Informatics in Education, 2013, Vol. 12, No. 1, 13–27

Adaptive Educational Software by Applying Reinforcement Learning

Abdellah BENNANE
Centre de Formation des Inspecteurs de l'Enseignement (CFIE)
UM5, Rabat Morocco& Laboratoire des Systèmes de Télécommunications et Ingénierie de la Décision (LASTID), Université Ibn Tofail, Kénitra, Morocco
e-mail: bennanea@yahoo.fr

Received: December 2012

Abstract. The introduction of the intelligence in teaching software is the object of this paper. In software elaboration process, one uses some learning techniques in order to adapt the teaching software to characteristics of student. Generally, one uses the artificial intelligence techniques like reinforcement learning, Bayesian network in order to adapt the system to the environment internal and external conditions, and allow this system to interact efficiently with its potentials user. The intention is to automate and manage the pedagogical process of tutoring system, in particular the selection of the content and manner of pedagogic situations. Researchers create a pedagogic learning agent that simplifies the manual logic and supports progress and the management of the teaching process (tutor-learner) through natural interactions.

Keywords: adaptive system, tutoring system, agent, environment, natural interaction, reinforcement learning, pedagogical process.

1. Problematic

The classical conception of intelligent tutoring systems is composed of four elements, the domain model (knowledge), the student model, pedagogic module, and communication module (Wenger, 1987).

The pedagogic module is the manager of the pedagogic action undertaken in a tutoring system. The choice of situations which the system introduces to the learners is the function of the pedagogic module. Traditionally, pedagogic module is collection of rules elaborated by author or designer of the system. These rules are defined and developed manually by being based on the experience of author or designer of tutoring system. This "manual" logic is more subjective than objective, because it is linked to the author and its pedagogic experiences.

The tendency of research in this field is to replace with statistical approaches the traditional approaches based on rules established manually. It is necessary to develop probabilistic approaches and flexible models, which allow the machine to learn (Aimeur, 2004) by itself from received data. The correlation of the learning performance is based on the engine of the statistical deduction which collects information about the user behaviors for every course of individualized study and creating a distribution of likelihood for the whole of training module (Sonwalkar, 2004). The research in machine learning was interested, for a big part, to develop the algorithms which can label new data, rather systems which learn naturally in interaction with people. I like to contribute to change this focus and to develop a new type of systems that learn with the users across a natural interaction (Picard *et al.*, 2004).

To solve the problem mentioned above, one asks, is it possible to replace the "manual logic" with a "numerical logic" which can exploit the student learning route? Can it assure the adaptability of the teaching environment to learners and be done objectively? The natural interaction between learner and teaching environment can automate the teaching module. The aim is to prove this hypothesis. Several machine learning techniques were used to solve this type of problem. The automation allows to spare effort and time and to surmount the numerous problems of the adaptability of tutoring systems. The pedagogic module can be dynamically generated, only based on the interaction of learners with its teaching environment. The invested efforts were in two directions, on one hand, the conception of a database which takes into account the significant characteristics of the pedagogic act. Secondly, to explore and to try among the learning techniques, based on numerical calculations, that is more appropriate in the field of study such as the reinforcement learning techniques.

2. Elaboration of Solution

The solution rests on two pillars. The first is pedagogic and concerns the way with which author organize and structure the teaching environment in order to answer differences which exist between learners. From this perspective, author find that differentiated pedagogy can be very useful. Second pillar is technical and concerns the use of a learning agent which will occupy the function of tutor (pedagogue) in the system. To this end, the author will use the reinforcement learning model that has demonstrated its effectiveness as a learning controller.

The goal of adaptive educational systems is to create an instructionally sound and flexible environment that supports learning for students with a range of abilities, disabilities, interests, backgrounds, and other characteristics (Shute, 2011).

3. Differentiated Pedagogy: Individualization and Variety of Methods

We linked the modular approach and differentiated pedagogy (Przesmycki, 1991) to structure and organize the teaching environment. The 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. And a situation is no more and no less an encounter of circumstances. A situation poses a problem when it puts subject in front of a task to be fulfilled, all procedures which it does not control (Hoc, 1996). An apprenticeship is a task that poses a cognitive challenge to learner.

Then, the development of teaching situation is based on two important parameters, individualization and variety (Bennane, 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 is a pedagogy which offers a range of approaches and strategies. This approach can help to solve the problem of school failure where the level of learners in the evaluation process is missing or ignored. In general, teachers and trainers when they prepare a teaching module, they focused their effort around the medium learner. Then differentiation between learners and individualization are omitted. 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. For this reason, a teaching situation will be a package of sub situations. How?

In general, the learners of a class form a heterogeneous public. In each class, author find five groups (subclass): good, relatively good, medium, weak, and very weak. From class decomposition and level learners, author 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 one concerns the heterogeneity of learners who can be regrouped in five levels. The second dimension concerns different learning strategies that will be used.

In developing a learning module, 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 pedagogic methods, a teaching module will have all chances to lead to the acquisition by learner the solid competences and directly operational.

EXAMPLE 1. Figure 1 is a pedagogic sequence of ten situations, with each situation having five sub-situation levels. One asks, what is the number of course possibilities (NCP) in this network (sequence)? NCP is the following: $5^{10} = 9{,}765{,}625$. This number

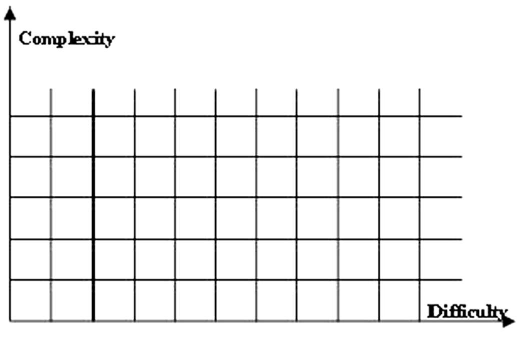

Fig. 1. Sequence: difficulty and complexity.

is very important. With support of this network that is opened to all possibilities, and not frozen, adaptation can come true. The agent learning designed and implemented offers its services to adapt the learning environment to learners and not contrary.

4. Reinforcement Learning

4.1. *Introduction*

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 (Fig. 2).

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 on several occasions 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. The positive earnings are called "rewards" and negative are called "punishment".

The reinforcement learning consists of a set of concepts and algorithms. RL is not defined by a certain class of algorithms but by the problem which it tries to solve, that is the optimum control.

RL is traditionally defined as part of a Markov decision processes (MDP). Bellman founded the theory of MDPs (Bellman, 1957a, 1957b) by the unification of previous work on the sequential analysis, functions statistical decision (Wald, 1950), and models of dynamic games for two persons (Shapley, 1953). An MDP is a quadruplet (S, A, P, R) such as S is a set of states, A is a set of actions, and P and R are the distribution of probabilities and rewards respectively.

At time t, an agent receive situation/state s_t, and chooses an action a_t; the world changes to situation/state s_{t+1}; and agent perceives situation s_{t+1} and gets reward r_{t+1}.

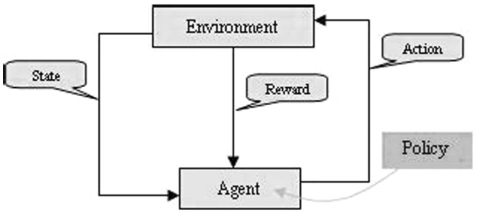

Fig. 2. Environment – agent.

Learning is a mapping from situations to actions in order to maximize a reinforcement signal (scalar reward).

The general reinforcement learning algorithm may be:

1. Initialize learner's internal state;
2. Do for a long time:
   - Observe current state (s);
   - Choose action (a) using the policy;
   - Execute action (a);
   - Let (r) be immediate reward, (s$'$) new state;
   - Update internal state based on (s, a, r, s').
3. Output a policy based on (Hayes, 2008).

4.2. *Methods of Resolutions*

In Reinforcement learning, there are three methods of solving the learning problem, **model-based** method, **model-free** method and **unified** method (**planning and learning** method).

The model-based method: 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) $\in S \times A \times R \times S$ generated on some experiences. This method is functioning on two steps:

1. Extraction of probabilities distribution and reinforcement function. The values estimation of these distributions are 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_t 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_t.
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:
   $Q^*(s, a) = \sum_{s'} P^a_{ss'}.(R^a_{ss'} + \gamma. \max_{a'} Q^*(s', a'))\forall(s, a) \in S \times A$; and γa discount factor, $0 \leqslant \gamma \leqslant 1$.

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

Planning and learning method (Unified method): is a unified view of methods that require a model of the environment, such as dynamic programming, and methods that can be used without a model, such as temporal-difference methods. We think of the former as *planning* methods and of the latter as *learning* methods. Within a planning agent, there

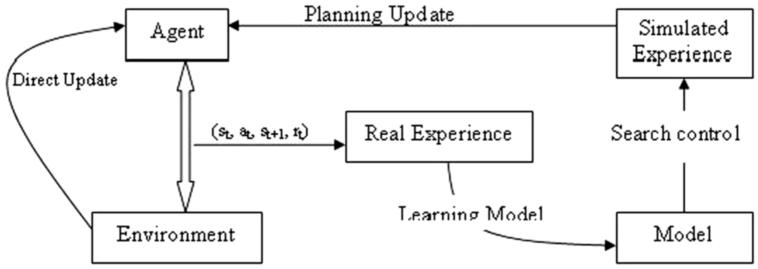

Fig. 3. The general Dyna architecture.

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; see Fig. 3).

The heart of planning and learning methods is the estimation of value functions by backup operations. The difference is that whereas planning use simulated experience generated by model, learning methods use 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 concerned by the design of the model but expensive in experiences number (Sutton, 1998).

4.3. *Interaction Scenario*

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

The choice starts from the following idea: in the teaching environment, one finds two agents. The first one is external to the system. It is a student who needs to learn a teaching module. This natural agent need 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 can't 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 (Fig. 4).

4.4. *Dynamic Generation of Pedagogic Multi-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 pedagogic 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 pedagogic

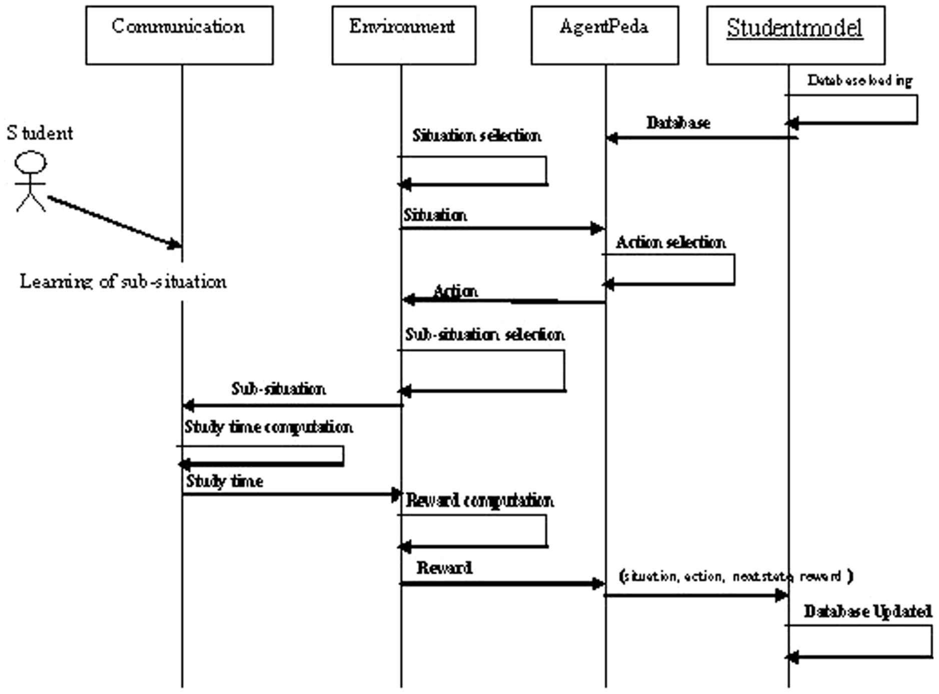

Fig. 4. Interaction scenario.

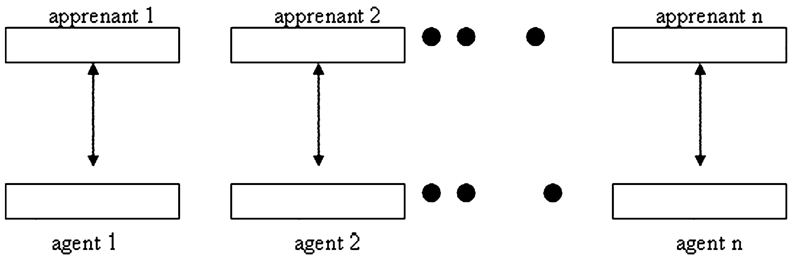

Fig. 5. Student and dynamic generation of agents.

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 (Fig. 5).

For distributed tutoring systems, the dynamic generation of pedagogical agents may become the key elements attending the adaptability of the system to potential users (Bennane, 2010).

4.5. *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, behaviour and construction of a collection of possibly pre-existing 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:

- $A = 1, \dots, n$ 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: $S \times U \times S \longrightarrow [0, 1]$ is the state transition probability distribution;
- r_i: $S \times U \times S \longrightarrow \mathcal{R}$ 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 $r_1 = r_2 = \dots . = r_n$; (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 applies 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. 2005). In this approach, it is certain that the influence of an agent on all other agents can be modelled 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 behaviour 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 behaviour 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 asingle 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 behaviour 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).

5. Student Model

How an agent can select sub situations adapted to student level? In order to answer to this question, we must first to present the student model function (Vanlehn, 1988). A student model is a qualitative representation of student knowledge for a domain, a particular subject or a competence which can take into account, fully or partially, specific aspects of student behavior (Sison *et al.*, 1998). In other words, it describes the objects and processes in spatial, temporal or causal relationship (Clancey, 1986). In this sense, the student model may be having two functions. First, it's a memory storing all transactions passing from an object to another and it is organized using the quadruplet (situation, action, next situation, reward) at least. The second function is a set of methods using dynamic resources to manage the student model. The pedagogic module selects the teaching course of student based on student behaviors. Whereas the pedagogic agent needs some data and methods in order to answer to goal task requirements. Its goal is to determine an optimal function Q^* (Sutton *et al.*, 1998) that will be the criterion of sub situation selection. It is the engine of the tutoring system adaptability. It use student model database in order to compute the optimal policy. In our case, first determine environment model, then compute optimal function value.

6. Looking for an Optimal Policy

6.1. *Model-Based Method*

In order to determine the optimal function Q^* which represent the selection criterion of student adaptive course, we have adopted the model based method because:

- With few environment interactions, one can deduce the optimal policy (Sutton *et al.*, 1998);
- State space dimension is modest;
- The exploration and exploitation phases will be separated. This separation allows easily compare the two phases according to the criteria given as "transition probabilities distribution", "reduction of time in the student learning process" etc.

The values of optimal function Q^* are computed as follows:

- collect m interactions (s_t, a_t, s_{t+1}, r_t);
- compute probabilities distribution values;
- compute reward function values;
- compute an optimal function values.

The environment model is determined, and then optimal values are updates using Q-Learning

6.2. *Probabilities Distribution and Reward Function*

According to the organization of a teaching sequence as we have defined, the successors of a given situation (s) are:

- (s) itself if the action is accomplish by failure;
- (s') if the action is achieved by success.

The reward function is defined from $S \times A \times S$ to R and noted fr. We underline the values Ra_1^{su}, Ra_1^{ec}, Ra_2^{su}, Ra_2^{ec}, Ra_3^{su}, Ra_3^{ec}, Ra_4^{su}, Ra_4^{ec}, Ra_5^{su}, Ra_5^{ec} used as indicative in Fig. 6 are unknowns.

The choice of the reward function is not evident because, in several cases, the proposed reward function allows not satisfying the given hypothesis (Beck *et al.*, 2000) in order to determine the optimal policy.

The probabilities distribution is defined from $S \times A \times S$ to [0, 1] and it is noted P. We underline that values α, χ, δ, β, and ϕ used as indicative in Fig. 6 are unknowns.

The values of two functions fr r and P are unknowns? Its will be learned by experience in order to determine the environment model.

The values of the probabilities distribution come as follow:

- $P(s,a,s') = \frac{|\{x \in LS/s_t(x)=s, a_t(x)=a, s_{t+1}(x)=s' ets' \neq s\}|}{|\{x \in LS/s_t(x)=s, a_t(x)=a\}|}$ si $s \neq s'$,
- $P(s,a,s') = \frac{|\{x \in LS/s_t(x)=s, a_t(x)=a, s_{t+1}(x)=s\}|}{|\{x \in LS/s_t(x)=s, a_t(x)=a\}|}$ si $s' = s$,
- $P(s,a,s') + p(s,a,s) = 1$

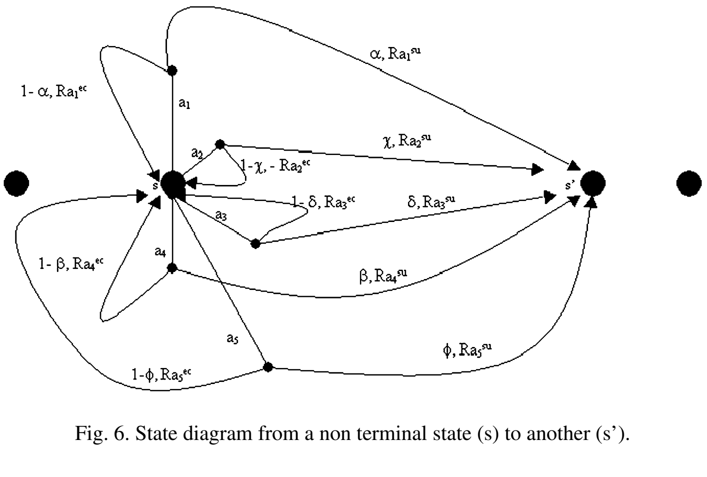

Fig. 6. State diagram from a non terminal state (s) to another (s').

The values of the reward function come as follow:

fr $(s, a, s') \cong \frac{\sum_{o\in\{x\in LS/s_t(x)=s, a_t(x)=a, s_{t+1}(x)=s'\}} r_t(o)}{|\{x\in LS/s_t(x)=s, a_t(x)=a, s_{t+1}(x)=s'\}|}$, where $r_t(o)$ is an immediate reward associated to the observation o, at time t. fr (s, a, s') is the average of immediate rewards received along the experience (Jodogne, 2007).

6.3. *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 state (s);

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

θ is an infinitely small, and γ is a parameter comprise between 0 and 1;

$\theta = 0.0000000001$;

$\gamma = 0.9$.

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 $s = 1|S|$ do
11.      For $a = 1|A|$ do
12.        $Q^* = Q^*(s, a)$;
13.        $Q^*(s, a) = \sum_{s'} P(s, a, s').[R(s, a, s') + \gamma. \max_{a'} Q^*(s', a'))]$;
14.        Delta = max (Delta, abs (Q* $-$ $Q^*(s$, a));
15.      Next $a$
16.    Next $s$
17. Until (Delta $< \theta$)
```

The update optimal function is done by Q-learning. We can only act on parameters such asα, we note that γ parameter was set when calculating the optimal function:

$Q(s, a) = Q(s, a) + \alpha.(R(s, a) + \gamma.\text{Max}_{a'}Q(s', a') - Q(s, a))$.

7. Validation of Solution

7.1. *Methodology*

We developed pedagogic software in order to verify the feasibility of our theory. The database of the pedagogic software is developed and produced by a set of teachers. It is endowed with a pedagogic agent that manages the pedagogic process between the learners and the teaching environment. In order ***to measure the agent learning evolution***, we used three assessment criteria.

- The first is the situations success probability in order to measure its evolution from a situation to another. From the learning system database, we calculate the success frequencies for every situation in order to calculate the probabilities distribution. This distribution allows us to see, on the one hand, the tendency of its curve, on the other hand, the improvement level of situation success probability. We practice this approach for two phases (exploration and exploitation) in the goal to compare their results.
- The second criterion is the study time to invest during the pedagogic sequence learning in order to measure the reduction rate of study time. Practically, we calculate the study time for every learning course and we compare the results of two phases and we deduct the reduction rate of study time.
- The third criterion is the relative gain that permits to measure the performance of the learners before and after the study of the pedagogic software in order to determine the impact of software on the learners (Bennane, 2003).

The goal is to learn an optimal sequence of actions that will displace the system from an arbitrary state to a goal state. The quality measure of actions choice of the pedagogic agent is the object of this work. If the student accomplishes situations with an interesting success probability and the pedagogic sequence with an optimal time, then one can say that the pedagogic agent assistance is achieved by success.

7.2. *Outcome*

We tried to verify three things, the study time reduction, and the improvement level of the situations success probability and finally the impact of pedagogical software on the learners.

To achieve this goal, we chose a sample of 60 students who are distributed into two groups. The first tried the pedagogic software in the exploration phase and the second tried the same software in the phase of the exploitation.

The learners sample size depends on reinforcement learning method used to calculate optimal policy. One underlines that with few environment interactions, one can deduce the optimal function where one uses model-based method.

At the exploitation phase, the pedagogic software is equipped explicitly by an optimally function that is deduce at the end of exploration phase, and implicitly by an optimal pedagogical strategy.

Table 1
Time study

Time Study	$T_{\min}$	$T_{\max}$	T_{average}	σ
Exploration	883	2500	1331.45	352.27
Exploitation	768.9	1130.38	941.71	97.92
Synthesis			$\downarrow$ 29.3 %	$\downarrow$ 72 %

Table 2
Success probability

Success probability	$P_{\min}$	$P_{\max}$	P_{average}	Synthesis
Exploration	0.508	0.556	0.538	
Exploitation	0.508	0.968	0.826	$\uparrow$ 73.6 %
Synthesis			$\uparrow$ 53.5 %	

Table 3
Relative gain

Notes	Min	Max	Average	GR
PRT	06 / 20	10 / 20	07.87 / 20	
PST	14 / 20	18 / 20	15.93 / 20	
Synthesis			$\uparrow$ 102 %	67%

We note that during this exploitation phase, we assessed the level of students before and after learning the pedagogical software to determine the relative gain. The aim is to verify the learning impact of the pedagogical software on learners (Table 1).

Between the two phases, we recorded 29.3% as reduction of study time and 72% as reduction of standard deviation (Table 2).

For success probabilities distribution criteria, we recorded a very clean improvement of 53.5% between the two phases, knowing that the average of the probabilities passed from 0.538 (in the exploration) to 0.826 (in the exploitation). In the exploitation, the curve tendency is generally upwards knowing that the success probability passed from 0.508 (1st situation) to 0.882 (last situation). This variation represents a rise (increase) of 73.6% (Table 3).

For the third criteria, the global average of the relative gain is 67%. We underline that the difference between the averages of the PRT (pretest) and the PST (post-test) is 8,06. This difference reflects a global improvement of student's level of 102%. These are good results that give a clear idea on very positive level of the pedagogic software impact on the set of its users.

8. Conclusion

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.

The recorded statistics represent good results. It is due to the intervention of the pedagogic agent while choosing the actions adapted to student level. The results of our experiences showed that the automation of the pedagogic management tasks of the tutoring system is feasible and insured. The automation permits to minimize the effort and to spare the time and to surmount the multiple problems of the tutoring systems adaptability. Our approach would exceed the approaches based on manual logic for writing rules of the teaching module. Then, the teaching module is generated dynamically based solely on the interaction of students with teaching environment.

The efforts invested in this project were in two directions, on the one hand, the conception of a knowledge base that holds in account the meaningful properties of pedagogic act, on the other hand, to look for and to try among the learning techniques based on the numeric calculations, those that are appropriated more to the study domain as the reinforcement learning techniques, etc. Authors underline that, we used an objective approach in the goal to generate an adaptive pedagogic sequence dynamically. The goal is based solely on natural interactions between the tutoring system and its potential users.

References

Aimeur, E. (2004). L'intelligence artificielle: quel avenir? L'Autre Forum, le journal des professeurs et des professeurs de l'université de Montréal, 8(2), Février, Canada.
`http://sgpum.umontreal.ca`.

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

Beck, J., Beverly, P.W, Beal, C.R (2000). Advison: a machine learning architecture for intelligent tutor construction. AAAI2000.

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. (2010). Tutoring and multi-agent systems: modeling from experiences. *Informatics in Education*, 9(2), 171–184.`http://www.mii.lt/informatics_in_education/`.

Bennane, A., D'hondt, T. (2003). Tutoring and adaptability: case study. In: *MLMTA'03*, 186–191, CSREA Press, USA.

Bennane, A., Manderick, B., D'hondt, T (2001). Generation of training situations and adaptive systems. In: *ICCE2001*, Korea. `http://www.icce2001.org/pdf/p09/BE002.pdf`.

Clancey, W. (1986). Qualitative student models. *Annual Review of Computer Science*, 1, 381–450.

Hayes, G. (2008). *Reinforcement Learning, Lecture*.
`http://www.inf.ed.ac.uk/teaching/courses/rl/slides08/`.

Hoc, J.M (1996). Supervision et contrôle de processus. La cognition en situation dynamique. Presse Universitaire de Grenoble.

Jodogne, S. (2007). Closed-loop learning of visual control policies. PhD thesis, University of Liege.
`http://www.montefiore.ulg.ac.be/~jodogne/cours/reinforcement-4.pdf`.

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

Picard, R. W., Papert, S., Bender, W., Blumberg, B., Breazeal, C., Cavallo, D., Machover, T., Resnick, M., Roy, D., Strohecker, C. (2004). Affective learning – a manifesto. *BT Technology Journal*, 22(4).

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

Shute, V.J., Zapata-Rivera, D. (2011). *Adaptive Educational Systems*. `http://myweb.fsu.edu/vshute/pdf/adaptivity.pdf`.

Sison, R., Shimura, M. (1998). Student modeling and machine learning. *International Journal of Artificial Intelligence in Education*, 9, 128–158.

Sonwalkar, N. (2004). *Adaptive Learning: A Dynamic Methodology for Effective Online Learning*. `http://idlsystems.com/media/brochures/ALS.pdf`.

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

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

Tuyls, K. *et al.* (2006). Evolutionary game theory and multi-agent reinforcement learning. *The Knowledge Engineering Review*, 20(01), 63–90. `http://www.cs.unimaas.nl/k.tuyls/publications/b2hd-TuylsKER05.html`.

VanLehn, K. (1988). Student modeling. In: Ploson and Richardson (Eds.), *Foundations of Intelligent Tutoring Systems*. Lawrence Erlbaum Associates.

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

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

Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge*. Los Altos, CA, Morgan Kaufmann Publishers, Inc.

A. Bennane is professor at the Training Center of Teaching Inspectors (Centre de Formation des Inspecteurs de l'Enseignement). He is a professional in applied informatics in education. His recent research is e-learning, teaching software and use of machine learning techniques.

Adaptyvi edukacinė programinė įranga, naudojanti mokymosi su pastiprinimu algoritmą

Abdellah BENNANE

Šis straipsnis skirtas intelektikos taikymui mokymo programinei įrangai kurti. Šios programinės įrangos plėtojimo procese pasitelkiami apsimokantys dirbtinio intelekto metodai siekiant adaptuoti mokymo programinę įrangą atsižvelgus į besimokančiojo charakteristikas. Naudojantis dirbtinio intelekto metodais – mokymosi su pastiprinimu algoritmais, Bajeso tinklais ir kitais – siekiama adaptuoti sistemą pagal vidines ir išorines aplinkos sąlygas ir leisti sistemai interaktyviai sąveikauti su potencialiu jos naudotoju. Pagrindinė idėja – automatizuoti ir valdyti pedagoginį vadovavimo sistemos procesą, tiksliau kalbant, parinkti mokymosi turinį ir valdyti pedagogines situacijas. Tyrėjai kuria pedagoginius mokymosi agentus, kurie palengvina rankiniu būdu sudaromą mokymosi proceso eigą ir palaiko mokymo proceso valdymą, pagrįstą besimokančiojo sąveika su sistema.