system. These comparisons will allow us to improve PaMAS.

REFERENCES

Alpaydin, E. (2004). *Introduction to machine learning*. MIT press.

Barlow, H. (1989). Unsupervised learning. *Neural Computation*, 1:295 – 311.

Bénaben, F., Hanachi, C., Lauras, M., Couget, P., and Chapurlat, V. (2008). A metamodel and its ontology to guide crisis characterization and its collaborative management. In *Proceedings of the 5th International Conference on Information Systems for Crisis Response and Management (ISCRAM), Washington, DC, USA, May*, pages 4–7.

Capera, D., Georgé, J., Gleizes, M., and Glize, P. (2003). The amas theory for complex problem solving based on self-organizing cooperative agents. In *Enabling Technologies Infrastructure for Collaborative Enterprises (WETICE)*.

Georgé, J., Gleizes, M., Glize, P., and Régis, C. (2003). Real-time simulation for flood forecast: an adaptive multi-agent system staff. In *Proceedings of the AISB*, volume 3, pages 109–114.

Lemouzy, S., Camps, V., and Glize, P. (2011). Principle and properties of a mas learning algorithm: a comparison with standard learning algorithms applied to implicit feeback. In *International Conference on Intelligent Agent Technology*, pages 228 – 235.

Mano, J., Georgé, J., and Gleizes, M. (2010). Adaptive multi-agent system for multi-sensor maritime surveillance. In *International Conference on Practical Applications of Agents and Multiagents Systems*, pages 285 – 290.

Sutton, R. and Barto, A. (1998). *Reinforcement Learning: an Introduction*. MIT Press – Cambridge.