Learning Pedagogical Policies from Few Training Data

Ana Iglesias and **Paloma Martínez** and **Ricardo Aler** and **Fernando Fernández** [1]

Abstract.

Learning a pedagogical policy in an Adaptive Educational System (AIES) fits as a Reinforcement Learning (RL) problem. However, to learn pedagogical policies requires to acquire a huge amount of experience interacting with the students, so applying RL to the AIES from scratch is infeasible. In this paper we describe RLATES, an AIES that uses RL to learn an accurate pedagogical policy to teach a course of Data Base Design. To reduce the experience required to learn the pedagogical policy, we propose to use an initial value function learned with simulated students, whose model is provided by an expert as a Markov Decision Process. Empirical results demonstrate that the value function learned with the simulated students and transferred to the AIES is a very accurate initial pedagogical policy. The evaluation is based on the interaction of more than 70 Computer Science undergraduate students, and demonstrates that an efficient guide through the contents of the educational system is obtained.

1 Introduction

Distance education is currently a hot research and development area. Traditionally, the courses in educational systems consist of static pages without student adaptability. However, since 1990s, researchers began to incorporate adaptability into their systems. Several Machine Learning (ML) techniques are used in educational systems, mainly, to model the students (personal scheduling rules and preferences): memory-based learning [11], supervised inductive learning [12], deductive learning [15], etc. Furthermore, different ML techniques are used to choose the best pedagogical strategy to be applied in each moment, like neural nets [13].

To represent the pedagogical knowledge based on Reinforcement Learning (RL) [1, 9] allows the educational system to adapt tutoring to students' needs. Thus, the system is able to sequence its content in an optimal way avoiding the definition of all static and predefined pedagogical policies for each student. Applying RL in AIES can be seen as *social navigation* in adaptive hypermedia systems [7], characterized by the use of other people's experiences in order to acquire knowledge for navigation.

However, RL algorithms need a great amount of experience in order to converge to a good pedagogical policy in AIES [1]. Moreover, in the initial trials of the learning, RL systems behave almost randomly, according to a value function randomly initialized. In an educational system, to teach the students in a reasonable way in every moment is essential, because the students could get bored and could stop working with the system.

Some works have demonstrated that Reinforcement Learning can be sped up if it uses knowledge previously acquired. For instance, advice rules [14] can be used to recommend that an action is preferred to another in a specified set of states.

Transfer learning refers to the injection of knowledge from previously learned problems. Policy Reuse probabilistically reutilizes policies learned for similar tasks [8]. The transfer of learning can also be performed by transferring the Q-function instead of the policy. These algorithms directly transfer the Q values associated to a policy that solves a task to initialize a new Q function in a new learning process [2]. However, if the source and target tasks are very heterogeneous, transfer learning also requires expert knowledge to perform the transfer [16].

Previous works in AIES have demonstrated that it is possible to reduce the time spent by the system in learning an optimal pedagogical policy when it has been previously initialized with another pedagogical strategy, even when this initialization does not completely match with the actual simulated students' needs [9].

In this paper, we propose to initialize the pedagogical policy of the educational system using simulated students, transferring the knowledge of their interactions with the system. The simulated students behavior is modelled as a Markov Decision Process, according to the information provided by a human expert. Thus, the system begins teaching the actual students based on a pedagogical policy according to the simulated student's model. Then, while the current students interact with the system, the pedagogical strategy is tuned according to their real needs.

In this paper, RLATES (Reinforcement Learning in Adaptive and inTelligent Educational System) is described. We focus on its pedagogical module, where RL has been applied in order to provide the students with direct navigation support through the system's contents. We demonstrate that the pedagogical policy learned with the simulated students is accurate and allows to teach the contents of the tutor to the students. We also demonstrate that RLATES is able to tune the initial pedagogical strategy according to the actual students' needs.

A second system has been implemented in order to compare the adaptation capability of RLATES. This system, called IGNATES (Indirect Guidance iN Adaptive and inTelligent Educational Systems) provides indirect navigation support, but the system does not learn how to teach better to the students.

The paper is organized as follows: first, the architecture of RLATES is summarized in Section 2. In Section 3, the pedagogical module of RLATES is formalized as a reinforcement learning problem. Then, the experiments setup is described in Section 4 and the experimental results are presented in Section 5. Finally, the main conclusions are given in Section 6.

2 RLATES Architecture

RLATES adopts the typical structure of an adaptive educational system, composed of four well differentiated modules: the student, do-

[1] Universidad Carlos III de Madrid, Spain, email: {aiglesia, pmf, aler, ffernand}@inf.uc3m.es

main, pedagogical and interface modules [3].

The domain module contains all the characteristics of the knowledge to teach. RLATES adopts a hierarchical knowledge structure, where each topic (knowledge item) has been divided into sub-topics, and these into others sub-topics, and so on. At the same time, each node of the tree contains tasks (definitions, examples, problems, exercises, etc.) in several formats (image, text, video, etc.). Figure 1 shows two examples of the knowledge trees. The first one shows the different tasks that can be executed for each topic. The second one only shows the topics.

The student module contains all important information about the student in the learning process: goals, student background knowledge, personal characteristics, historical behavior, etc. The user model is defined as the explicit representation of learning characteristics of each student. User models are usually used for looking ahead in the student's future behavior, his/her preferences or whatever s/he needs. We have represented the student characteristics using the overlay model [4], where the domain module overlays the student module showing when the student knows or not each domain topic.

The pedagogical module decides what, how and when to teach the domain module contents, following pedagogical decisions according to the user needs. Based on the pedagogical module, the system decides which is the best way to teach the knowledge items and tasks to each student (which is the best sequence of topics and tasks). The definition of this problem as a Reinforcement Learning problem allows the system to learn to teach each student based only on previous interactions with other students with similar learning characteristics. Moreover, the system is not only able to choose the next tasks to teach to the student, but also chooses the format in which the knowledge is going to be taught. Section 3 explain in detail how Reinforcement Learning is applied to the RLATES pedagogical module.

Finally, the interface module facilitates the communication between the system and the student. The adaptive techniques used in the interface module of the RLATES system are described in Section 4.2, where direct navigation support (based on the pedagogical module) and indirect navigation support (based on the domain knowledge) are distinguished.

3 Reinforcement Learning applied to Educational Systems

Reinforcement learning deals with agents connected to their environment via perception and action. On each step of the interaction, the agent observes the current state, s, and chooses an action to be executed, a. This execution produces a state transition and the environment provides a reinforcement signal, r, indicating how good the execution of the action has been to solve the task. The final goal of the agent is to choose the actions that tend to increase the long-run sum of values of the reinforcement signal, r [10].

The reinforcement learning components in the educational system environment are briefly defined next:

1. State space ($\mathcal{S}$): A state is defined as the description of the student's knowledge. It is represented by a vector of values related to the domain knowledge items (internal nodes of the domain knowledge tree). The *i-th* value of the vector represents the knowledge level of the student about the i-th topic. To maintain a reduced size of the state space, these values are defined in the set $\{0, 1\}$. 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].
2. Action space ($\mathcal{A}$): The actions that the tutor can execute are to show the knowledge items defined in the leaves of the knowledge tree.
3. Transition function ($\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$): This function indicates how the system changes its state when an action is executed. In an educational system, an student could change his/her knowledge state when a knowledge item is shown by the system (learning the knowledge or not according his/her current learning characteristics). The system perceives the current knowledge state of the student by evaluating his/her knowledge by tests.
4. Reward ($\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow R$): This function defines the reinforcement signals (rewards) received from the environment. This reinforcement function supplies a maximum value upon arriving to the goals of the tutor; i.e., when the student learns the total of the system's contents.

Using Reinforcement Learning, an action policy is learned by systematic trial and error, guided by a wide variety of algorithms. RLATES uses Q-learning [18] whose update equation of the Q table is shown in equation 2. This equation updates the values of the Q function only from experience obtained in the exploratory process (notice that information provided by human experts -as *is-prerequisite* relationships among topics- are not needed). The γ parameter controls the relative importance of future actions rewards with respect to new ones, and the α parameter is the learning rate, that indicates how quickly the system learns. Table 2 describes the Q-learning algorithm adapted to the AIES domain.

Q-learning *Algorithm Adapted to Educational Systems*

- For each pair $s \in \mathcal{S}, a \in \mathcal{A}$, initialize the table entry $Q(s, a)$.
- For each student
  - Test the current student's knowledge, obtaining s
  - Repeat
    - Select a knowledge tree leave, a, to show to the student, based on the Boltzmann exploration strategy, shown in Equation 1.

$$P(a) = \frac{\epsilon^{\frac{Q(s,a_i)}{\tau}}}{\sum_{a_j=1}^{n} \epsilon^{\frac{Q(s,a_j)}{\tau}}} \quad (1)$$

    - Test the current student's knowledge, s'.
    - Receive the immediate reward, r. A positive reward is received when the system goal is achieved. A null reward is obtained in any other case.
    - Update the table entry for $Q(s, a)$ according to Equation 2.

$$Q_t(s, a) = (1 - \alpha_t) Q_{t-1}(s', a') + \alpha_t [R(s, a) + \gamma \max_{a'} Q_{t-1}(s', a')] \quad (2)$$

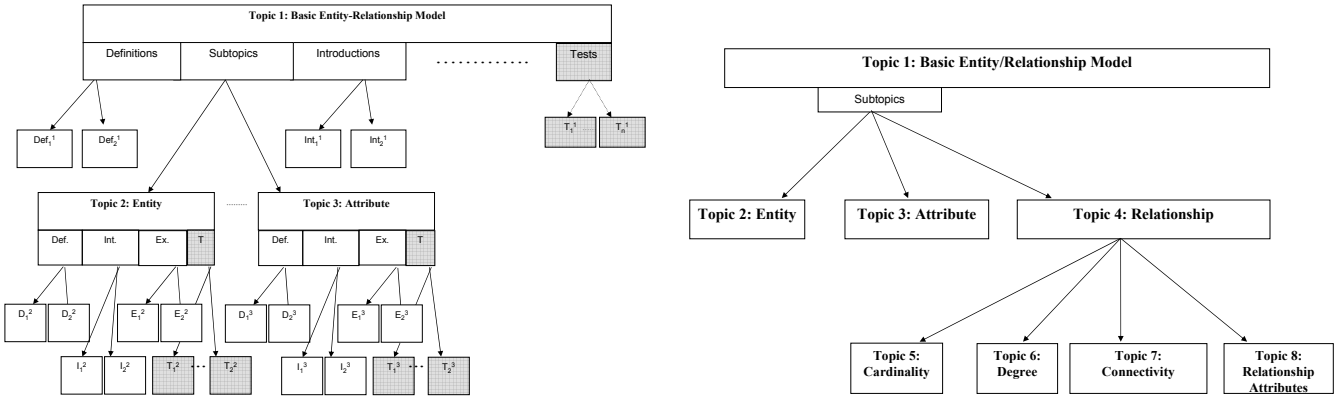
    - Set s to the current student's knowledge state, s'.

  Until s is a goal state.

Figure 2. Q-learning adapted to AIES.

The quality of the selected exploration/exploitation strategy determines the efficiency and efficacy of the learning process. A great

[2] This simplification allows to represent the Q function as a table of (state, action) pairs. Furthermore, the state space size is small enough, so no state generalization methods are required.



(a) Domain Model A (b) Domain Model B

Figure 1. Domain Models

variety of exploration strategies could be used [17]. RLATES uses the Boltzmann exploration policy, that estimates the probability of choosing the action a according to the Equation 1, where τ is a positive parameter called *temperature*. If the τ is high, all the probabilities of the actions have similar values and if the *temperature* is low, it causes a great difference in the probability of selecting each action.

4 Experimentation Setup

In order to study the scalability of RLATES, we have performed experiments with two different domain models. On the one hand, the domain model *A* is shown in Figure 1(a) and contains three topics (knowledge items) and 16 tasks, where most of the topics have 2 definitions, 2 introductions and 2 examples in two different formats: text and video. On the other hand, the domain model *B* is shown in Figure 1(b), and presents eight topics. This domain module has fifty-two tasks (not represented in the figure), where most of the topics have two definitions tasks, two introductions tasks and four examples tasks for execution in text and video formats.

Both domain models contain information about the *Conceptual Modelling* in *Database Design* proposed by Chen in [5], where *Entity-Relationship Model* is explained.

The next subsection describes how we perform the knowledge transfer from the expert to the AIES by initializing the Q table.

4.1 Initialization

For the experiments, the table Q is previously initialized with information about pedagogical strategies to teach *Database Design*. These pedagogical strategies have been learned by reinforcement learning using simulated students, which have been modelled as a Markov Decision Processes (the model is assumed to be markovian). The model of the students has been built with information provided by a human expert about student learning characteristics and relationships between topics in the *Conceptual Modelling* domain.

The predefined behavior of the simulated students is considered as a Markov Decision Process (MDP), where it does not matter which states the student has visited. That it, we assume that in order to get to a particular state of knowledge, it does not matter the previous history. This is a simplification, given the behavior of actual students can hardly be called Markovian, but only the current student's state is necessary to know if he/she is able to learn the Web page's content.

The MDP used to initialize RLATES for teaching the domain model *B* is shown in Figure 3, where only the actions that produce state transitions appear. It is supposed that the student does not change his/her state when a different action (an action that does not appear in the state) is executed. That is to say, the students' uncertainty deals with by assessing the students' understanding of the material.

For the construction of the MDP, two kinds of information are provided by the expert: on the one hand, the *is-prerequisite* relationships between topics; on the other hand, preferences on the students about the format and the type of the contents. In Figure 3 we can see how when the student is in the state0 (the students does not know anything about *Conceptual Modelling* and the action12 in executed (a Web page with an *Introduction* in *text format* about the *Entity* subtopic is shown), the student will stay at the same state with a probability of 75% and will change his/her state to the state1 (s/he obtained the knowledge expected by the system for the topic *Entity*) with a probability of 25%. It is important to notice that the information provided by the expert is only used to build the MDP.

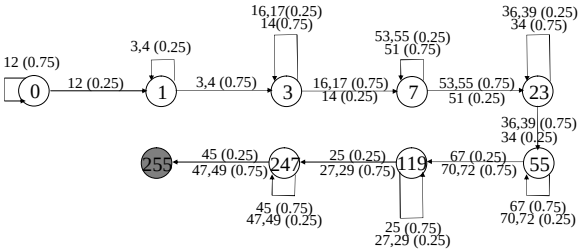


Figure 3. MDP modelling simulated students behavior.

4.2 System Versions

In order to evaluate the advantages of adapting the teaching strategies according to the student characteristics, we have implemented two versions of the educational system. The first one is RLATES (Reinforcement Learning in Adaptive and inTelligent Educational Systems). The second one is IGNATES (Indirect Guide Navigation in Adaptive and inTelligent Educational Systems).

The interface is very similar in both system versions, where the content page is divided in two frames. The left frame contains the system knowledge structured as a tree. The right frame shows to the student a task (definition, introduction, example, etc.) about the current topic (marked in bold red at the knowledge tree).

The main difference between RLATES and IGNATES is the navigation support system, explained next for each system.

4.2.1 *IGNATES system*

The students interacting with the IGNATES system are guided in an indirect way through the knowledge tree (notice that to guide in an indirect way to the students is better than not guide at all). The student chooses the next topic to visit, based only on the information provided by the system and changes the color of the knowledge tree links (using annotation). This information summarizes which topics the student has previously visited (blue links), which are passed (the student answered a test correctly; green links) and which ones are not (orange links).

When the student clicks on a tree link (a specific topic), the system shows him/her all the information about the topic (definitions, introductions, examples, tests, etc.).

The interface of the IGNATES system is presented in Figure 4. If the student clicks in the *Next* button the system shows him/her the next topic of the knowledge tree, and if s/he clicks the *Previous* button, the system shows him/her the previous topic in the knowledge tree.

In this system, the student also decides when s/he is ready to answer the test about a specific topic and when to finish the interaction.

Figure 4. IGNATES interface.

4.2.2 *RLATES system*

The students interacting with the RLATES system are guided directly by the *Next* button at the interface right frame.

In this system, the students can see the knowledge tree, where the color of the links follows the annotation rules of the IGNATES system, but they can not click on these links. The students are only allowed to click on the *Next* button to keep on learning.

When the student clicks the *Next* button, the system provides him/her several links (see Figure 5(a)). Each link shows a specific topic tasks (definition, introduction, example, etc.) in a specific format (image, text, etc.). The system chooses the five most appropriate tasks for the student to learn according to the Reinforcement Learning algorithm and the *Boltzmann* exploration/exploitation policy provided by the pedagogical module, as defined in Section 3.

Then, the student decides which is the best task to learn next and in which format to do it, based on the system recommendation, providing him/her an interaction control feeling.

When the student chooses a specific action, the system shows him/her only two tags (see Figure 5(b)). The first tag shows the tasks chosen for the student and the second one shows a test. The student must answer the test in order to continue. Depending on whether s/he passes the test or not, a knowledge state transition is generated. This state transition is used to update the Q table.

It is important to notice that, in order to study the learning evolution of the system, the students interact with RLATES sequentially.

4.3 Experimentation Environment

More than seventy students have interacted with the system. All of them are 2nd course undergraduate students of Computer Science at the same university and, therefore, it is supposed that all of them have similar level of knowledge about the system contents. Nevertheless, their knowledge about the system material was confirmed with an initial questionnaire about *Database Design*. In this way, we can assume that all the students belong to the same cluster of students, and modelled with the MDP provided by the expert.

In order to avoid the effects of the environment *noisy* variables, some rules can be applied [6]. For instance, we have randomly distributed the students into experimentation groups, arbitrarily assigning the students interaction time; the room where the students interact with the system is quiet and it has no windows, posters or other distractions; etc.

In addition, *blind* experiments have been carried out in order to ease the subjectivity that the experimenter could add to the student interaction. I.e., the responsible does not know the version of the system the student is interacting with (RLATES or IGNATES).

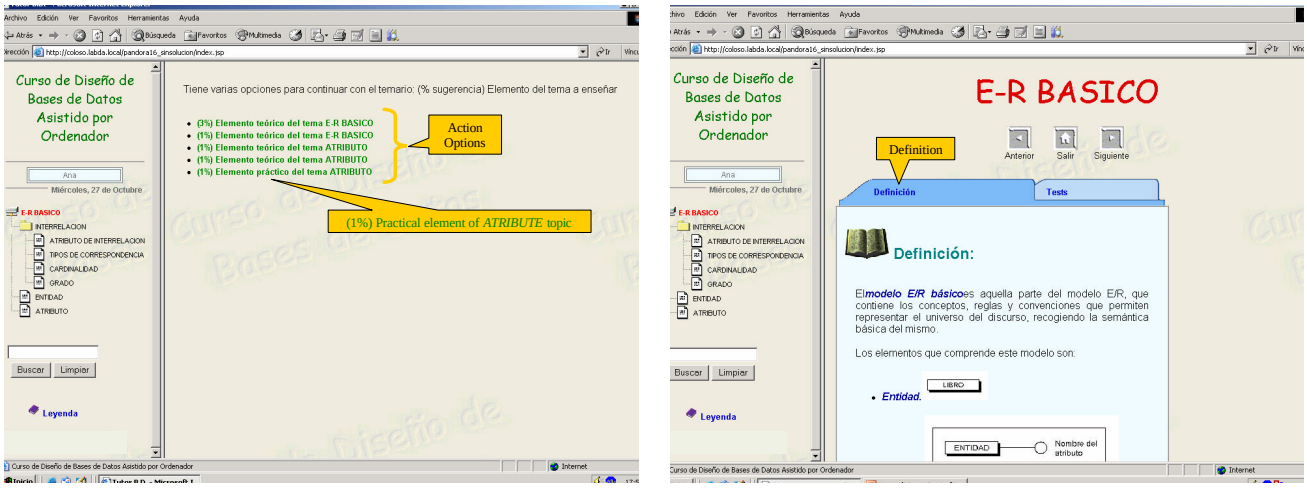
5 Results

In the following experiments, we study the performance of RLATES, and compare it with the performance of IGNATES. The performance of both systems is measured by using three features: (i) the number of web pages (actions) that they need to show to each student so that the student learns the contents of the course; (ii) the total time that each student is interacting with each system; and (iii) the final student's level of knowledge after the interaction with the systems. The experiments have been divided in two sections, depending on the knowledge tree used.

5.1 Domain Model *A*

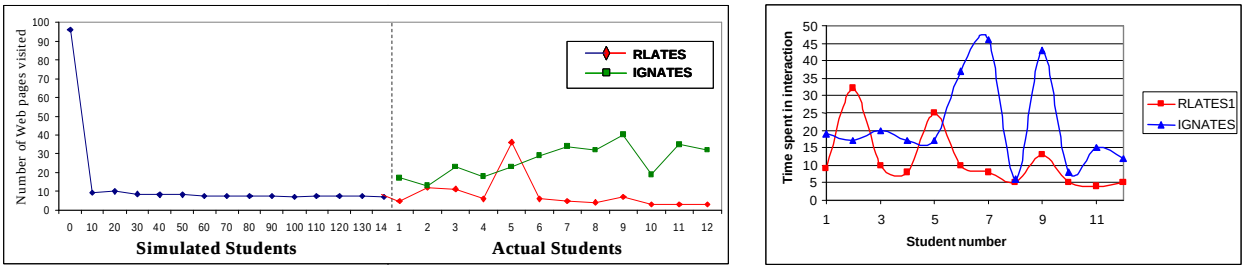
Figure 6(a) shows the number of web pages required by RLATES to teach the content of the AIES. The x-axis shows the number of students that have interacted with the system. The figure is divided in two parts. In the left, we show the learning performance when simulated students interact with RLATES. The students follow the model provided by the expert (and represented by an MDP). Initially, RLATES needs around 90 actions to teach the content of the AIES to the simulated students. However, after only 10 students interacting with the system, the performance decreases down to 10 actions. After the 150 simulated students, the pedagogical policy is tuned, obtaining a performance of only 8 actions.

Then, the Q table obtained with the simulated students is used to initialize the pedagogical module of RLATES with actual students. The result of the interaction with the actual students is shown in the



(a)Actions provided by the system (b) Content tags

Figure 5. RLATES interface.



(a) Number of web pages (b) Time

Figure 6. Results of learning the Domain Model *A*.

right part of Figure 6(a). Notice that the unit of the x-axis differs to the left part of the figure. For the initial students, RLATES needs around 10 actions to teach the content of the AIES. However, while the students are learning the tutor's contents, RLATES also modifies the pedagogical policy according to their actual learning characteristics by tuning the Q table obtained with the simulated students. Then, the policy is improved, and after a while, some students only need to visit 3 Web pages.

For comparison, we also include in the right part of the figure the number of Web pages visited by a different set of students that interact with IGNATES. We can observe how the students interacting with IGNATES visit more Web pages that students interacting with RLATES, even when RLATES is still tuning the teaching policy. That demonstrates that the pedagogical policy used by RLATES is very useful for the students.

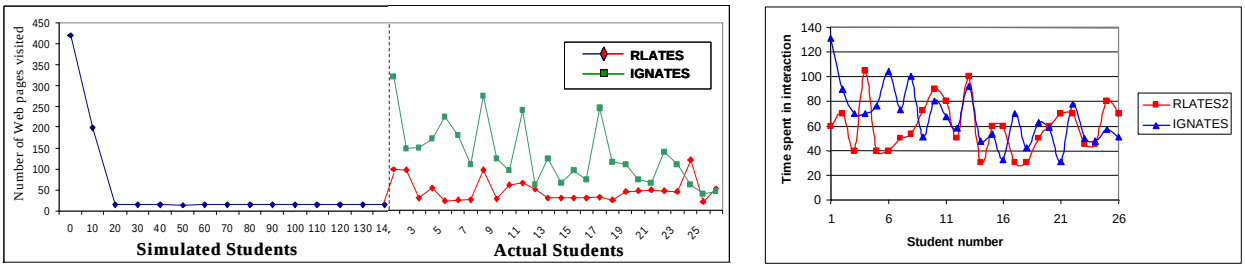
Figure 6(b) shows the time that the students take to learn the content of the AIES. We can conclude that the students interacting with RLATES need less time (on average) to finish the interaction than the students interacting with IGNATES. Again, this is an indication that the RLATES teaching policy is good.

Finally, the level of knowledge of the students after their interactions with the systems is studied. The students had to carry out an exam with open-ended questions and they were evaluated by a human tutor. The IGNATES student average qualification was 9.58 (marking from 0 to 10) and the RLATES student average qualification was 9.62. With respect to the standard deviation, the IGNATES student standard deviation between the interactions was 0.37 and the RLATES student standard deviation was 0.35. Then, we can conclude that there is not significant differences between the student's final level of knowledge.

5.2 Domain Model *B*

Figure 7(a) shows the learning performance of RLATES and IGNATES teaching the domain module B. The figure follows the same structure than Figure 6(a). We can observe that, after the simulated students interact with the system, a performance of around 10 actions is achieved. Then, RLATES tunes the teaching policy according to the actual students' needs, obtaining an average number of Web pages visited by the students of 44.73, with an average deviation of 26.65. Again, these values are much better than the ones obtained by IGNATES (132.88 Web pages, with an standard deviation of 75.86).

Regarding the total time spent interacting with the system, in Figure 7(b) we can observe that the time spent in the interaction is not directly proportional to the number of Web pages visited. For instance, the RLATES curve shows that some students spent more time reading and assimilating the Web page content than other students



(a) Number of web pages (b) Time

Figure 7. Results of learning the Domain Model *B*.

and the curve does not seem to converge. The average time spent by IGNATES students is higher (64.15 minutes) than the average time spent by RLATES students (59.61 minutes). Furthermore, the standard deviation of the time spent in interactions is similar for both systems, around 21 minutes.

As in the previous case, the students level of knowledge after their interaction with the systems was studied. The average qualification of the IGNATES students was 9.44 ± 0.47 (marking from 0 to 10) and the average qualification of the RLATES students was 9.54 ± 0.55, so the differences are not significant either.

6 Conclusions

In this paper we describe the pedagogical module of an AIES as a RL problem. The implemented system, called RLATES, updates the pedagogical policy automatically according to the students' needs in each moment of the interaction. The pedagogical policy is based only on previous experience with other students with similar learning characteristics.

Due to the great amount of experience required in order for the system to teach the students properly, we propose to initialize the pedagogical policy with the one obtained using simulated students. Simulated students are modelled as an MDP built based on information provided by a human expert. With this transfer of knowledge, the system is able to sequence its contents in an accurate way even for the first student. Therefore, while it interacts with new students, it tunes the teaching policy according to the actual students' needs.

More than 70 undergraduate students have interacted with the system, demonstrating that the system pre-training is useful for the actual students to learn the contents of the educational system, avoiding to use pseudo-random teaching policies with the first students when RL is applied from scratch.

In the future we will use a model-based reinforcement learning strategy in order for the system to learn faster accurate pedagogical strategies for the current students.

ACKNOWLEDGEMENTS

This work was supported by the project GPS (TIN2004/07083).

REFERENCES

[1] J. Beck, *ADVISOR: A machine learning architecture for intelligent tutor construction*, Ph.D. dissertation, University of Massachusetts Amherst, 2001.

[2] Michael Bowling and Manuela Veloso, 'Bounding the suboptimality of reusing subproblems', in *Proceedings of IJCAI-99*, (1999).

[3] Hugh Burns and Charles Capps, 'Foundations of intelligent tutoring systems: An introduction', in *Foundations of Intelligent Tutoring Systems*, ed., Lawrence Erlbaum Associates, 1–19, Hillsdale, N.J., (1988).

[4] B. Carr and I. Goldstein, 'Overlays: A teory of modeling for computer aided instruction', Technical report, AI Laboratory, Massachusetts Institute of Technology, Cambridge, MA. Technical Report AI Memo 406, (1977).

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

[6] D. Chin, 'Empirical evaluations of user models and user-adapted system', *User Modelling and User Adapted Interaction*, **11**(1), 181–194, (2001).

[7] A. Dieberger, 'Supporting social navigation on the world wide web', *International Journal of Human-Computer Studies*, **6**(46), 815–825, (1997).

[8] Fernando Fernández and Manuela Veloso, 'Probabilistic policy reuse in a reinforcement learning agent', in *Proc. 5th Int. Joint Conference on Autonomous Agents and Multiagent Systems*, (2006).

[9] A. Iglesias, P. Martínez, R. Aler, and F. Fernández, 'Learning content sequencing in an educational environment according to student needs', in *Proc. 15th International Conference on Algorithmic Learning Theory*, eds., S. Ben-David, J. Case, and A. Maruoka, pp. 454 – 463. LNCS, (2004).

[10] Leslie P. Kaelbling, Michael L. Littman, and Andrew W. Moore, 'Reinforcement learning: A survey', *Journal of Artificial Intelligence Research*, **4**, 237–285, (1996).

[11] R. Kozierok and P. Maes, 'A learning interface agent for scheduling meetings', in *Proc. Int. Workshop on Intelligent User Interfaces*, pp. 81–88, (1993).

[12] P. Langley and S. Ohlsson, 'Automated cognitive modeling', in *Proceedings of the Second National Conference on Artificial Intelligence*, (1984).

[13] M. Lebowitz, 'Experiments with incremental concept formation: Unimen', *Machine Learning*, **2**, 103–138, (1987).

[14] Richard Maclin, Jude Shavlik, Lisa Torrey, Trevor Walker, and Edward Wild, 'Giving advice about preferred actions to reinforcement learners via knowledge-based kernel regression', in *Proceedings of the Twentieth National Conference on Artificial Intelligence*, (July 2005). To appear.

[15] D. Sleeman and S. Brown, *Intelligent Tutoring Systems. Computers and People Series*, Academic Press, London, 1982.

[16] Matthew E. Taylor and Peter Stone, 'Behavior transfer for value-function-based reinforcement learning', in *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, (July 2005). To appear.

[17] Sebastian Thrun, 'Efficient exploration in reinforcement learning', Technical Report C,I-CS-92-102, Carnegie Mellon University, (January 1992).

[18] C. J. C. H. Watkins, *Learning from Delayed Rewards*, Ph.D. dissertation, King's College, Cambridge, UK, 1989.