Reinforcement learning of pedagogical policies in adaptive and intelligent educational systems

Ana Iglesias *, Paloma Martínez, Ricardo Aler, Fernando Fernández

Department of Computer Science, University Carlos III of Madrid, Avda. Universidad, 30, 28911 Leganés - Madrid, Spain

ARTICLE INFO

Keywords:
Reinforcement Learning
Adaptive and intelligent educational systems
Distance learning
Artificial intelligence applied to intelligent tutoring systems

ABSTRACT

In an adaptive and intelligent educational system (AIES), the process of learning pedagogical policies according the students needs fits as a Reinforcement Learning (RL) problem. Previous works have demonstrated that a great amount of experience is needed in order for the system to learn to teach properly, so applying RL to the AIES from scratch is unfeasible. Other works have previously demonstrated in a theoretical way that seeding the AIES with an initial value function learned with simulated students reduce the experience required to learn an accurate pedagogical policy. In this paper we present empirical results demonstrating that a value function learned with simulated students can provide the AIES with 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 and useful 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.

To represent the pedagogical knowledge based on Reinforcement Learning (RL) [1,8] 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. However, RL algorithms need a great amount of experience in order to converge to a good action policy [1]. Moreover, if the system has not been previously initialized to a pedagogical strategy, in the initial trials of the learning, RL systems behave almost randomly according to a value function initialized randomly. 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 learning can be speeded up if the value function is initialized with another which was learned for solving a similar task with a similar model [4]. Similarly, initializing the value function with pre recorded experience tuples may accelerate the learning of the action policy [10]. In previous work [7], we have verified empirically with simulated students that the size of the learning phase can be reduced by initializing the system with a pedagogical strategy, even when the initialization does not completely match with the current students' needs.

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. Two different AIES have been implemented in order to obtain experimental results: RLATES and IGNATES. RLATES (Reinforcement Learning in Adaptive and inTelligent Educational System) applies RL in order to provide the students with direct navigation support through the system's contents. 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. This work demonstrates that the pedagogical policy learned with the simulated students is accurate and allows to teach the contents of the tutor to the students using RLATES. We also demonstrate that RLATES is able to tune the initial pedagogical strategy according to the actual students' needs[1].

The paper is organized as follows: first, the architecture of RLATES is summarized in Section 2. 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.

* Corresponding author.
E-mail addresses: aiglesia@inf.uc3m.es (A. Iglesias), pmf@inf.uc3m.es (P. Martínez), aler@inf.uc3m.es (R. Aler), ffernand@inf.uc3m.es (F. Fernández).

[1] This work has been partially supported by the Software Process Management Platform: Modelling, reuse and measurement project (TIN2004/07083) and the "APEINTA" project founded by the Spanish Minister of Education and Sciences.

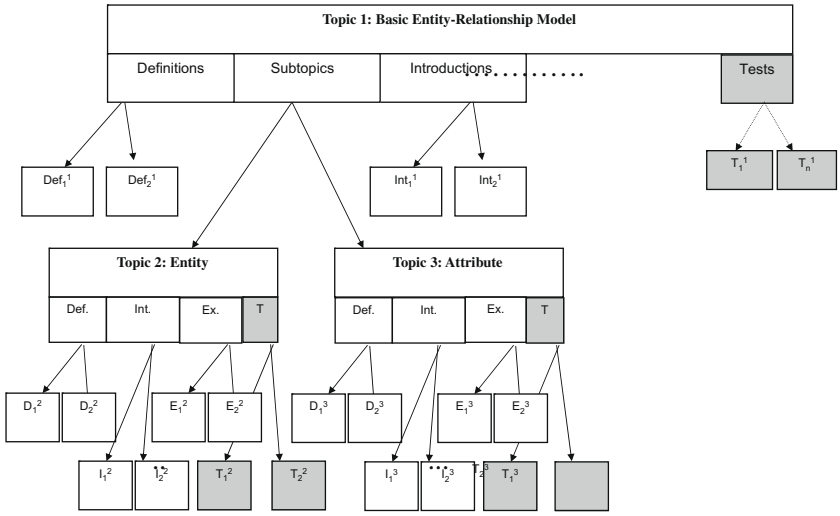

Fig. 1. Example of domain model.

2. RLATES Architecture

RLATES adopts the typical structure of an adaptive educational system, composed of four well differentiated modules: the student, domain, pedagogical and interface modules [2].

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, etc. At the same time, each node of the tree contains tasks (definitions, examples, problems, exercises, etc.) in several formats (image, text, video, etc.). Fig. 1 shows one example of knowledge tree. This model contains three topics (knowledge items) and 16 tasks, where most of the topics have two definitions, two intro ductions and two examples in two different formats: text and video.

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 charac teristics using the overlay model [3], 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 top ics 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 sim ilar 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. In previ ous works, the pedagogical module of RLATES is formalized as a reinforcement learning problem [7], using the *Q* learning algo rithm [11] and the Boltzmann exploration/exploitation strategy. In Fig. 2, how the *Q* learning algorithm is adapted to the tutor do main is explained, based on the definition of the Reinforcement Learning components for an adaptive and intelligent educational system.

In each step for each student, the system chooses a task to show to the student (an action to execute), based on the *Q* table. Then, the system evaluates by a test if the student has understood the knowledge shown in the last task. After that, the system receives the immediate reward and update the *Q* table entry according to this reward.

Finally, the interface module facilitates the communication be tween the system and the student. The adaptive techniques used in the interface module of the RLATES system are described in Section 4.1, where direct navigation support (based on the pedagogical module) and indirect navigation support (based on the domain knowledge) are distinguished.

3. System functional phases

The use of RLATES requires four phases in order to adapt better to each student in every moment of the interaction:

Student clustering: RLATES requires to cluster the students according to their learning characteristics (level of knowledge, the Web pages format that they prefer, etc.). The system maintains one *Q* table for each cluster of students. This allows the system to adapt better to each student cluster. In this work the clustering is performed by using expert knowledge based on evaluations of students, but an automatic approach could be used [9].

System initialization: We model each student cluster with infor mation provided by a human expert about student learning characteristics and relation ships between topics in the Database domain. The result is a Markov Decision Pro cess (this is a simplification, given the behav ior of real students can hardly be called Markovian.). An MDP for the domain model described in Fig. 1 is shown in Fig. 3, where only the actions that produce state transi tions appear. For the construction of the MDP, two kinds of information is provided by the expert: on the one hand, the is prere quisite relationships between topics; on the other hand, preferences on the students about the format and the type of the con

Q-learning *Algorithm Adapted to Educational System Domain*

- 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
  - · Do

    Select a knowledge tree leaf, a, to show to the student, based on an exploitation strategy derived from Q.
    Test the current student's knowledge, s', based on the last knowledge shown.
    Receive the immediate reward, r. A positive reward is received when the system goal is achieved. A reward of 0 is obtained in any other case.
    Update the table entry for $Q(s^{\varsigma} a)$ according to next equation:

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

    Let us set s to the current student's knowledge state, s'.
  Until s is a goal state.

Fig. 2. Q-learning adapted to educational system domain.

tents. It is important to notice that the infor mation provided by the expert is only used to build the MDP, not to directly learn to teach the students. From the MDP obtained, an initial Q Table is generated using the Q Learning algorithm, which seeds the follow ing phase.

System Training: In this phase, the system interacts with real students and explores new pedagogical alternatives in order to teach the system knowledge, sequencing the domain content in different ways. At the same time the sys tem interacts with the students and updates the appropriate Q table, adapting the teach ing policy according to their necessities based only on previous interaction with other students.

System Use: When the system has converged to a good pedagogical strategy, it is time to use this information to teach other students with similar learning characteristics. These stu dents will achieve their knowledge goals in the best way the system has learned.

4. Experimentation setup

In order to study the scalability of RLATES, we have performed experiments with the domain model shown in Fig. 1. More than seventy students have interacted with the systems, all of them 2nd course undergraduate students of Computer Science at Uni versidad Carlos III de Madrid. The experimentation environment has been carefully studied in order to avoid the effects of the *noisy* variables. Some rules have been applied [5] and *blind* experiments have been carried out.

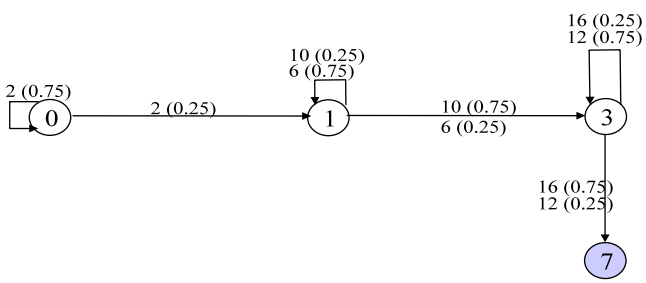

Fig. 3. MDP modelling simulated students' behavior.

4.1. System versions

In order to evaluate the advantages of adapting the teaching strategies according to the student characteristics, we have imple mented two versions of the educational system. The first one is RLATES (Reinforcement Learning in Adaptive and inTelligent Edu cational Systems). The second one is IGNATES (Indirect Guide Nav igation 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 navi gation support system, explained next for each 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 informa tion provided by the system and changes the color of the knowl edge 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 infor mation about the topic (definitions, introductions, examples, tests, etc.).

The interface of the IGNATES system is really similar to the interface of RLATES (see Fig. 4), but the *Next* and *Previous* buttons have different functionalities. If the student clicks in the *Next* but ton 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.

On the other hand, 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.

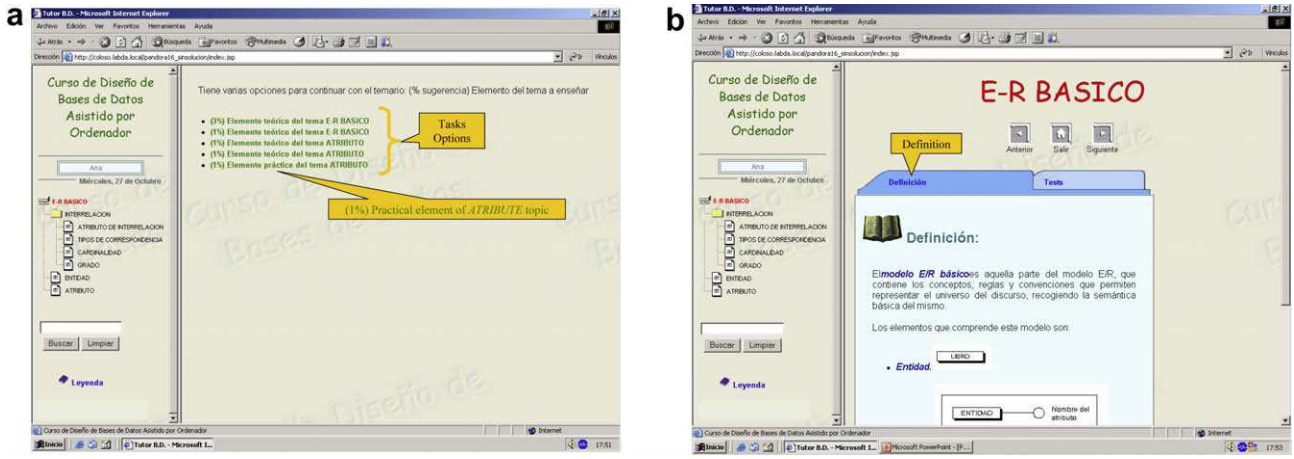

Fig. 4. RLATES interface.

When the student clicks the *Next* button, the system provides him/her several links (see Fig. 4a). 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 previous works [7].

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 Fig. 4b). 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.

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.

Fig. 5a 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 simulated students follow the model provided by the expert (and represented by the MDP shown in Fig. 3). 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 eight 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 right part of Fig. 5(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 three 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.

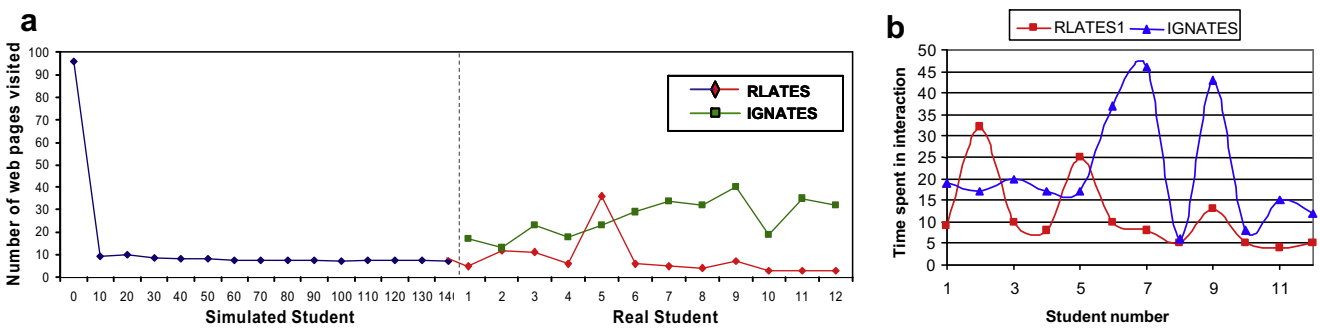

Fig. 5. Results of learning the domain model *A*.

Fig. 5b 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.

In the literature additional evaluations for different domain models can be found showing qualitatively similar results [6].

6. Conclusions

In this paper, we present empirical results demonstrating the pedagogical module of an AIES can be described as a RL problem. In this way, the system is able to update the pedagogical policy automatically according to the students's needs in each moment of the interaction, based only on previous experience with other students with similar learning characteristics. Moreover, we demonstrate that a value function learned with simulated students defined as MDPs can provide the AIES with a very accurate initial pedagogical policy.

More than 70 undergraduate students have interacted with the system, demonstrating two main issues: first, the direct navigation support based on reinforcement learning is really useful for the students to learn the contents of the educational system; second, the previous pedagogical policy initialization with simulated students reduces the system *Training* phase.

References

[1] J. Beck, ADVISOR: A machine learning architecture for intelligent tutor construction, PhD thesis, University of Massachusetts Amherst, 2001.

[2] Hugh Burns, Charles Capps, Foundations of intelligent tutoring systems: An introduction, in: Lawrence Erlbaum Associates (Ed.), Foundations of Intelligent Tutoring Systems, Hillsdale, NJ, 1988, pp. 1–19.

[3] B. Carr, I. Goldstein, Overlays: A teory of modeling for computer aided instruction, Technical report ai memo 406, AI Laboratory, Massachusetts Institute of Technology, Cambridge, MA, 1977.

[4] J. Carroll, T. Peterson, Fixed vs. dynamic sub-transfer in reinforcement learning, in: International Conference on Machine Learning and Applications, 2002.

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

[6] A. Iglesias, P. Martínez, R. Aler, F. Fernández, Learning pedagogical policies from few training data, in: Proceedings of the 17th European Conference on Artificial Intelligence, Workshop on Planning, Learning and Monitoring with Uncertainty and Dynamic Worlds, Riva del Garda (Italy), 2006, pp. 13–18.

[7] A. Iglesias, P. Martínez, R. Aler, F. Fernández, Learning teaching strategies in an adaptive and intelligent educational system through reinforcement learning, Applied Intelligence, in press.

[8] A. Iglesias, P. Martínez, F. Fernández, An experience applying reinforcement learning in a web-based adaptive and intelligent educational system, Informatics in Education International Journal 2 (2003) 1–18.

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

[10] William D. Smart, Leslie Pack Kaelbling, Practical reinforcement learning in continuous spaces, In Seventeenth International Conference on Machine Learning, 2000, pp. 903–910.

[11] C.J.C.H. Watkins, Learning from Delayed Rewards, PhD thesis, King's College, Cambridge, UK, 1989.