Informatics in Education, 2010, Vol. 9, No. 2, 171–184

Tutoring and Multi-Agent Systems: Modeling from Experiences

Abdellah BENNANE
Centre de formation des inspecteurs de l'enseignement (CFIE)
& Centre international de pédagogie et de gestion universitaires (CIPEGU)
Université Mohamed 5 Souissi, Rabat, Morocco
e-mail: bennanea@yahoo.fr

Received: January 2010

Abstract. Tutoring systems become complex and are offering varieties of pedagogical software as course modules, exercises, simulators, systems online or offline, for single user or multi-user. This complexity motivates new forms and approaches to the design and the modelling. Studies and research in this field introduce emergent concepts that allow the tutoring system to interact efficiently with potential users, by enhancing ergonomic service, performing response time and allowing better adaptability. The introduction of concepts such as multi-agent systems (MAS) allowed web technology to improve the process of modeling and designing for distance learning, and thus offer convincing solutions. The presentation of some relevant projects that associate MAS to the Web may highlight the benefits of this association in an innovative way.

Keywords: agent, environment, stochastic games, web, reinforcement learning, multi-agent systems, tutoring.

1. Introduction

The agents in the system autonomously plan and pursue their actions and sub-goals to cooperate, and coordinate to respond flexibly and intelligently to dynamic and unpredictable situations. There is a wide range of existing application domains that are making use of the agent paradigm and develop agent-based systems, for example, in software technology, robotics, and complex systems (Taghezout *et al.*, 2009).

The major potential in multi-agent systems relate to the ability of agent systems to support personalized and informal learning. In e-learning domain, one observes a move from a personal learning environment into a space where students are taking more control on their learning, either as monolithic applications to help students manage their resources and time, or as a collection of online tools. This online learning environment has a lot of potential to support informal learning, because in a decentralized agent system, there is no need for a central authority to orchestrate collaborations and learning activities (Bentivoglio *et al.*, 2010).

The ubiquitous learning environment provides an interoperable, pervasive, and seamless learning architecture to connect, integrate, and share three major dimensions of

learning resources: learning collaborators, learning contents, and learning services (Sung, 2009).

Then, an intelligent tutoring function can support the teacher/tutor in his actions, guide students to complete their courses on the base of their performance, progress and styles of learning, towards the personalized learning processes. The concept of personalization and adaptation is fundamental to the innovation process in e-Learning (Acqua, 2009).

Adaptive learning is based on the idea of adapting learning methodologies to students' learning styles. The concept is that an individualized method of teaching will help students learn at a faster pace, more effectively, and with greater understanding. Some of the elements of adaptive learning include: monitoring student activity, interpreting the results, understanding students' requirements and preferences, and using the newly gained information to facilitate the learning process (Sung, 2009).

We underline that the implementations of pedagogical software are a subject matter oriented, that is bound to a specific content and properly built for a particular purpose. The implementation of a knowledge base requires much time and energies and cannot be used in a different field. The debate on how to sort out this problem is wide open (Rossi, 2009).

One notes the multiplication of the platforms producing online pedagogical applications. The first information to be drawn is: it is a field of research where techniques of design and modelling are tested, among which one finds multi-agents technology and artificial intelligence techniques.

The exploration and the study of these platforms can enable us to draw from the information relevant in order to elaborate work out innovating pedagogical projects.

The aim of this study is, on one hand, to present some pertinent projects and specificities of each one, summarizing the state of art of tutoring and multi-agent systems. On another hand, we present our contribution via an application that we developed in order to try some learning techniques for cognitive agents.

2. Multi-Agent Systems and Tutoring

2.1. *Characteristics of MAS*

The use of modelling techniques has been exploited by many applications in different areas namely e-Learning, systems consulting, research assistance, adaptive hypermedia and electronic commerce (e-commerce).

The multi-agent systems (MAS) are a society organized, constituted by semi autonomous agents, which interact with others, aiming to resolve collaboratively some problems, or to achieve some individuals or collectives goals. The agents may be homogeneous or heterogeneous and have common goals or not, but still maintain a degree of communication between them.

The use of MAS in software development devoted to training is promoted that the nature of teaching and learning problems is a study, research and experience field. Some methods and techniques from machine learning, for example, have been tried in this area.

The MAS requires a change in the way of thinking knowing that the scientific and technical design is based on the idea that a system is a piece consisting of subsystems identified and frozen and not on the idea that system is a population of autonomous entities in complex interactions.

Among the mains characteristics of MAS are: (1) a social organization with which groups of agents are organized within the system, depending on the role, characteristics and responsibilities; (2) cooperation between agents, who comes from sharing intermediate results to find a solution to individual targets, and contribute to the overall goals of the system; (3) coordination that allows agents to coordinate actions and behavior, which enables systems to avoid conflict situations between agents and to be consistent; (4) control that is the basic mechanism of implementing coordination in the MAS. The control parameters are two types, global and local; (5) communication between agents, the individual or societal or system (where they live) allowing the goal be achieved. This approach reproduces a sophisticated social organization of a modern society for artificial systems.

2.2. *Environment Using MAS Technologies*

The use of MAS technology in the design and modeling process of teaching and training environments has evolved spectacularly in parallel with the popularization of Internet technology. Experience has shown that a large quantity of projects has emerged in the world where information and communications technology are almost been embedded. In this section, we'll introduce some projects that will allow us to see different approaches, fields of applications and their relationship with classical architecture of tutoring systems.

1. BAGHERA (Webber, 2001). The approach of BAGHERA project reposes on the following tree principles: (1) the design of educational environments considers the collaboration between humans and artificial agents as a fundamental principle; (2) The knowledge of the learner permit, a variety of designs and their effectiveness in certain areas of practice; (3) Education is the result of an emergent complex process. It may be not the result of an isolated strategy action or achieved a goal of a single agent. The concept design is used to represent the student's knowledge in areas of practice. In essence, it considers education as a complex phenomenon can emerge from the interactions between agents having different abilities and complementary. This is a community of agents that interact with each other based on their skills to cooperate and carry out educational tasks collectively. The functional purpose is to build a platform flexible and adaptable for distance learning where each student will be assisted by three types of agents: the student companion, mediator and tutor. Similarly, the teacher will be assisted by two types of agents: interface, and assistant. The platform BAGHIRA is an open multi-agent system, where the number of agents in society increases or decreases depending on the number of connected users. For example, at one time, while a number n of students and a number m of teachers are connected, then the number of artificial agents will be $3n + 2m$. This is an important point, since the number of connections is not limited and the number of agents is not fixed in the society. The dynamic behavior of agents may be observed.

2. ALLEGRO (Viccari, 2007) is an intelligent environment that allows offering an individualized learning to the manner of CSCL (computer Supported Collaborative Learning). The use of MAS offers ALLEGRO autonomy, flexibility and adaptability. ALLEGRO is based on three theories of learning: behaviorism, cognitivism and historico-social. In the environment ALLEGRO, there are six agents: (1) tutor who will guide the learning process, decides what action to teach, how and when; (2) student model allows maintaining the learning student model; (3) interface, is the bridge between the user and the artificial agents; (4) expert manages the content specific purpose of learning-teaching; (5) diagnosis is responsible for selecting and classifying the knowledge level of the learner; (6) collaboration, at the request of the tutor agent, it seeks other learners who are interested in the same subject in order to establish synchronous or asynchronous collaborative communications.

3. Mathema (Postal, 2004). He proposes a multi-agent architecture that integrates the various formalisms to facilitate the task of teachers to develop the contents of a tutor and at the same time provide adaptability and flexibility in the presentation. The MAS Technology has been a great help in reducing the distance between the ideal and what systems can be really. In other words, the MAS allows simplifying tasks modeling and structuring through the distribution to the various models such as the domain model and student model. The proposed architecture is based on a conceptual model, called MATHEMA, which offers content oriented methodology for planning and domain layout and teaching strategies. The system is designed to teach the data structure to engineers-students at Santa Catarina University, at Brazil.

4. MACES (Multi-Agent Architecture for a Collaborative Educational System; Jacques, 2003) a system for distance learning collaborative work. Its architecture is composed of human agents (learners and tutors) and five artificial agents (diagnosis, mediation, collaboration, social and semiotics).

5. AMPLIA (Intelligent Probabilistic Multi-Agent Learning Environment) (Seixas, 2002): an additional resource for training medical students. Users (students, teachers and applications) are represented by autonomous agents that are part of a social organization based on objectives such as communication, cooperation and negotiation. AMPLIA has tree artificial agents: student, domain and mediator.

6. MAS-PLANS (Peña, 2001) an environment that intending to provide features accommodations based on learning styles to support distance learning via the Web. The environment is composed of two levels of agents: those of the higher level (agent programmable SONIA, Synthetic SMIT, Monitors and Surfing), and the lower level (didactic agent, user).

7. JADE (Java Agent framework for Distance learning Environments) (Silveira, 2002). It offers a set of resources to facilitate the development and implementation of computing environments for use as tools for learning. The agents have no mobility; they are managers of the content, exercises, examples, interactions, student model and communication model.

8. MathTutor (Pozzebon, 2004) in the case of MathTutor, the knowledge domain (data structure) is divided in two contexts (theoretical and practical), and each of these

contexts is elaborated in two courses (procedural abstraction and data abstraction). Therefore, the tutorial agent society consists of four agents, each one responsible for one of the following sub-domains: TP (theoretical procedural abstraction); PP (practical procedural abstraction); TD (theoretical data abstraction); PD (practical data abstraction). According to the internal view of the MATHEMA conceptual model, the knowledge associated with each sub-domain (TP, PP, TD and PD in the case of MathTutor) is organized into one or more curricula. Each curriculum consists of a set of pedagogical units and each pedagogical unit is associated to a set of problems. The knowledge side includes: computer architecture, programming languages, in particular Scheme language, the complexity of analysis, software engineering, among others. The formalism adopted for the domain model is a database structured, and inspired from constructivism and social knowledge theory of Vygotsky. The idea is to allow the learner to acquire and construct knowledge through interaction with the tutoring system, which is designed to enhance active participation of the learner in the learning process. To achieve this goal, the interaction is based on the cooperation of solving problems by combining learning by doing and learner by being informed.

9. ADIS (Warendorf, 1997). It is an example, designed as a pedagogical tool to provide courses on data structures. ADIS has the ability to graphically display data structures on the computer screen and the graphics allow manipulating the data structure created. Tutorial incorporating exercises, where students can learn visually the basis algorithms of data structures. ADIS is fully implemented in Java, while the tutor resides in a Java applet that is downloaded and executed on the customer machine. The student model resides on the server. The same student has access to instruction at times and locations. ADIS and MathTutor share the use of Internet browsers and a centralized student model.

10. RoboTA (Kenneth, 2002). Robot architecture is a colony of agents. It was developed by need to create virtual labs. One prototype, CyclePad, for example, allows learning thermodynamics engineering. Learners have found the system design motivating. The criterion of motivation allows the student to learn the fundamentals deeper. The authors soon realized that moving the model of simple agent to model of the agents" colony (where certain agents specialized to provide infrastructure services for the rest) allows the system to support multiple projects, adding only a small amount of complexity. One of the most important aspects was to create a system that can easily be extended. In order to develop RobotTA useful and usable for different types of applications, designers have divided the system into a central server and agents for specific application.

The core of Robot system is client-server, with a central server, the Postoffice, and multiple customers, or agents. RoboTA uses to communicate the TCP/IP to send and receive messages between the Postoffice and its agents. Each component has its own port on which it receives messages. Any communication with the user will go through the Postoffice. Users will never communicate directly with agents. A key feature of the design of Robot is that should be easy to extend the functionality of the system. This allows RoboTA to be used for a wide variety of goals. Add a new agent requires two basic steps: (1) writing a rule that allows agents to identify the appropriate response to an incoming message; (2) write an agent who handles the e-mail, and generates a result for

the user. When writing a new agent, there are three main steps: reading the contents of the message line by line; treatment of inputs and the generation content of a response to the user.

Robot is North American project, architecture of a colony of agents to support pedagogical tasks. Two key advantages provided by this architecture: (1) the sharing of communications functions through the PosteOffice and offer a RoboTA Toolkit to simplify the creation of new agents for other courses, and facilitate the manufacturing more courses; (2) the ability to host agents on different machines RoboTA has a practice interest that may be critical given the budgetary constraints of many teaching institutions.

2.3. *Innovating Ideas*

The most projects presented above are platforms. Each one attends, on one hand, to edit pedagogical modules with specialized agents allowing the author to achieve their task; on another hand, there is a generator of tutoring/pedagogical software. This approach enables the production of pedagogical software in easy way and low cost (Bennane, 2001).

From one project to other, the architecture differs, from lonely agents, agents systems, to multi-agents system. Some have functionally reactive agents and lonely; others privilege the cognitive agents that allow assisting students in their learning considering their individual characteristics. The classic architecture model of tutoring systems is not much respected, such as domain model, student model and pedagogical module.

The dynamic creation of new agents following every new connection is some of the innovating ideas that aim to personalize the interaction between customer and server. This recent field of study and research is still vast with less potential for real applications and reuse.

In the next section, we will present an application that we developed. It focuses particularly on the question of the design and implementation of cognitive agents. It shows the architecture of Information Systems Oriented Training (SIOF), some basic elements of reinforcement learning (RL) and multi-agent systems (MAS). We hope that the presentation of this application will be more useful and value added in comparison to the systems described above (Section 2.2.).

3. Information System Oriented Training

3.1. *Approach and Design*

Our design is based on two pillars. The first one is pedagogical and concerns the way with which we structure and organize the teaching environment in order to answer to the differences that exist among learners. From this perspective, we find that differentiated pedagogy can be very useful. Second one is technical and concerns the use of a learning agent, which will occupy the function of tutor in the system. At this end, we use the reinforcement learning that had demonstrated its effectiveness in the learning control.

We linked the modular approach and differentiated pedagogy (Przesmycki, 1991) to structure and organize the teaching environment. Differentiated pedagogy renews the conditions of training by opening a maximum of access for learners. In this sense, a pedagogic sequence is a succession of learning situations. In this context, the situation is no more and no less an encounter of circumstances. A situation poses a problem when it puts the subject in front of a task to be fulfilled, all procedures which it does not control. An apprenticeship is a task that poses a cognitive challenge to the learner. Then, the development of teaching situation is based on two important parameters, individualization and variety (Bennane *et al.*, 2001). Individualized teaching is a pedagogy that recognizes the student as an individual with its own representations of the teaching act. Individualized teaching or learning is adapting to both efficiency levels, the rhythm of work, reactions to failure and success, etc. While a variety of teaching situations are pedagogy which offers a range of approaches and strategies, this approach may help to solve the problem of school failure where the level of learners in the evaluation process is lacking and ignored. In general, teachers and trainers when they prepare a teaching module or course, they are constructed toward the median learner and there are no individualization. The design calls for an extra effort from teachers and trainers so that they will take into account the individual characteristics – like study level – when they prepare their teaching module (course). For this reason, a teaching situation will be a package of sub situations. How? In general, the classroom is constituted by a heterogeneous public of learners. In each class, authors find five groups (subclass): good, relatively good, medium, weak, and very weak students. From class decomposition and level learners, authors deduce the value 5. We propose this value (5) in order to move forward, knowing that the choice of this value depends on the domain teaching and the level of public heterogeneity. Every situation will contain five sub-situations by taking into account two dimensions. The first dimension concerns the heterogeneity of learners who can be regrouped in five levels. The second dimension concerns different learning strategies. In developing a course, it is advisable: (1) to solicit permanently the learner activity; (2) to treat situations first simple; (3) to lead the learner progressively to master the lesson goals by offering situations more and more of increasing complexity. By following an approach that respects the progressiveness, the gradual difficulty, and variety of pedagogical methods, a course will have all chances to bring learners to the acquisition of solid competencies.

The choices of a reinforcement learning model are due to the nature of its model. It is adapted to human learning as has been done since the work of pioneers such as Watson, Pavlov, Skinner, Bellman, etc.

Our choice is supported by the following idea: in the teaching environment, there are two agents. The first one is external to the system. It is a student which wants to learn a teaching module. This natural agent needs a tutor who can select situations adapted to the student level. Then the second agent is internal and represents the tutor (the pedagogic agent). The pedagogic agent cannot fulfill this function, unless it has the ability to learn. The RL theory and model can provide the pedagogic agent to learn through trial and error (experience). In other words, the pedagogic agent learning can be achieved only through student learning.

3.2. *Models and Interaction Scenario*

The tutoring system has four agents. Each of them takes in charge a model (environment or domain model, pedagogical module, student model, communication module).

The environment forwards a situation (state) to the pedagogical agent who performs an action. The environment responds by sending a reward (positive or negative) to the pedagogical agent following the action selection quality. The quality of the agent's action is determined by the outcome of student learning (success, failure). If the sub situation identified by (s, a) lead to learner's success, a positive reward is sent to the agent. If not, a negative reward is sent to the system (Bennane, 2006).

The current situation, action, reward and next situation are sent to student model in order to make the study course for every learner. The course is recorded if the learner achieves the pedagogical sequence. Otherwise, the learner's path is ignored by the system.

3.3. *Dynamic Generation of Pedagogical Agents*

Teaching and learning can be anywhere, synchronous or asynchronous. Our goal is that the characteristics of learners such as level of study among others must be taken into account by tutoring system via specialized agents in order to produce pedagogical adaptive courses. In this optic, the tutoring systems must have, on one hand, an agent that supports the student model. On the other hand, the dynamic generation process of pedagogical

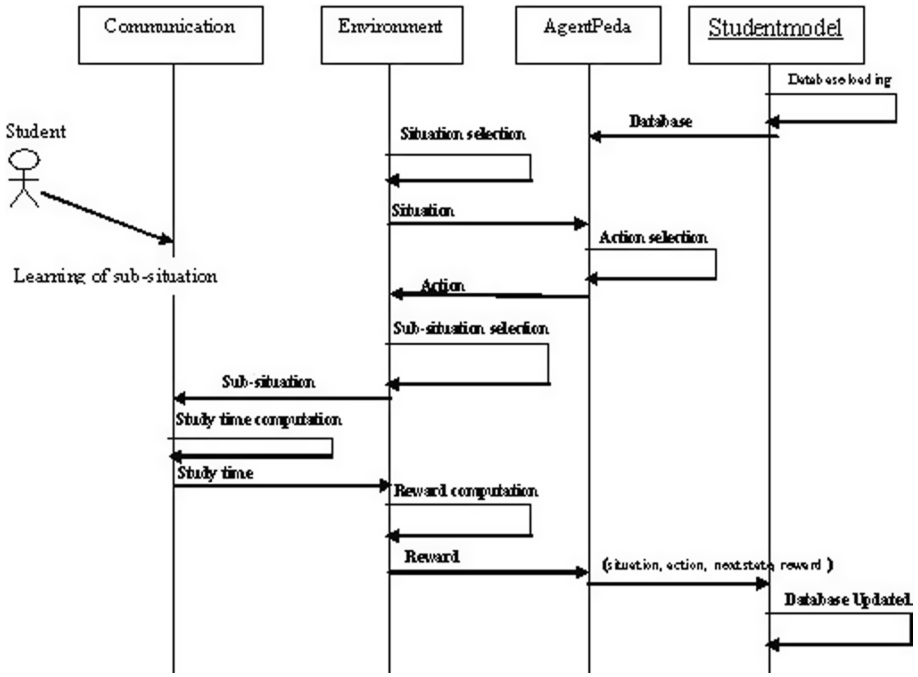

Fig. 1. Models and interactions.

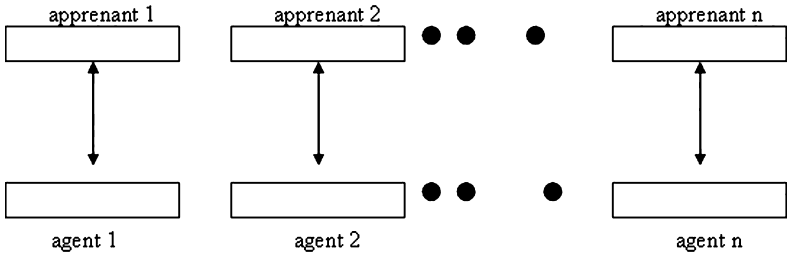

Fig. 2. Student and dynamic generation of agents.

agents starts each time that a student is connected to the system, and a new agent is created automatically. Student will be assisted by an agent who individualizes their learning courses taking into account their particular features.

For distributed tutoring systems, the dynamic generation of pedagogical agents may become the key elements attending the adaptability of the system to potential users.

4. Reinforcement Learning and Single Agent

4.1. *Methods of Resolutions*

The reinforcement learning (RL) is the study of how animals and artificial systems can learn to optimize their behavior to rewards and punishments. It was developed the RL algorithms that are closely related to the methods of dynamic programming, which is a general approach to optimal control (Sutton *et al.*, 1998). It was observed phenomena RL in psychological studies of animal behavior, in neurobiological investigations, etc. (Dayan *et al.*, 2001). One way in which animals learn complex behavior is by learning to get rewards and avoid punishments. For this type of learning, RL theory is a model of a formal calculus.

The paradigm of reinforcement learning standard, an agent is connected to an environment by perception and action. A learning agent (an animal, a robot, etc.) observes the state of the environment and then selects and executes an action. The execution of action changes the state of the ”world” and the agent acquires an immediate numerical reward, consequently. The positive earnings are called ”rewards” and negative are called ”punishment”.

At a time t, an agent receives situation/state $s_{t,}$ and choose an action a_t; the environment changes into a situation/state s_{t+1}; and agent perceives a situation s_{t+1} and gets a reward r_{t+1}. Learning is a mapping from situations to actions in order to maximize a scalar reward/reinforcement signal.

In general, tree methods are used: model-based method, model-free method and, planning and learning method (unified method).

Model-based method. It allows finding the environment model P and R, using dynamic programming techniques. Let given a database with m observations

$(s_t, a_t, s_{t+1}, r_t)\hat{I}S \times A \times R \times S$ generated on some experiences. This method is functioning on two steps:

1. Extraction of the distribution probabilities and the reinforcement of functions. The values' of the estimation of these distributions is based on the occurrences of the (s_t, a_t, s_{t+1}, r_t) in the database:

$P(s_{t+1}|s_t, a_t)$ is the probability that taking action a in state s_t will result in a transition to state s_{t+1}.

$r(s_t, a_t, s_{t+1})$ is the expected reward when transitioning from s_t to s_{t+1} by action a.

2. Usage of the dynamic programming techniques in the end to determine the optimal policy. This goal is achieved indirectly by computing the Q-values, using the Bellman optimality equation:

$$Q^*(s,a) = \sum_{s'} P^a_{ss'} \cdot (R^a_{ss'} + \gamma \cdot \max_{a'} Q^*(s', a'))\forall(s,a) \in S \times A$$

and γ a discount factor, $0 \leqslant \gamma \leqslant 1$.

Model-free method. It allows avoiding the explicit calculus of the environment model. The Monte Carlo and Temporal difference techniques are using. The Q-Learning for example, generates a suite of functions: Q: $Q1,\ Q2,\ Q3$, etc. It is an approximation of Q value, online and without a model. It's allows to construct an optimal policy directly.

Planning and learning method (Unified method)**.** Unified method is a unified view of methods that require a model of the environment such as dynamic programming and heuristic search and methods that can be used without a model such as Monte Carlo and temporal difference methods. Within a planning agent, there are at least two roles for real experience: it can be used to improve the model, and it can be used to directly improve the value function and policy using the direct reinforcement learning methods (Sutton *et al.*, 1998).

The heart of planning and learning methods is the estimation of value functions by backup operations. The difference is that, whereas planning use a simulated experience generated by model, learning methods use a real experience generated by the environment.

Both direct and indirect methods have advantages and disadvantages. Indirect methods often make fuller use of a limited amount of experience and thus achieve a better policy with fewer environmental interactions. On the other hand, direct methods are much simpler and are not affected by biases in the design of the model (Sutton *et al.*, 1998).

4.2. *Model-Based Method and Algorithm Computing Optimal Function*

Thereafter, we present the algorithm that allows computing the environment model $(P,\ R)$ and the values of optimal function Q^*.

Notation:

– $C(s, a, s')$: the number of transition from state (s) to state (s') while acting action (a);

– $C(s, a)$: the number of time the agent act action (a) at a state (s);

– $r(s, a, s')$: the sum of rewards received by agent after acting action (a) at a state (s) in order to transit to next state (s').

– q an infinitely small, and g a parameter with a value between 0 and 1.

– $q = 0.0000000001$.

– $g = 0.9$.

Algorithm:

1. Collect M interactions $(s_t, a_t, r_{t+1}, s_{t+1})$;
2. Compute $C(s, a, s')$;
3. Compute $C(s, a)$;
4. Compute $r(s, a, s')$;
5. $P(s, a, s') = C(s, a, s')/C(s, a)$;
6. $R(s, a, s') = r(s, a, s')/C(s, a, s')$;
7. Initialise $Q^*(s, a)$ to 0, $\forall(s, a) \in S \times A$
8. Repeat
9. $Delta = 0$
10. For $s = 1$ do
11. For $a = 1$ do
12. $Q^* = Q^*(s, a)$;
13. $Q^*(s, a) = \sum_{s'} P(s, a, s') \cdot \left[R(s, a, s') + \gamma \cdot \max_{a'} Q^*(s', a'))\right]$;
14. $Delta = \max(Delta, abs(Q^* - Q^*(s, a))$;
15. Next a
16. Next s
17. Until $(Delta < \theta)$

5. Reinforcement Learning (RL) and Multi-Agent Systems (MAS)

Agent-based systems technology has generated lots of excitement in recent years because of its promise as a new paradigm for conceptualizing, designing, and implementing software systems. This promise is particularly attractive to creating software that operates in environments that are distributed and open, such as the internet. Currently, the great majority of agent-based systems consist of a single agent (Sycara, 1998).

Research in MAS is concerned with the study, behavior, and construction of a collection of possibly preexisting autonomous agents that interact with each other and their environments.

A stochastic game is a tuple $(A, S, \{U_i\}_{i \in A},\ p,\ \{r_i\}_{i \in A})$ where:

$A = 1, \ldots, n$ is a set of agents;

S is a finite state space;

Ui is the finite set of agent's actions, $U = x_{i \in A} U_i$;

p: $S \times U \times S \rightarrow [0, 1]$ is the state transition probability distribution;

r_i: $S \times U \times S \rightarrow \Re$ is the agent's reward function.

State transition and rewards depend on the joint action. Then the consequence of learning is that optimal policy depends on policies of other agents.

According to tasks, one may be classifying the MAS tasks on three: (1) cooperative tasks where $r_1 = r_2 = \ldots = r_n$; (2) competitive tasks where $r_1 = -r_2$ (zero – sum games); (3) mixed tasks is a general case (general – sum games).

The RL approach in MAS may be three. The first one apply single agent reinforcement learning by ignoring the presence of other agents. It considers other agents indirectly, troughs the reward signal. The second approach guarantee convergence independently of other agents, but it is aware of learning agents. The third, an agent is adapted to other agents in order to strive for optimality (Babuska, 2006). Better results can be obtained if the agents attempt to model each agent. In this case, each agent maintains an action value function $Q(i)(s, a)$ for all states and joint action pairs. A joint action a is continually under process in the state s and a new state s' is observed, thus each agent i updates its $Q(i)(s, a)$. The first and second approaches are the simplest case that the agents learn independently of each other. Each agent treats other agents as part of environment, and does not attempt them or predict their actions (Vlassis, 2003).

This is a model of stochastic games, also called Markov Games (Tuyls *et al.*, 2005). In this approach, it is certain that the influence of an agent on all other agents can be modeled so that the Markov property still stands. This, combined with a unique solution, such as the notion of bootstrap Stackelberg equilibrium, is generally performed in the RL techniques (Könönen, 2004).

In reinforcement learning, the behavior of a single agent is specified by a policy. In games stochastic, policies are separate from each agent and the goal of each agent is to find an equilibrium policy, i.e., a policy that has the best response in comparison with "Adversaries" policies.

The RL independent agents try to optimize their behavior without any form of communication with other agents. They use only the feedback received from the environment. These independent agents can use the traditional RL algorithms developed in the stationary case for a single agent. Note that the feedback from the environment is generally dependent on a combination of actions taken by multiple agents, not only by a single agent. In this case, the Markov property no longer holds, and the problem becomes dependent and non-stationary.

In multi-agent environments, if the behavior of other agents converges, i.e., the selection distribution becomes stationary at the limit, the Q-learning updated to converge to optimal function Q* with probability one.

In many applications where real RL techniques are used in MAS, RL techniques for a single agent are applied directly. Although some assumptions that underlie these learning models are violated, methods work surprisingly and in many cases (Könönen, 2004).

6. Conclusion

MAS is a growing field of study and research and its applications are increasing in various fields, including education and training. Theoretically, the MAS is linked to both game theory and reinforcement learning, combining their model and their technique.

In the learning and training field, MAS tries to follow the same way as tutoring systems on problematic such as student modeling, the famous problem of teaching strategies and development of lower cost training software. The association of MAS and applications of Web technology have created new concerns: how users (students) should be supported in a connected multi-user system?

We have developed architecture of a tutoring system that has four agents, in addition to the dynamic creation agent at the request of any connection. We stress the pedagogical potentials of some rising ideas such as (1) the model student is centralized and can store all interactions that they are exploited by other agents such as the teaching agent and (2) the dynamic creation of new agents with every new connection in order to individualize and personalize the user interaction system.

References

Acqua, L. (2009). Key factors for an integrated, multi-learner elearning environment using the PENTHA ID model. In: *Proceedings of the 4th International LAMS Conference 2009: Opening Up Learning Design*. Sydney, LAMS Foundation, pp. 54–64. Retrieved from: http://lamsfoundation.org/lams2009sydney/papers.htm.

Appl, M. (2000). *Model-Based Reinforcement Learning in Continuous Environments*. PhD thesis, Technical University of Munich. http://www.igi.tugraz.at/ril-toolbox/papers/ContinousStateLearning/FuzzyModelBasedLearning.pdf.

Augustin, J.P., Everton, B., Vicari, R.M. (2003). *Considering Student's Emotions in Computational Educational Systems*. XIV Simpósio Brasileiro de Informática na Educação – NCE-IM/UFRJ. http://www.nce.ufrj.br/sbie2003/publicacoes/paper54.pdf.

Babuska, R., Busoniu, L. (2006). *Reinforcement Learning for Multi-Agent Systems*. http://www.dcsc.tudelft.nl/~rbabuska/transp/cabs_handout.pdf.

Bellman, R. (1957a). *Dynamic Programming*. Princeton University Press.

Bellman, R. (1957b). A Markovian decision process. *Journal of Mathematics and Mechanics*, 6, 679–684.

Bennane, A., Manderick, B., D'hondt, T. (2001). Generation of training situation and adaptive systems. In: *Proceedings of International Conference on Computer in Education* (ICCE/SchoolNet2001), 2, Seoul. http://www.icce2001.org.

Bennane, A. (2004). L'apprentissage par renforcement et la conception d'un système tuteur adaptatif ; 7ème colloque Africain sur la recherche en informatique (CARI'04). http://www.cari-info.org.

Bennane, A. (2006). Agent d'apprentissage et la gestion du processus pédagogique ; 8ème colloque Africain sur la recherche en informatique (CARI'06). http://www.cari-info.org/prog_cari.php.

Bentivoglio, C.A., Bonura, D., Cannella, V., Carletti, S., Pipitone, A., Pirrone, R., Rossi, P.G., Russo, G. (2010). Intelligent agents supporting user interactions within self regulated learning processes. *Journal of e-Learning and Knowledge Society*, 6(2), 27–36.

Dayan, P., Watkins, C. (2001). Reinforcement learning. *Encyclopedia of Cognitive Science*. http://www.gatsby.ucl.ac.uk/~dayan/papers/dw01.pdf.

Könönen, V. (2004). *Multiagent Reinforcement Learning in Markov Games: Asymmetric and Symmetric Approaches*. PhD thesis, Helsinki University of Technology, Helsinki, Finland. http://www.cis.hut.fi/kononen/

Nikos, V. (2003). *A Concise Introduction to Multiagent Systems and Distributed AI*. http://www.fdaw.unimaas.nl/.education/4.1MAS/Literature/MAS%20tutorual.pdf.

Peña, C.-I., Jose-L Marzo Josep-Lluis de la Rosa (2001). *Intelligent Agents in a Teaching and Learning Environment on the Web*. http://bcds.udg.es/papers/icalt94.pdf.

Postal, A., Pozzebon, E., Frigo, L.B., Bittencourt, G., Cardoso, J. (2004). *MATHTUTOR: A Multi-Agent Intelligent Tutoring System*. http://w3.univ-tlse1.fr/irit/soc/perso/cardoso/cardoso-aiai.pdf.

Rossi, P.G. (2009). Learning environment with elements of artifi cial intelligence. *Journal of e-Learning and Knowledge Society*, 5(1), 191–199.

Seixas, L.J., Flores, C.D., Vicari, R.M., Ladeira, M. (2002). *An Architecture for an Intelligent Learning Environment with a Constructivist Approach*.
http://www.inf.ufrgs.br/~dflores/publicacoes/SEAMED_ITS2002.pdf.

Shapley, L.S. (1953). Stochastic games. In: *Proc. of the National Academy of Sciences of the United States of America*, Vol. 39, pp. 1095–1100.

Silveira, R.A., Gomes, E.R. (2002). FIPA *Compliant Pedagogical Agents in Distributed Intelligent Learning Environments*.
http://minerva.ufpel.edu.br/~ergomes/trabalhos/E-society2002.pdf.

Sven E. Kuehne Kenneth D. Forbus; RoboTA (2002). *An Agent Colony Architecture for Supporting Education*. http://www.eecs.northwestern.edu/docs/techreports/2002_TR/nwu-cs-02-10.pdf.

Sung, J.-S. (2009). Application protocol design for collaborative learning. *International Journal of u- and e-Service, Science and Technology*, 2(4).

Sutton R., Barto, A.G. (1998). *Reinforcement Learning, A Introduction*. A Bradford Book.

Sycara, K.P. (1998). *Multiagent Systems*.
http://www.aaai.org/AITopics/assets/PDF/AIMag19-02-2-article.pdf.

Taghezout, N., Adla, A., Zaraté, P. (2009). A multi-agent framework for group decision support system: application to a boiler combustion management system (GLZ). *International Journal of Software Engineering and Its Applications*, 3(2).

Tuyls, K., Nowé, A. (2006). Evolutionary game theory and multi-agent reinforcement learning. *The Knowledge Engineering Review*, 20(01), Cambridge University Press. Printed in the United Kingdom. pp. 63–90.
http://www.cs.unimaas.nl/k.tuyls/publications/b2hd-TuylsKER05.html.

Viccari, R.M., Ovalle, D.A., Jim´enez, J.A. (2007). *Teaching/Learning Multi-Agent Environment Using Instructional Planning and Cases-Based Reasoning (CBR)*.
http://www.clei.cl/cleiej/papers/v10i1p4.pdf.

Wald (1950). *Statistical Decision Functions*. John Wiley.

Warendorf, K. (1997). *ADIS – An Animated Data Structure Intelligent Tutoring System or Putting an Interactive Tutor on the WWW*. http://www.contrib.andrew.cmu.edu/~plb/AIED97_workshop/Warendorf/Warendorf.html.

Webber, C., Bergia, L., Pesty, S., Balacheff, N. (2001). The Baghera project: a multi-agent architecture for human learning. In: *Proceedings of the Workshop Multi-Agent Architectures for Distributed Learning Environments*, AIED2001, San Antonio, TX, USA. pp. 12–17.
http://julita.usask.ca/mable/webber.pdf.

A. Bennane is professor at the Training Center of Teaching Inspectors, and member of International Center of Academics Pedagogy and Management (Faculty of Education Sciences, University Mohamed 5 Souissi), Rabat, Morocco. He is professional in applied informatics in education sciences. His recent research is e-learning, development of the teaching software and use of machine learning techniques.

Konsultavimo ir daugiaagentinės sistemos: modeliavimas iš patirties

Abdellah BENNANE

Konsultavimo sistemos tapo sudėtingos, jos siūlo didelę įvairovę pedagoginės programinės įrangos, pavyzdžiui, kursų moduliavimo, pratimų sprendimo, imitavimo, įvairias internetines sistemas, skirtos vienam ar daugiau naudotojų. Šis kompleksiškumas skatina naujų formų ir metodų projektavimą ar modeliavimą. Šioje srityje atliekami tyrimai įveda besiformuojančias sąvokas, kurios leidžia konsultavimo sistemai efektyviau sąveikauti su potencialiais naudotojais, remiantis ergonomiškomis paslaugomis, rodyti atsako laiką ir suteikiant geresnį adaptyvumą. Straipsnyje nagrinėjama, kaip sąvokų, tokių kaip daugiaagentės sistemos (angl. MAS) įvedimas, leidžia interneto technologijomis pagerinti modeliavimo ar projektavimo procesus nuotoliniame mokymesi ir tada pasiūlyti patikimesnius sprendimus. Atitinkamų projektų, kurie susieja daugiaagentines sistemas (MAS) su internetu, analizė, gali pabrėžti šio susivienijimo naudą inovatyviu būdu.