Informatics in Education, 2003, Vol. 2, No. 2, 223–240

An Experience Applying Reinforcement Learning in a Web-Based Adaptive and Intelligent Educational System

Ana IGLESIAS, Paloma MARTÍNEZ, Fernando FERNÁNDEZ
Universidad Carlos III de Madrid
Avda. de la Universidad 30, 28911-Leganés (Madrid) Spain
e-mail: {aiglesia, pmf, ffernand}@inf.uc3m.es

Received: March 2003

Abstract. The definition of effective pedagogical strategies for coaching and tutoring students according to their needs is one of the most important issues in Adaptive and Intelligent Educational Systems (AIES). The use of a Reinforcement Learning (RL) model allows the system to learn automatically how to teach to each student individually, only based on the acquired experience with other learners with similar characteristics, like a human tutor does. The application of this artificial intelligence technique, RL, avoids to define the teaching strategies by learning action policies that define what, when and how to teach. In this paper we study the performance of the RL model in a DataBase Design (DBD) AIES, where this performance is measured on number of students required to acquire efficient teaching strategies.

Key words: web-based adaptive and intelligent educational systems, intelligent tutoring system, reinforcement learning, curriculum sequencing.

1. Introduction

Web-based education (WBE) is currently a hot research and development area. They have two useful benefits: classroom independence and platform independence. Traditional web-based courses usually are static hypertext pages without student adaptability. However, since last ninetieths, several research teams are implementing different kinds of adaptive and intelligent systems for WBE (Brusilovsky, 1999).

Web-based Adaptive and Intelligent Educational Systems provide intelligence and student adaptability, inheriting properties from *Intelligent Tutoring Systems* (ITS) and *Adaptive Hypermedia Systems* (AHS). "Intelligent Tutoring Systems (ITSs) are computer-aided instructional systems with models of instructional content that specify *what* to teach, and teaching strategies that specify *how* to teach" (Wenger, 1987). Adaptive Hypermedia Systems adapt the content of a hypermedia page to the user's goals, knowledge, preferences and other user's information for each individual user interacting with the system (Brusilovsky, 1999).

Both, ITSs and AHS, share one of the oldest problems at educational systems: the ITS *Curriculum Sequencing*, which goal is to provide the student with most suitable

individually planned sequence of knowledge units to learn and sequence learning tasks (examples, questions, problems, etc.) to work with the AHS *Adaptive Navigation Support* (ANS) technology, which goal is to support the student in hyperspace orientation and navigation by changing the appearance of the pages and their visible links.

In *Curriculum* Sequencing, *pedagogical strategies* must specify how the content is sequenced, what kind of feedback is provided, when and how to coach, explain, summarise, give an exercise, problem, question or analogy, etc. The choice of the best strategy has been widely studied, but there exist some drawbacks in their definition (Beck, 1998). On the one hand, it is expensive to encode all of the pedagogical rules that the system requires to teach effectively. On the other hand, it is difficult to incorporate all the knowledge of the human tutor and to decide how much strategies are necessary, the differences among them, the moment to apply them, why they fail, how to solve them, etc. To overcome the difficulty derived from the definition and election of pedagogical strategies, other artificial intelligence techniques have been incorporated, like semantic nets, neural nets, and others (Murray, 1999).

In this paper, we eliminate the pedagogical strategy concept using a knowledge representation based on a Reinforcement Learning (RL) model (Kaelbling *et al.*, 1996) that allows AIESs to adapt tutoring to student's needs, sequencing the content in an optimal way based on the student's performance, lesson objectives and the relationships between course modules, avoiding to define all static and predefined pedagogical strategies for each student (Iglesias *et al.*, 2002). The RL is an artificial intelligent technique that is able to find the optimal behaviour policy for a system, based only in previous experience. In this work, an example of an DataBase Design (DBD) AIES is presented and some experiments on how the tutor learns to teach by trial and error at the same time that simulated students learn AIES's contents are explained, where the main goal is to determine how many students are necessary for the AIES convergence to the optimal policy on teaching the system's contents. This will allow to conclude if it is viable to apply this method in real tutoring systems applied on real students.

The paper is organised as follow: first, the architecture of the DBD AIES used in this work and the reinforcement learning model are described in Sections 2 and 3 respectively. In Section 4, the definition of the DBD AIES system as a reinforcement learning problem is defined. Then, different experiments using different learning parameters of the system are presented in Section 5, and finally, the main conclusions and further research of this work are given.

2. Architecture of an Adaptive Intelligent Educational System

A typical structure of an ITS, and hence, of an AIES, is composed of four well differentiated modules (Burns and Capps, 1988). The *student module* contains all important information about the student in the learning process: student knowledge, personal characteristics, historical behaviour, etc. The *interface module* facilitates the communication between the AIES and the student. The *domain module* contains all characteristics of

the knowledge to teach. Finally, the *pedagogical module* decides *what*, *how* and *when* to teach the domain module contents, taking the better pedagogical decisions according to the user needs.

2.1. *Student Model*

It is necessary, for the effectiveness of this proposal, to construct a good *student model* and to classify learners according to their critical characteristics in learning the AIES knowledge. A great variety of student models and techniques have been studied (Sison and Shimura, 1998), and any classification technique, like C5.0 (Quinlan, 1993) or Reinforcement Learning techniques (Beck, 2000), could be used to assort students according to their learning characteristics. That is to say, the learning process of all the students of a certain class will be similar.

For instance, three classes of students can be defined, depending only on two characteristics (in order to simplify the example): what kind of tasks must be used in order they learn (definition, introduction and problem) and what format the material must be shown (video, text, etc.). Table 1 shows the three classes of students, where the 100% of the *class1* students require the *definition* task and the *video* format to learn. *Class2* students require, like *class1* students, only the *definition* task, but a 95% of them learn only with the *video* format and at the other 5% learn only with *text* format. Finally, *class3* students require *definitions* and *introductions*, both with a percentage of 50%, and the *video* and *text* format with a percentage of 75% and 25% respectively.

2.2. *Domain Model*

AIES knowledge is stored in the *domain module*. The traditional knowledge structure based on topics, sub-topics, tasks, etc. could be an advantage in the pedagogical strategy in AIES. That is why we propose to use this hierarchy to define the tutor system's objectives. Fig. 1 shows a proposal of a hierarchical knowledge structure, where each topic has been divided into sub-topics, and these in others sub-topics, and so on. At the same time, each node of the tree contains sets of definitions, examples, problems, exercises, etc. in several formats (image, text, video, etc.).

Table 1
Student classes and their characteristics

Student Classes	**Tasks**	**Formats**
Class 1	Definitions	Video
Class 2	Definitions	Video (95%) & Text (5%)
Class 3	Definitions (50%) & Introductions (50%)	Video (75%) & Text (25%)

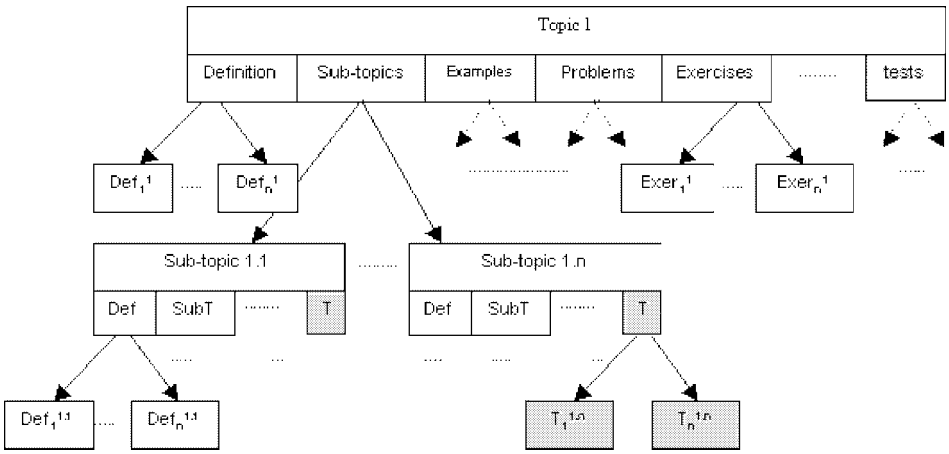

Fig. 1. Hierarchical structure of knowledge.

Database Design (DBD) domain have been proposed as example of an AIES domain. This domain has been included too in PANDORA project[1] (Castro *et al.*, 2002), whose main goal is to define methods and techniques for database development implemented in a CASE tool, useful for students and practitioners, where one of the modules of this project is concerned with tutoring student learning via web.

The knowledge model used for this system is based on the methodology proposed by Teorey (Teorey *et al.*, 1986) in which three main phases in the database design are defined: conceptual, logical and physical phases. In order to simplify the example, we focus at the Entity Relationship (E/R) model (Chen, 1976), and, exactly, on the *Conceptual Design* topic. In Fig. 2, the AIES DataBase Design knowledge of the example is presented.

In order to experiment with the convergence of the *Q-learning* algorithm in this system, we have worked with two reduced knowledge model. The *A-model* of *knowledge* (see Fig. 3) presents only three topics, but each one of them has ten tasks (*definitions, introductions and problems*) that can be shown to the student in order to teach him/her this knowledge material. In the other hand, the *B-model* of *knowledge* presented in the Fig. 4 has eleven topics, but each topic only has two tasks for their execution, to be exact only one *definition*, but in two different formats: *text* and *video*.

2.3. *Interface Model*

In the *interface model*, the AIES adapts the content of a hypermedia page to the user's goals, knowledge, preferences and other user's information for each individual user interacting with the system, changing the appearance of the pages and their visible links.

In the experiments presented at this paper, we have not used the interface model, because the system have interacted with simulated students. Now, we are building an interface model in order the real students can learn the system's contents.

[1]CASE Platform for Database development and learning via Internet. Spanish CICYT project (TIC99-0215).

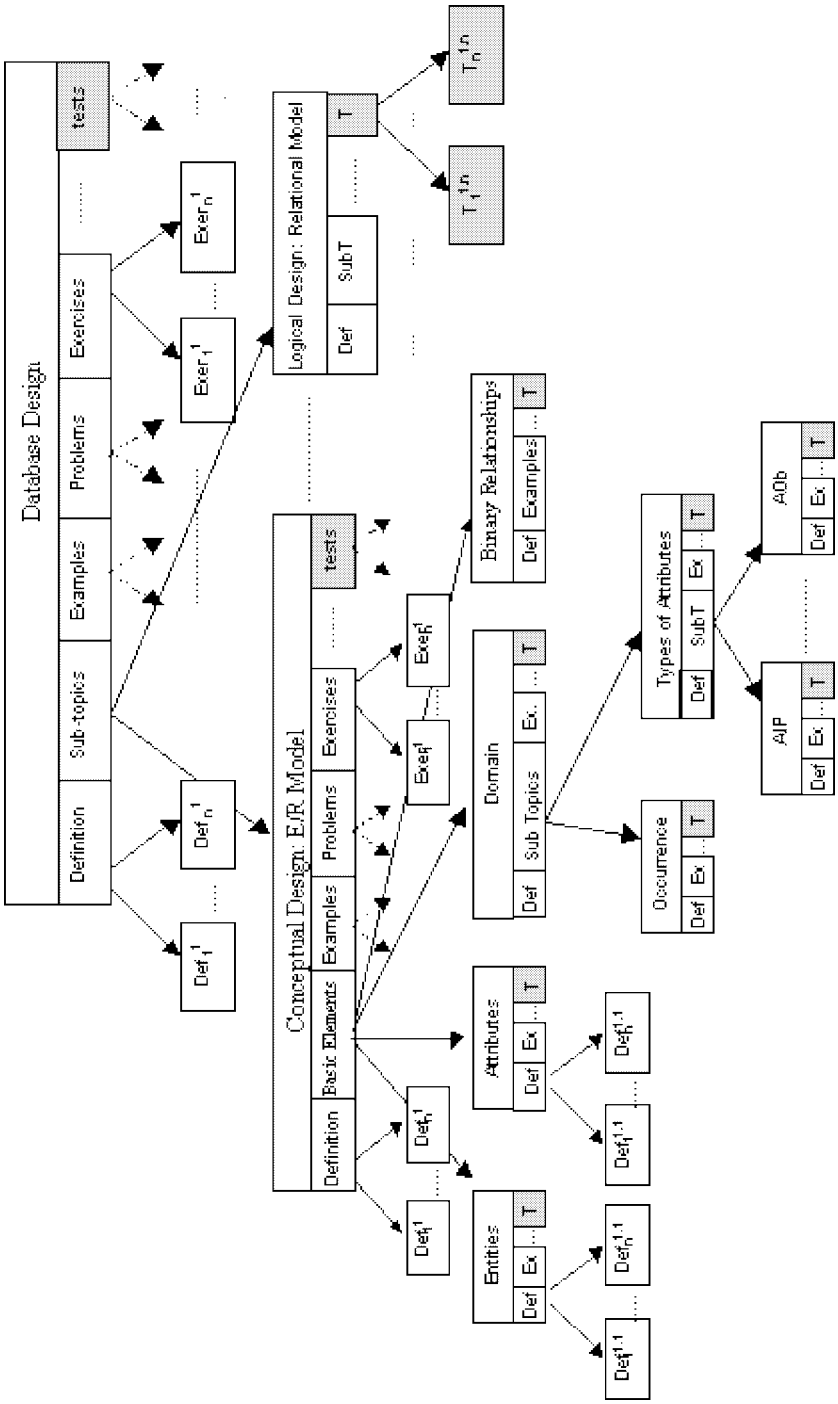

Fig. 2. AIES DataBase Design (DBD) knowledge.

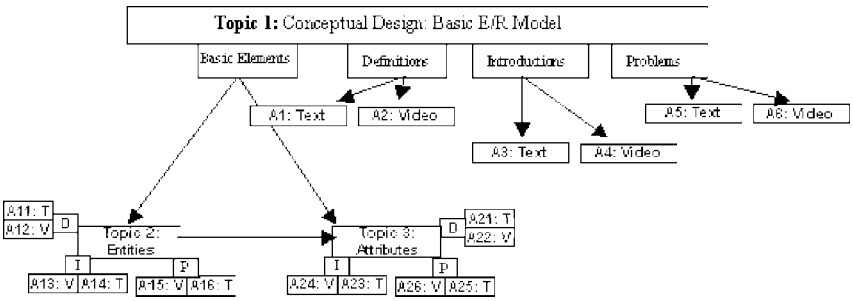

Fig. 3. A-model of knowledge.

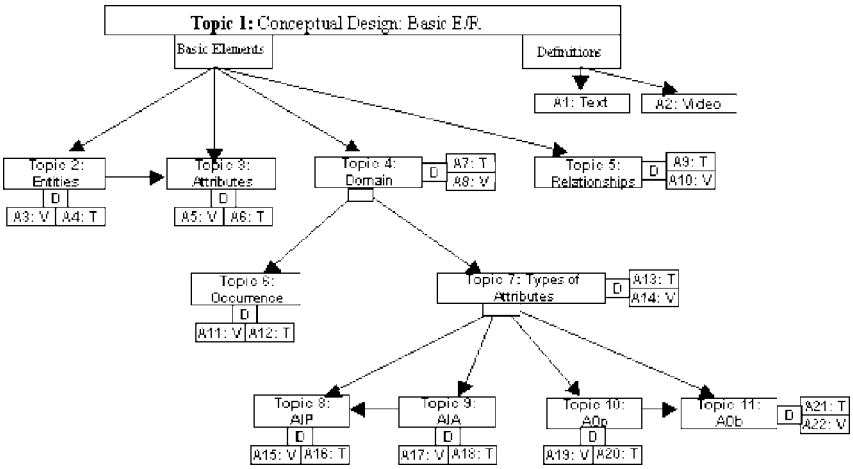

Fig. 4. B-model of knowledge.

2.4. *Pedagogical Model*

In the *pedagogical model*, the AIES finds the best way to teach the *knowledge items*, corresponding with the internal nodes of the tree (topics), to the current student. The definition of this problem as a Reinforcement Learning problem is fulfilled in the Section 4, where a guide to apply Reinforcement Learning model in Intelligent Tutoring Systems is given, defining the RL components introduced in next section.

3. Reinforcement Learning

Reinforcement Learning (RL) is defined as follows (Kaelbling *et al.*, 1996): "an agent is connected to its environment via perception and action. On each step of interaction the agent receives as input, i, some indication of the current state, s, of the environment; the

Table 2
Q-learning algorithm

***Q*-learning Algorithm**
• For each pair ($s \in S$, $a \in A$), initialise the table entry $Q(s, a)$
▷ Observe the current state, s
▷ Do forever
– Select an action, a, and execute it
– Receive the immediate reward, r
– Observe the new state, s'
– Update the table entry for $Q(s, a)$ as follows: $Q(s, a) = (1 - \alpha)Q(s, a) + \alpha \{r + \gamma \max_{a'} Q(s', a')\}$
– Set s to s'

agent then chooses an action, a, to generate as output. The action changes the state of the environment, and the value of this state transition is communicated to the agent through a scalar *reinforcement signal*, r. The agent's behaviour, B, should choose actions that tend to increase the long-run sum of values of the reinforcement signal. It can learn to do this over time by systematic trial and error, guided by a wide variety of algorithms."

3.1. *Q-learning Algorithm*

Updating the action policy can be performed by different algorithms, for instance, the Q-learning algorithm (Watkins, 1989). This algorithm allows to learn directly from the experience (systematic trial and error) with others students with similar learning characteristics. It is based on the *value-action* function, $Q(s, a)$, to define its action policy. One of the main characteristics of this algorithm is that it does not require that system executes optimal sequences to converge (Mitchell, 1997), being able to learn the Q function and, therefore, the optimal policy. This is very important for ITS, given that learning is based on the interaction with users, and this implies a high cost.

The Q-learning algorithm is described in Table 2. It requires the definition of the possible states, S, the actions that the agent can perform in the environment, A, and the rewards that it receives at any moment for the states it arrives to after applying each action, r. It dynamically generates the action-value table, $Q(s, a)$, that allows to follow a potentially optimal policy. The γ parameter controls the relative importance of future actions rewards with respect to new ones, and α parameter is the learning rate, that indicates how quickly the system learns.

4. Application of Reinforcement Learning in Adaptive and Intelligent Educational Systems

In this section, we define how the Reinforcement Learning model is applied in Adaptive Intelligent Educational Systems and how the Q-learning algorithm is adapted to the tutor domain.

4.1. *Reinforcement Learning Components*

In this section, the components of a reinforcement learning problem in the database design adaptive intelligent educational system environment are defined.

*1. Set of states (**S**).* The state is defined as the description of the student knowledge. It is represented by a vector which stores as many representative values of learner knowledge items (internal node of the knowledge tree) as is wished the student learns. In order to simplify the example, it is supposed that these values are defined in the set {0,1}. The items of the knowledge tree are enumerated in an *pre-order* way. The zero value indicates that the student does not know the item, and the one value indicates that the item has been correctly learned.

*2. Set of actions (**A**).* The actions that the tutor can execute are those that teach the ITS subject, i.e., to show the knowledge items defined in the leaves of the knowledge tree. It is allowed to show *macro-actions*, that define a set of leaves of the tree that will be shown at the same time. It is necessary to know if a leaf of the tree has been shown to a student, because it affects on the state of the system. For instance, a student could learn a topic just after teaching him/her a leaf, but he/she could understand it thanks to the fact that previously, the system has shown him/her another topic. Thus, let us suppose a system with the macro-actions allowed to teach the topic 1 (E/R model) of the reduced A-model of knowledge model (see Fig. 3) defined in Table 3. There are actions composed by only one leaf and actions composed by several leaves. Let us suppose, that the student is in a state s and the system executes the action $A1$, i.e., the system shows to the student the definition one (def1). If the student would not learnt the item, the system have to execute another action that will content the previous one, for instance, action $A7$.

3. Perception of the environment ($\boldsymbol{I}$: $\boldsymbol{S} \rightarrow \boldsymbol{S}$). This function indicates how the AIES perceives the state the student is into. In order to perceive the consequences of the execution of to evaluate the student knowledge, a given action is required. The only way an AIES could perceive the knowledge state of the student is through evaluating his/her

Table 3
Macro-actions for *topic1* of the A knowledge model

$A1$ = to show the Definition in a Text format
$A2$ = to show the Definition in a Video format
$A3$ = to show the Introduction in a Text format
$A4$ = to show the Introduction in a Video format
$A5$ = to show the Problem in a Text format
$A6$ = to show the Problem in a Video format
$A7$ = to show the $A1$+$A3$ at the same time
$A8$ = to show the $A2$+$A4$ at the same time
$A9$ = to show the $A1$+$A5$ at the same time
$A10$ = to show the $A2$+$A6$ at the same time

knowledge by tests (presented by shadow shapes in the Fig. 1 and Fig. 2). Each knowledge item in the ITS has associated a set of tests, that will allow to evaluate how much the student knows about the item. These tests will be centred in the last action executed by the AIES, assuming that other student capabilities evaluated before do not vary from the last evaluation. However, this could be false, varying them. This problem implies that an AIES could receive from the environment states with different levels of information, that could make the system not to differentiate some states from others, or perceiving two different states as equal (POMDPs: Partially Observable Markov Decision Problems), as it happens in teaching with human tutors. To avoid this problem, there will be necessary to accomplish continuous re-evaluations.

4. Reinforcement ($\boldsymbol{R}$**:** $\boldsymbol{SxA \rightarrow R}$*).* This function defines the reinforcement signals (rewards) provided by the environment. This reinforcement function supplies a maximum value upon arriving to the goals of the tutor; in conclusion, the aim of the AIES will be to maximise its long term reward. For example, following knowledge tree defined in Fig. 3 a goal for the tutor would be that the user learns the "Entities" item, receiving only a positive reinforcement signal when the component of the state vector in this item is one. It is only in this moment when the teaching process is considered finished. The application of the reinforcement signal is a key issue in the learning technique, because to decide when and how to apply the reward is crucial for the learning system. The AIES learning is a delayed positive reward problem, given that it is impossible to know the kindness of the actual state perceived immediately after the performance of an action. So, the system does not immediately know whether it has achieved its goal after executing an action, since it might be only achieved (or not) after the execution of several ones. Furthermore, a higher reinforcement signal will be applied when the student may have learnt in less time and better (positive impact on achieving the goal).

5. Value-action function ($\boldsymbol{Q}$**:** $\boldsymbol{SxA x \Pi \rightarrow R}$*).* This function estimates the usefulness of showing leaves of the tree to a student when he is in certain knowledge state. This function provides an evaluation method for the tutor action policies. Therefore, the goal of the learning process is to find the policy such that it maximises this function, i.e., to obtain the optimal value-action function.

In its learning, the AIES uses the state description (s) as input of the function approximator (Q), using this function to select a teaching action (a). After executing actions, the student changes its state to the new one perceived by the AIES (using tests). Depending on the kindness of the state reached in order to attain the goals, the system will receive a positive or null reward.

In (Iglesias *et al.*, 2002) appears an example of the learning to teach process of an Database Design ITS with the *Q-learning* algorithm.

4.2. *Adaptation of the Q-learning Algorithm*

In this section, how the Q-learning algorithm is adapted to the tutor domain is explained. In Table 4 a comparison of the theoretical Q-learning algorithm and the adapted Q-learning algorithm to de AIES domain is presented.

Table 4
Adaptation of the Q-learning algorithm to AIES domain

	Q-learning Algorithm			Q-learning adapted to AIES domain
•	For each pair ($s \in S$, $a \in A$), initialise the table entry $Q(s, a)$	→	•	For each pair ($s \in S$, $a \in A$), initialise the table entry $Q(s, a)$
▷	Observe the current state, s	→	▷	Test the current student knowledge, s'.
▷	Do forever	→	▷	Do forever
	– Select an action, a, following an explotation strategy, and execute it	→		– Select a set of knowledge tree leaves, a, to show to the student.
	– Receive the immediate reward, r	→		– Receive the immediate reward, r. A positive reward is received when the AIES goal is achieved. A null reward is obtained in any other case.
	– Observe the new state, s'	→		– Test the current student knowledge, s'.
	– Update the table entry for $Q(s, a)$ as follows: $Q(s, a) = (1 - \alpha)Q(s, a) + \alpha \{r + \gamma \max_{a'} Q(s', a')\}$	→		– Update the table entry for $Q(s, a)$, that estimates the usefulness of executing the a action when the student is in a particular knowledge state: $Q(s, a) = (1-\alpha)Q(s, a)+ \alpha \{r + \gamma \max_{a'} Q(s', a')\}$
	– Set s to s'	→		– Let us s the current student knowledge state, s'.

5. Experiments

In this section we present the experiments carried out on the AIES learning process. First, the learning variables are explained and finally, the experimental results are discussed setting our attention specially in how these variables affect the system learning convergence.

5.1. *Learning Variables*

It is very important to define the learning variables of our experiments and how they can vary. We have chosen four learning parameters to study:

- Domain size: In the experiments, we are going to use the A- and B-models of knowledge described in Fig. 3 and Fig. 4, respectively, and explained at Section 2.2. The A-domain model has only three topics, but thirty actions to execute (tasks) and the B-domain model has eleven topics, but only twenty-two tasks.
- Determinism of students learning capability: When using the *Q-learning* algorithm in a given domain, it is very important to define if the execution of an action when the agent is in a given state, does or not always arrive to the same state. When it does not achieve always the same state is because of noise associated to perception and actions. This implies that the reward of executing an action in a given state can be different depending on whether the next observed state is a goal state or not. In

Adaptive and Intelligent Educational System, like at Intelligent Tutoring Systems, there exists indeterminism in actions, because their execution could arrive to different final states due, for instance, to the different students learning characteristics (the student can learn or not the item, with different results on the tests). In the experiments we have used the three student classes defined in Section 2.1, each of them with different determinism level.

- Exploration versus exploitation: Exploration implies to try new alternatives, while exploitation implies to use pre-acquired knowledge. It is important to define the exploration/exploitation system strategy because it influences the learning rate and its quality. In our experiments, we are going to use de *e-greedy* exploration policy, where the e parameter indicates the probability of choosing the action with the best value of $Q(s, a)$ being at the s state at this moment. We are going to vary the e parameter from the value 1 (when the system choose the next action to execute randomly) to the value 0.1 (when the system choose the next action trusting in the Q values obtained by the previous experience).
- The learning rate (α): This parameter indicates how quickly the system learns. We are going to vary the α parameter of the Q update function ($Q(s, a) = (1 - \alpha)Q(s, a) + \alpha\{r + \gamma \max_{a'} Q(s', a')\}$) at *Q-learning* algorithm from the value 0.9 (when the system quickly learns) to the value 0.1 (when the system learns to slowly teach).

5.2. *Experimental Results*

In this section, the results of the experimentation varying the learning parameters of the system described in previous section are commented. All of these experiments are going to show how these parameters affect the convergence of the AIES learning, dealing with the goal of the AIES: to minimise the time spent in teaching process, using less actions as possible. Therefore, the goal of these experiments is to analyse the number of students needed to use the system until the AIES learn to teach optimally.

We divide the experiments in two groups: the experiments with the *A-model of knowledge*, and the experiments with the B domain model:

5.2.1. *A-model of Knowledge*

In the first set of experiments (see Figs. 5, 6 and 7), we have used the A-model of knowledge shown in Fig. 3 and we have fixed the e parameter of the *e-greedy* exploration policy to 1 (the system randomly choose the next action to execute). The goal is to obtain the learning curve of the AIES in order to study how many students must work with the AIES in order the AIES learns an optimal pedagogical strategy on teaching policy for each class of students.

- Fig. 5 shows learning evolution of the system interactions with *class1* students (see Table 2). We can see different learning evolution depending on learning rate α. We can observe that when de value of α is high, the curve decreases faster than when

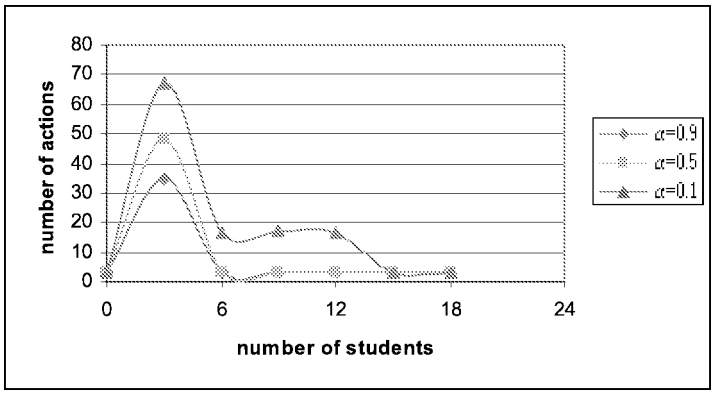

Fig. 5. AIES learning curve for ***A*-model of knowledge** students and ***1-greedy*** exploration policy interacting **class1 students**.

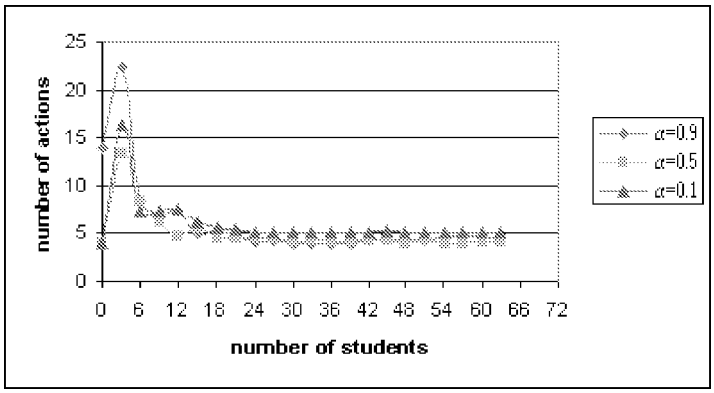

Fig. 6. AIES learning curve for *A*-**model of knowledge** students and ***1-greedy*** exploration policy interacting **class2 students**.

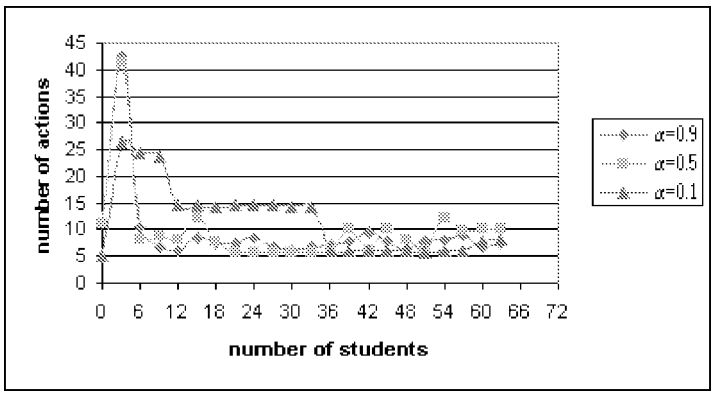

Fig. 7. AIES learning curve for *A*-**model of knowledge** students and ***1-greedy*** exploration policy interacting **class3 students**.

de value of α is low, because the system learns slower. Furthermore, the figure shows that the average number of actions needed to arrive to the AIES goal is 3, and that the system learns to teach optimally all topics when only six students have interacted with de tutor if $\alpha > 0.5$. When $\alpha = 0.1$, the system needs 15 *class1* students to effectively teach.

- In Fig. 6 the AIES learning evolution for interactions with *class2* students (see Table 2) is represented. From this figure, we can affirm that the best policy for this kind of student needs to show an average of four tasks, and when the learning rate (α) is equal to 0.1, the system learns slowly. When the system has interacted with 63 students, it has not learn the best policy.
- In Fig. 7, it is represented the system learning evolution when the tutor is interacting with students of *class3* (see Table 2). It is shown that the actions for these students have a high level of indeterminism, and this is why the curve has a lot of "pikes", that represent changes at the action policy of the system. Moreover, we can observe how the system learns more constantly when $\alpha = 0.1$, obtaining the optimal policy when the system needs six actions to arrive the goal with 36 students.

Next, we have modified the e parameter of the *e-greedy* exploration policy, and we have used the reduced A-model of knowledge in Fig. 3.

- In Fig. 8 we present the learning evolution of the system, where *class3* students interact with the tutor. We have fixed the learning rate parameter (α) to 0.9 (the system learns quickly). In this figure we can observe that when $e = 0.5$, the tutor keeps in a local minimum (of eight actions), because it does not explore enough, and that is why is does not reach the optimal policy (six actions) when 72 students have interacted with the system.
- In Fig. 9, a comparison of the learning curve when the three classes of students interact is presented. We have fixed in this experiment the parameters $e = 1$(randomly) and $\alpha = 0.9$ (learns quickly). Seeing the graphic, we can conclude that the tutor learns slower when indeterminism in actions exists.

5.2.2. *B-model of Knowledge*

In the second set of experiments, we have used the reduced B-model of knowledge shown in Fig. 4 and we have fixed the e parameter of the *e-greedy* exploration policy to 1, as well as the previous experiment. Due to students usually implies non-determinism in actions, in the following experiments, we have used only class2 and class3 students.

- In Fig. 10, *class2* students have interacted with our system. We can compare the learning evolution of Fig. 10 with the Fig. 5, because they are similar: when the value of α is high, the curve decreases faster than when de value of α is low, because the system learns slower. Furthermore, the average number of actions needed to arrive to the AIES goal is around 30, and the system learns to teach optimally all topics when only 18 students have interacted with de tutor if $\alpha > 0.5$. When the system have interacted with 18 students and the learning rate (α) is 0.1, the

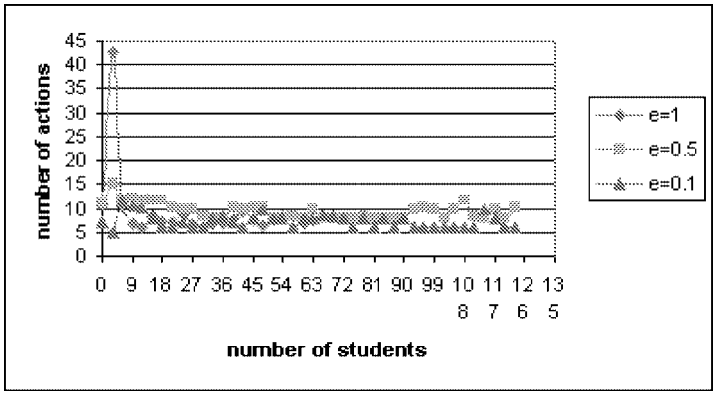

Fig. 8. AIES learning curve for A-**model of knowledge** students and $\alpha = 0.9$. Curve varying the e parameter of the *1-greedy* exploration policy.

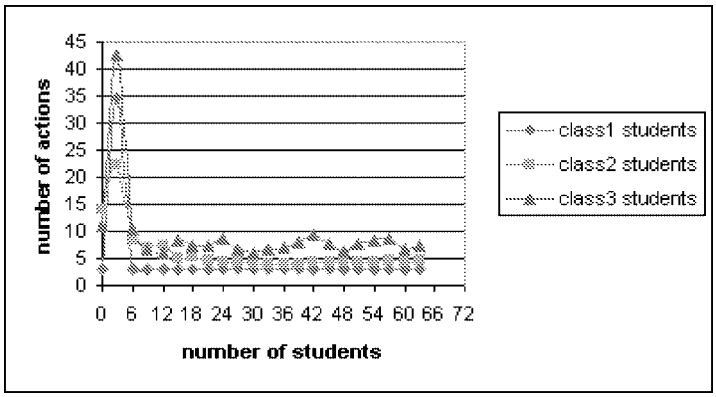

Fig. 9. AIES learning curve for A-**model of knowledge** students $\alpha = 0.9$ and $e = 1$. Comparison of the indeterminism in actions for different students.

systems needs to teach all the topics executing approximately 60 tasks, that is not a very good teaching policy.

- In Fig. 11, *class3* students have interacted with the AIES, and, again, we have modified the α parameter for the learning evolution study. In this figure we can clearly see how the learning rate affects to the system learning: we observe that the tutor learns more quickly when α is close to 1 and the tutor learns slower when the parameter is close to zero. In this way, the learning curve when $\alpha = 0.9$ converges to more or less 30 actions when 12 students have interacted with the system; when $\alpha = 0.5$ the curve converges to the same value when 45 students have interacted; and when $\alpha = 0.1$ the evolution curve have not yet converge when 66 students have interacted with the AIES.

Then, we have modified the e parameter of the *e-greedy* exploration policy, where we have used the B-model of knowledge in Fig. 4. In Fig. 12 we present the learning evolution of the system, where *class3* students interact with the tutor and we have fixed

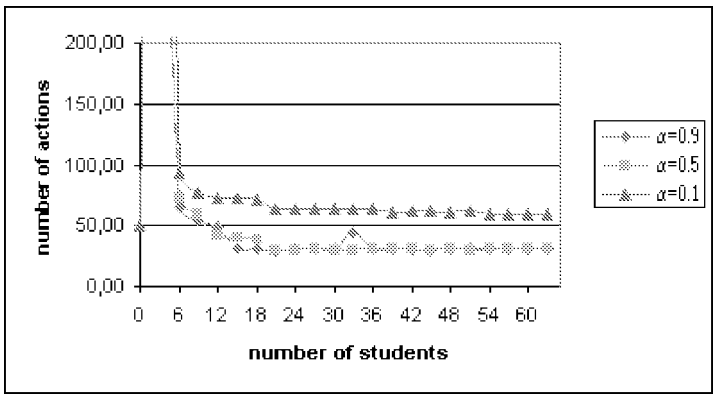

Fig. 10. AIES learning curve for B-**model of knowledge** students and ***1-greedy*** exploration policy interacting **class2 students**.

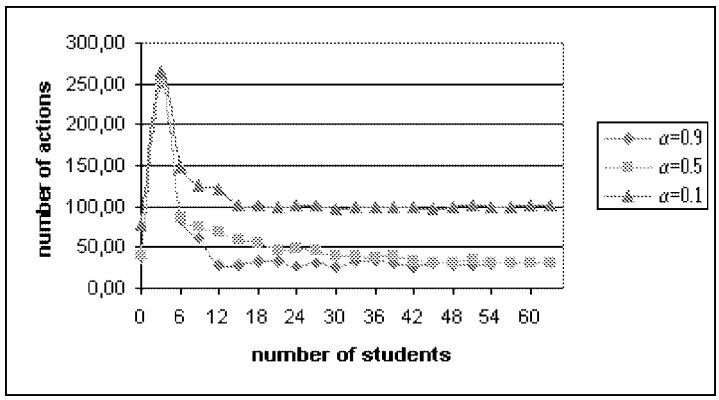

Fig. 11. AIES learning curve for B-**model of knowledge** students and ***1-greedy*** exploration policy interacting **class3 students**.

the learning rate parameter (α) to 0.9. We can see at this figure that the tutor keeps in local minimums when the exploration is not complete (when the e parameter is different that 1 (completely random)): when $e = 0.5$, the tutor keeps on the average of 70 actions; when $e = 0.5$, the tutor keeps on 100 actions, but at the 69 students, the system changes its action policy and decrease to 78 actions; finally, when the tutor randomly chooses the action, the system achieves the optimal policy of around 30 actions.

To conclude, we have done experiments comparing the learning curve when class2 and class3 students interact with the system. For this experiment, we have fixed the parameters $e = 1$(randomly) and $\alpha = 0.9$ (learns quickly). In Fig. 13 we can see how the two curves converge at the same time.

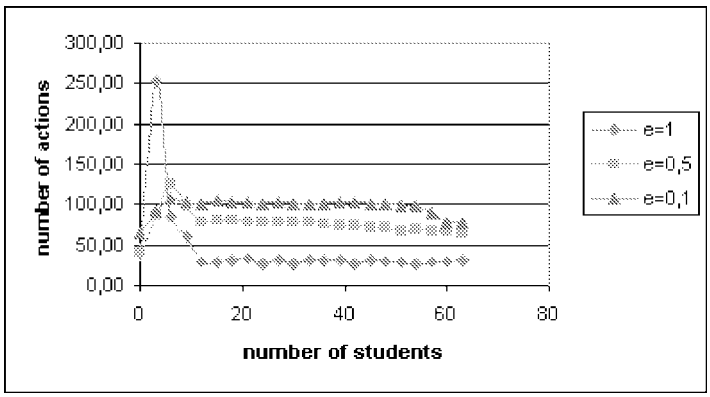

Fig. 12. AIES learning curve for B-**model of knowledge class3** students and $\alpha = 0.9$. Curve varying the e parameter of the *1-greedy* exploration policy.

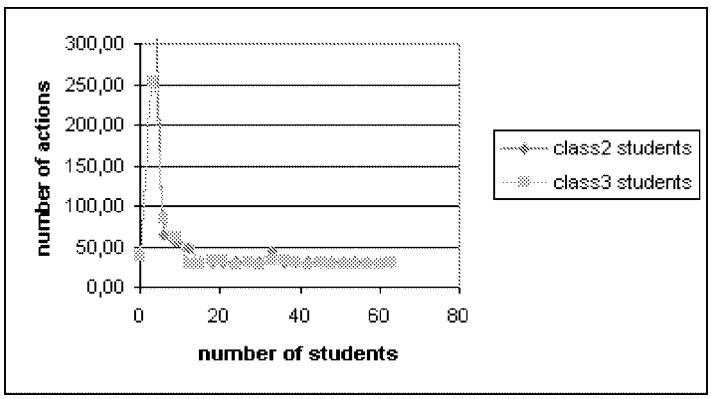

Fig. 13. AIES learning curve for B-**model of knowledge** students $\alpha = 0.9$ and $e = 1$. Comparison of the indeterminism in actions for different students.

6. Conclusions and Further Research

In this paper we propose the use of a Reinforcement Learning model that avoids the problems derived from construction of different pedagogical strategies for each student in adaptive intelligent educational systems. This model consists on defining the pedagogical module of any AIES (domain independent) as an action policy learned by trial and error (as human tutors do), applying the reinforcement learning model to the pedagogical module of the Adaptive Intelligent and Educational System. The use of reinforcement learning techniques provides many advantages on the techniques of traditional dynamic planning. First, it avoids the cost associated to rules construction and heuristic creation for each teaching strategy adapting to the needs of each student. Second, it allows the system to individually adapt to the student in real time, based only on previous information of interactions with other students with similar characteristics. Third, the system is able to adapt to the user not only deciding what to show, or when to show a given knowl-

edge, but also how to do it. Fourth, the system updates tutoring models that serves to generalise teaching interaction over several students of a particular classification. Finally, it is considered a general technique, that can be applied in any ITS independently of the domain.

This paper exposes an example of an AIES (a database design adaptive and intelligent educational system) and system executions with simulated students. We have done experiments with some parameters of the learning algorithm in order to study their effects at the convergence of the system and, at the same time, we have experimentally shown that AIES can learn an optimal policy to teach students interacting with reasonably few students. This makes possible its implementation in AIES with human students.

Actually, we are involved in the evaluation of the system with real students instead of simulated students and the study of hierarchical or compositional reinforcement learning algorithms.

Acknowledgements. We thanks Juan Carlos Martínez for his support on the implementation of the Q-learning algorithm.

References

Beck, J. (1998). Learning to teach with reinforcement learning agent. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*.

Beck, J., B. Woolf and C.R. Beal (2000). Advisor: a machine learning architecture for intelligent tutor construction. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*.

Brusilovsky, P. (1999). Adaptive and intelligent technologies for web-based education. Kunstliche Intelligenz. *Special Issue on Intelligent Tutoring Systems and Teleteaching*, **4**.

Burns, H., and C. Capps (1988). Foundations of intelligent tutoring systems: an introduction. *Foundations of Intelligent Tutoring Systems*. Hillsdale, N.J., Lawrence Erlbaum Associates, 1–19.

Castro, E., D. Cuadra, P. Martínez and A. Iglesias, (2002). Integrating intelligent methodological and tutoring assistance in a CASE platform: the PANDORA experience. In *Proceedings of the Informing Science & IT Education Conference*. Cork, Irland.

Chen, P. (1976). The entity-relationship model – toward a unified view of data. *ACM Transactions on Database Systems*, **1**(1).

Iglesias, A., P. Martínez, D. Cuadra, E. Castro and F. Fernández (2002). Learning to teach database design by trial and error. In *4th International Conference on Enterprise Information Systems*. Ciudad Real (Spain), pp. 500–505.

Kaelbling, L., M.L. Littman and A.W. Moore (1996). Reinforcement learning: a survey. *International Journal of Artificial Intelligence Research*, 237–285.

Mitchell, T.M. (1997). *Machine Learning*. McGraw-Hill, Boston.

Murray, T. (1999) Authoring intelligent tutoring systems: an analysis of the state of the art. *International Journal of Artificial Intelligence in Education*, **10**, 98–129.

Quinlan, J.R. (1993) *C4.5 Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, California.

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

Teorey, T., D. Yang, J. Fry (1986). A logical design methodology for relation databases using the extended entity-relationship model. *Computer Surveys*, **18** (2).

Watkins, C.J.C.H. (1989). *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK.

Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems*. Los Altos, CA, Morgan Kaufmann.

A. Iglesias Maqueda got a degree in computer science from Universidad Carlos III of Madrid in 1999. Since 2000, she works as assistant lecturer at the Advanced Databases Group in the Computer Science Department at Universidad Carlos III of Madrid. She is currently teaching database design and advanced databases. She has been working in several national research projects about advanced database technologies and CASE environments. Her research interests include adaptive intelligent educational systems, machine learning and database design.

P. Martínez Fernández got a degree in computer science from Universidad Politécnica of Madrid in 1992. Since 1992, she has been working at the Advanced Databases Group in the Computer Science Department at Universidad Carlos III of Madrid. In 1998 she obtained the PhD degree in computer science from Universidad Politécnica of Madrid. She is currently teaching database design, advanced databases in the Computer Science Department at the Universidad Carlos III de Madrid. She has been working in several European and national research projects about natural language processing, advanced database technologies, knowledge-based systems and software engineering.

F. Fernández is currently lecturer at the Computer Science Department of the Carlos III University of Madrid (UC3M). He received his PhD from UC3M in 2003, and he received his BSc in 1999 from UC3M, both in computer science. In the summer and fall of 2000, Fernández was a visiting student at the Center for Engineering Science Advanced Research at Oak Ridge National Laboratory (Tennessee). From 1998, he has participated in several European funded research and development projects. Fernández's research interests include reinforcement learning, autonomous robotics, neural networks and nearest neighbour approaches for supervised and unsupervised learning.

Sustiprinto mokymo modelio taikymas žiniatinkliu pagrįstose adaptyvinėse ir intelektinėse mokymosi sistemose

Ana IGLESIAS, Paloma MARTÍNEZ, Fernando FERNÁNDEZ

Vienas svarbiausių uždavinių taikant adaptyvines ir intelektines mokymo sistemas (AIES) yra pasirinkti tinkamą pedagoginę studentų mokymo strategiją. Taikant sustiprinto mokymo modelį sistema automatiškai išaiškina, kaip mokyti kiekvieną studentą individualiai, remiantis jau įgyta patirtimi mokant kitus panašių sugebėjimų studentus, – lygiai kaip elgtųsi ir mokytojas. Sustiprinto mokymo modelio taikymas nereikalauja išlaidų susijusių su taisyklių kūrimu ar kiekvienos mokymo strategijos euristine analize, kuria būtų siekiama išsiaiškinti ko, kada ir kaip reikėtų mokyti kiekvieną studentą. Be to, sustiprinto mokymo modelis leidžia sistemai taikytis prie studento individualiai – išanalizuojamas prieš tai vykęs panašaus pajėgumų mokinių mokymas. Šiame straipsnyje aptarta adaptyvinių ir intelektinių mokymo sistemų struktūra bei išnagrinėta, kaip sustiprinto mokymo modelis yra taikomas bet kuriam adaptyvinių sistemų pedagoginiam moduliui (šios sistemos technika gali būti pritaikoma mokant daugelio disciplinų), pasitelkiant *Q-learning* algoritmą. Pateiktame pavyzdyje paaiškintas sustiprinto mokymo modelio vaidmuo, kai sistema taiko bausmes nepažangiems studentais. Su kai kuriais algoritmų mokymosi parametrais buvo atlikti eksperimentai, siekiant ištirti jų daromą įtaką sistemų konvergencijai, išsiaiškinti sistemos efektyvumo kriterijai bei patikslinta, kiek sistema įvykdė veiksmų mokydama studentus atitinkamų dalykų. Šiame straipsnyje eksperimentiškai parodyta, kad sustiprinto mokymo modelio taikymas adatyvinėse mokymo sistemose leidžia šiai sistemai parinkti optimalią studentų mokymo strategiją, ypač kai su šia sistema dirba nuovokūs studentai. Aprašytąjį modelį galima taikyti internete.