Modeling Students Self-Studies Behaviors

Pedro Mota
INESC-ID
Instituto Superior Técnico
Lisboa, Portugal
Carnegie Mellon University
Pittsburgh, PA, USA
pedro.mota@l2f.inesc-id.pt

Francisco S. Melo
INESC-ID
Instituto Superior Técnico
Universidade de Lisboa
Lisboa, Portugal
fmelo@inesc-id.pt

Luísa Coheur
INESC-ID
Instituto Superior Técnico
Universidade de Lisboa
Lisboa, Portugal
luisa.coheur@inesc-id.pt

ABSTRACT

In this work we propose a decision-theoretic approach to Intelligent Tutoring Systems (ITSs) that seeks to alleviate the need for extensive development and hand-tuning in the design of such systems.

Given a set of available learning materials, our approach enables the ITS to track the students' difficulties and provide the right material at the right time. We model the learning process as a Partially Observable Markov Decision Process (POMDP), where the hidden information corresponds to the student's familiarity with each of the topics to be learned. The student's progress is monitored from his/her performance in different test exercises and, depending on this performance, the ITS actively determines which type of materials should be provided to the student. We deploy our proposed approach in a learning scenario and compare the ability of our system to model the students' self-study behaviors of the learning materials. Our initial results show that such behaviors are not trivial to model and that our proposed POMDP approach better matched the observed student behaviors in comparison with a baseline teaching policy that corresponds to a fix set of actions hand-designed by a human expert.

Categories and Subject Descriptors

K.3.1 [**Computing Milieux**]: Computers and Education; I.2 [**Computing methodologies**]: Artificial intelligence

General Terms

Algorithms

Keywords

Virtual agents in games and education; Single and multi-agent planning and scheduling

1. INTRODUCTION

The education community has long been familiar with the use of ITSs to improve the efficiency of teaching [28]. ITSs directly address the problem of providing individualized teaching to students, which is one of the main challenges in education. In a traditional classroom environment, one teacher faces multiple students, making it impractical to attend to each student's specific needs. Instead, teachers commonly opt for giving an "average" class, which is known to be less than optimal [12]. Solving the previous problem is far from trivial, since there are many variables that influence the student's learning process. Despite the complexity of this area, impressive achievements have already been accomplished, showing that ITSs can improve learning significantly [2, 11, 22, 29].

One well-known issue with ITSs, however, is the significant effort required to develop such systems [7]. Several works proposed *authoring tools* that seek to minimize the design effort involved in the development of ITSs, with different levels of success [17]. Recently, an alternative line of work proposed the integration of Reinforcement Learning (RL)-based approaches in ITSs [5, 6, 13, 14, 26, 27]. The main difference between the latter and the former is that, in well-defined environments, RL-based approaches seek to completely eliminate the authoring effort required, instead of just reducing it.

In spite of the achievements of the previously mentioned works, the development cost behind ITSs remains high, since there is a need to design the possible pedagogical actions (such as hints) that the ITS can chose from, depending on the current state of the student's learning process. The cost involved in defining such actions is significant, and constitutes an important bottleneck that probably prevents the massification of ITSs.

In this paper we propose an alternative perspective on the use of ITSs. We argue that it is possible to alleviate the design effort needed to come up with pedagogical actions for such systems, investigating their potential in a novel and unexplored tutoring scenario. In particular, we propose that ITSs should actively intervene during the period in which students are doing their self-studies, since the latter is a key point of the student's learning process. Self-study is part of the common learning paradigm in which students complement the traditional classroom environment with periods of self-study with some learning materials. Most existing ITSs are unable to properly accomodate this aspect of the learning process and, instead, focus on "repairing" the student in an exercise context [10, 11, 22].

In this work we describe the first steps toward an ITS that explores a student's self-study scenario. In the designed scenario, students work with a set of slides at their own pace, similarly to what they would do when preparing for an exam. These slides are grouped into subtopics and at the end of

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May, 4–8, 2015, Istanbul, Turkey.*

each of them students can perform drill exercises. If needed, they can revisit the slides in-between exercises. We designed an ITS based on a decision-theoretic model (POMDP) that is able to successfully track the students' needs and provide the right type of content at the right time, in order for the student to overcome his/her difficulties when performing the exercises. One of the benefits of using this methodology is that we transfer the burden of having to manually define some help mechanism to the learning materials that already exist. Another advantage is that this setup can bootstrap from past experience. For example, the ITS can use information about the learning materials that previous students revisited when stuck in a specific exercise to better guide other students in a similar situation.

2. RELATED WORK

As discussed in Section 1, recent years have witnessed an increasing interest in the application of artificial intelligence techniques to the development of ITSs. In particular, several approaches propose the use of decision-theoretic models and RL to automatically discover an optimal teaching policy for some pedagogical decision that the ITS needs to take, by observing data instead of manually specify it *a priori*. This constitutes an important advantage over other types of user modeling approaches applied in ITSs, such as Bayesian Knowledge Tracing [18, 21] or Deep Learning-based techniques [16], which although allow the dynamic prediction of student behavior do not provide a concrete teaching policy.

One example of an ITS that makes use of RL-based techniques is Wayang Outpost, which prepares students for the Scholastic Aptitude Test in the math domain [2]. The implemented ITS is able to learn optimal hint sequences to show the students when they are interacting with the system. The motivation for this approach is that it should not be assumed that scripted hints reflect the true nature of the students that request them, since they can be of different levels. Therefore, efficiency can be achieved, for instance, by skipping the hints in skills that the student already knows.

Another line of research uses RL to address the optimal activity sequencing task [14, 27]. One example of such ITSs is AnimalWatch, which teaches students through an interactive application themed with wild life [6]. The underlying pedagogical decision to be carried out consists in selecting the actions that allow the ITS to achieve a certain goal. For instance, the goal specification "Students should make precisely 1 mistake per problem" allows tailoring the learning process to the student: students that are already proficient need harder exercises than students that have more difficulties. Addressing this problem also means that the ITS controls how fast the student progresses, because it might be necessary to chose exercises from harder topics.

Still along the optimal activity sequencing line of research, but from a different perspective, is the work of Lopes et al. [13] that describes an educational game where students must provide a combination of currency (coins and bills) such that it matches the buying price of a presented toy. By manipulating the price to be matched, the ITS is able to make the student practice different math skills, such as addition, multiplication, or the use of real numbers. In this context, the decision of which activity to provide is based on an estimation of learning gains. Therefore, the activity that the ITS estimates as conveying the highest learning gains is the one provided to the student.

Other related works not only perform the optimal activity sequencing task in the context of choosing an exercise, but also consider the teaching of some content as an activity itself [4, 8], which is closer to the work we propose. One of such ITSs is the DataBase Design (DBD) web application, where students learn database design and implementation [15]. The learning materials are in the form of texts, images, videos or animations, and are organized in hierarchical topics. Associated with these contents are assessments, which include problems, exercises, tests and simulations. In this setting, the tutoring strategy revolves around finding the best curriculum sequence for a specific student, such that his learning process is optimal, which implies that several types of decision have to be made. These include what learning material to show (what topic to present), in what format (text, video, etc.), what exercises to present and when to present them.

Finally, there is also work that uses RL in order to decide which dialog moves are more adequate in a tutorial dialog context [26]. An example of this type of pedagogical decision is in the ITS Cordillera [5], which uses natural language dialogs to discuss physics problems students are trying to solve. When solving such problems students need to apply several different steps. Traditionally these steps are coarse steps from the ITS's point of view, but Cordillera further divides them in micro-level steps. Some examples of micro-level steps are: selecting the principle to apply, writing the corresponding equation, solving the equation, or engaging in some qualitative discussion about the current step. The following examples demonstrates the use of micro-level steps:

1. **T:** So let's start with determining the value of KE_0.
2. **T:** Which principle will help you calculate the rock's kinetic energy at T_0? Please provide the name of the principle, not an equation. **(ELICIT)**
3. **S:** Definition of kinetic energy.
4. **T:** Yes, I agree. Now I will write the equation for applying the definition of kinetic energy to the rock at T_0: $KE_0 = \frac{1}{2} \times m \times v_0^2$ **(TELL)**

The labels **T** and **S** designate tutor and student, respectively. The dialog turns labeled with **ELICIT** or **TELL** represent the pedagogical approach to the micro-step, which means that the ITS can ask the student information with a question or can tell this information. The ITS may also require that students justify their answer. In this context, the tutoring strategy that needs to be determined is whether to elicit or to tell a micro-level step and to require or not a justification for the student's answer.

Given the described current state-of-the-art, it is possible to conclude that the strategies for providing feedback to the students are based on pedagogical content developed for specific exercises. The only exception is the DBD system, which although uses the learning materials for feedback, it uses them in a coarse manner. The consequence is that the feedback becomes generic, and, thus, is not possible to help the students in fine grained difficulties. In this context, we believe that the work proposed is an important step forward for ITSs because it is to our knowledge the first one that takes advantage of existing learning materials to provide fine grained support to students.

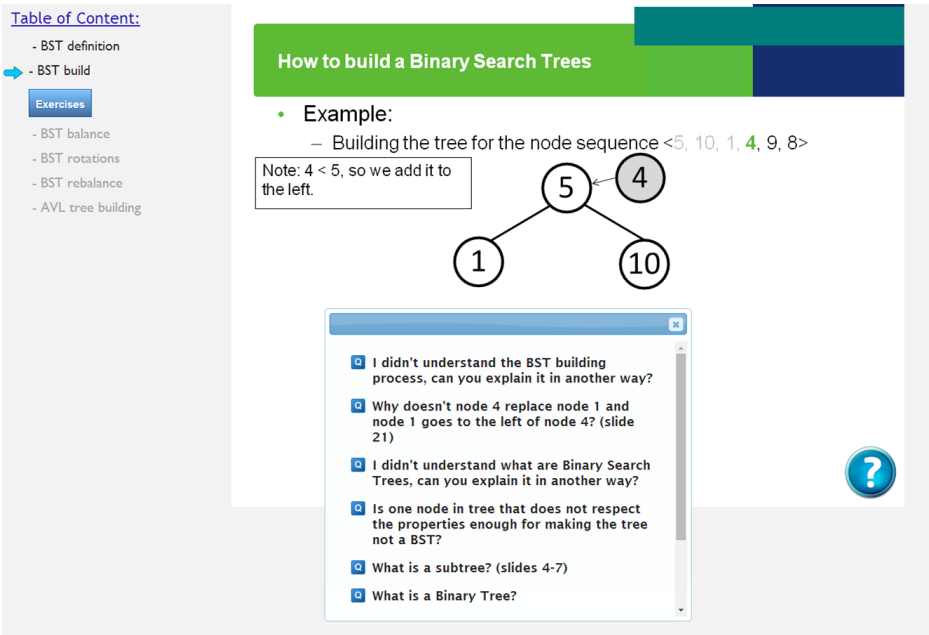

Figure 1: Screenshot of the web application for learning Adelson-Velskii and Landis' (AVL) trees.

3. LEARNING ENVIRONMENT

In our approach, we argue that ITSs should actively intervene during the period in which students are doing their self-studies, since the latter is a key point of the student's learning process. As such, it is essential to analyze how students make use of learning materials to surpass their difficulties during the self-study periods. Such analysis provides key insights both on the situations that students have difficulties with and on the learning materials they resort to in such situations. If an ITS can successfully model this type of interaction, great educational benefits can be achieved.

In this context, we developed a learning environment that enables the aforementioned analysis to be carried out. The topic presented to the students was the AVL trees data structure [1]. It is a topic currently taught in the first year of the Computer Science degree at Técnico Lisboa University, and one in which students generally have difficulties.

The students learn AVL trees by interacting with a web page (Figure 1). The main study material is composed by a set of 88 slides divided in 6 different subtopics:

- Definition of Binary Search Tree (BST)
- Building of BST
- Balanced BST
- Rotation of BST
- Re-balance of BST
- Building AVL trees

On the bottom right of the slides is a help button that, when pressed, pops up a window with a list of Frequently Asked Questions (FAQs) designed by the course instructor. The different available FAQs are organized by subtopic and are only shown in the window if the student reached that subtopic. Another characteristic of the FAQs is that they can either relate to a specific slide or to some general aspect within the corresponding subtopic.

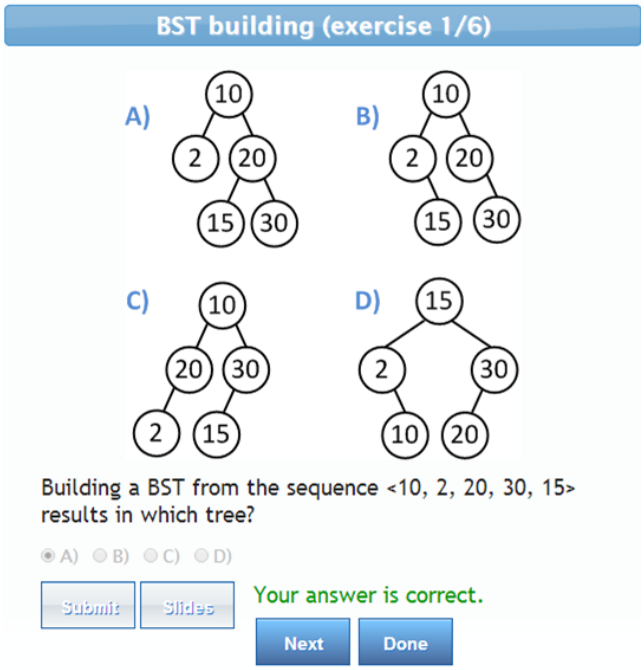

Figure 2: Example of a drill exercise.

The webpage also includes the possibility of performing drill exercises. Whenever a student goes over all slides within a certain subtopic, a message is presented indicating that a set of exercises will be presented for the student to solve. A total of 6 exercises is provided for each subtopic. An example of an exercise is shown in Figure 2.

All the exercises are multiple choice questions. After the student submits his answer, feedback that indicates whether it was correct one or not is provided. Students have multiple attempts to solve each exercise. Additionally, when solving exercises, students have the option of using a button that leads them back to the first slide of the current subtopic. Upon successfully concluding an exercise, the student can then choose to do the next exercise or to move on to the next subtopic. This possibility was provided to allow students with the same flexibility when using the web application that they have in their normal study.[1]

The last feature in the designed learning environment is the presence of a table of contents menu on the left side of the web application. The subtopics colored in black are the ones the student has already concluded or is currently studying. Students can click on these subtopics at any time, which leads them to the first slide of that subtopic. Also, in the current subtopic, students can click on the exercises button to immediately go to the drill exercises. The subtopics in gray are locked until the student completes at least one exercise from the previous subtopic, which means that the order of the subtopics to which the student is exposed is sequential.

4. EXTRACTION OF STUDY BEHAVIORS

Having developed the web interface for self-study described in Section 3, we allowed a total of 18 first year CS students to use it in their individual study. The students were invited to carry out their study in a room in which access to alternative materials was controlled, so that the data from each student's study process could be monitored as closely as possible. The interactions of the students with the interface were then used to extract and model the observed study behaviors. The resulting model was used to design and implement an ITS for self-study.

In the conducted study, we were particularly interested in monitoring the process by which students addressed their difficulties. Specifically, we want to detect when a student is faced with a drill exercise that it cannot immediately solve and revisits the learning materials to successfully address the exercise. Adequately modeling such behaviors is important because if an ITS is able to understand that a student is having difficulties in a given exercise and provides the appropriate learning material, significant learning gains can be expected, specially with students that would not try to go back to the learning materials.

Towards the goal of integrating the study behavior described above in our ITS, the first step is to extract the students' study behaviors and identify the one we are interested in. For the purpose of the work in the present paper, we focus on the subtopic of *Balanced BSTs*, the first subtopic that raises some real difficulties.

We categorized the individual slides provided to the students using the notion of Knowledge Components (KCs), which correspond to the smallest units of knowledge that are being taught (a specific definition or concept), or a combination of knowledge units (representing a more general principle or technique) [19]. The target subtopic was divided in 4 KCs:

- Know the height of an empty tree (KC_1)
- Know the height of a single node tree (KC_2)
- Calculate the height of a non-empty tree having more than one node (KC_3)
- Calculate the balance factor of a node (KC_4)

Each of the 25 slides documenting the subtopic were annotated with the corresponding KCs by the instructor, where each slide could address multiple KCs. Additionally, the KC identification procedure was not performed in the FAQ materials because students did not consult them in the analyzed subtopic.

After obtaining the KC annotations for the slides, the interactions corresponding to slide reviews were analyzed. These are the situations where students performed a review after which they were able to correctly answer the next exercise. In order to determine which KCs the student actually reviewed, an average of the time spent on each slide was computed, and those slides in which the student spent a time period inferior to that average were discarded. This filtering was performed because students seeking a specific part of the presentation will rush through the other slides until they reach the intended part. The resulting counts for each of the reviewed KCs are in Table 1, and they were extracted from 8 different students. In the table the *None* reference represents the situation where students needed to perform some review, but afterward they would provide the correct answer on the first attempt.

Table 1: Number of times students needed to review KC_i to get an exercise correct.

KC	# Visits
KC_1	5
KC_2	5
KC_3	5
KC_4	8
None	27

5. MODELING STUDY BEHAVIORS

We now describe how the study behaviors described in Section 4 were modeled using a decision-theoretic model– namely a POMDP. We provide a general overview of this model before introducing our proposed modeling.

5.1 MDPs for Intelligent Tutoring Systems

Formally, a Markov Decision Process (MDP) is a tuple $\{S, A, T, r, \gamma\}$. The set S represents the *state space*, each element of which (a *state*) describes a particular configuration of the system upon which a decision must be made. In our scenario, this state includes all relevant information about the student, corresponding to the traditional student model in the ITS literature [28]. The set A is the set of possible *actions* which, in our case, correspond to the tutorial actions that can be taken by the ITS. T represents the state transition probabilities. $T(s' \mid s, a)$ represents the probability of the system moving to a state s' at time step $t+1$ given that the state at time-step t was s and action a was taken. This probability is defined for each possible pair of states

[1]While some students may want to do as many exercises as they can, others prefer to do fewer exercises if they feel that they have successfully learned a certain topic.

and actions and then for each possible subsequent state. In our scenario, this represents the "learning dynamics" of the students, i.e., the probability of a student learning each particular KC upon being presented to the material prescribed by a particular tutorial action. The function r represents the *reward*, which encodes the goal of the decision process. In our case, this reward assigns a positive value every time a student shows learning progress. Finally, γ is a discount factor taking values in $[0, 1]$.

The defined MDP model describes a general stochastic environment in which it is possible to *act*, in the sense that the successive selection of actions determines (to some extent) how the state evolves. A *policy* π maps each state in S to an action in A. Upon fixing a policy π it is possible to compute the *value* of each state $s \in S$ as the expected sum of discounted rewards obtained when following policy π from the state s onward:

$$V^{\pi}(s) = r(s) + \gamma \sum_{s' \in S} T(s'|s, \pi(s)) V^{\pi}(s').$$

Solving an MDPs consists of determining the policy π^* that maximizes the value of each state. Such a policy is called the *optimal policy* and the associated value function is denoted as V^*. The optimal value function for a given MDP can be computed iteratively using the recursive relation

$$V^*(s) = \max_{a \in A} \left[r(s) + \gamma \sum_{s' \in S} T(s'|s, a) V^*(s') \right]$$

POMDPs [25] extend the MDP model to those situations in which the state of the system is not unambiguously observable. Instead, the state must be "tracked" from noisy observations. In the process of modeling the learning process of a student, POMDPs offer a more adequate modeling framework, since the knowledge acquired by a particular student cannot be directly observed, but must be tracked by the student's performance in the drill exercises.

A POMDP can be described as a tuple $\{S, A, O, T, \Omega, r, \gamma\}$, where S, A, T, r and γ are as in a MDP. The set O includes the *observations* available to the decision-maker (in our case, the ITS). Ω represents the *observation probabilities*. In particular, $\Omega(o \mid s, a)$ represents the probability of making observation o when at state s and action a was taken. Although computationally harder than MDPs, POMDPs can also be solved exactly using dynamic programming, or approximately using any of a wide range of available methods [24]. Most such methods involve tracking of the underlying state of the system in the form of a probability distribution known as the *belief*, since it roughly expresses, at each time-step, the decision-maker's belief about the underlying state of the system.

Using the previous framework, we obtain the following model:

- States are defined through the assignment of 0 or 1 values to four different state variables: $s = \{s_{KC_1}, s_{KC_2}, s_{KC_3}, s_{KC_4}\}$. Therefore, the state space S corresponds to all combinations of state variable assignments. Also, the state variables match the identified KCs and their value represents that students either know the KC or not.

- The set of possible actions is defined as $A = \{sc_{KC_1}, sc_{KC_2}, sc_{KC_3}, sc_{KC_4}, ex\}$. The sc_{KC_i} actions correspond to showing content to the student regarding the corresponding KC. These actions only represent a situation where the ITS has to provide learning in order to help the student, thus, it does not provide information about a particular learning material (there are several different slides related to the same KC). The ex action corresponds to give the student an exercise to solve. The fact that there is a single exercise action indicates that there is no distinction between the different possible exercises, since all of them address all KCs.

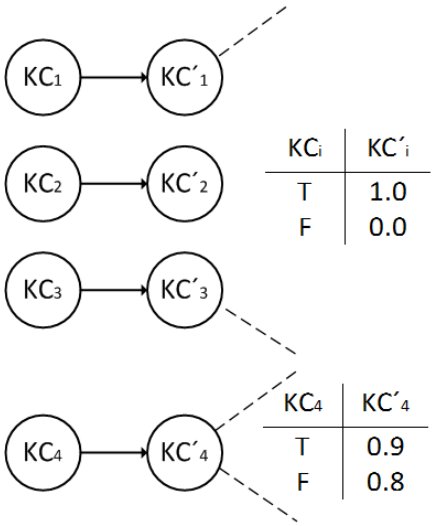

Figure 3: CPTs examples for the action that corresponds to showing content relative to KC_4.

- The observation model is defined as $O = \{correct, incorrect, none\}$. The first two observations can only be obtained when the action ex is taken, and correspond to the student's answers. The *none* observation is obtained when a sc_{KC_i} action is performed. It represents the fact that it is not possible to directly observe the impact of showing content to students.

- The reward model r provides a positive reward of 1 when the students are in the state that corresponds to knowing all the KCs.

5.2 Factored POMDPs

Since the POMDP model used a featured state-space S which can be factored into four subsets, each corresponding to one of the state variables sc_{K_i}, it is possible to leverage POMDP solution methods that take advantage of such factored representation [3].

In particular, the factorization of the POMDP allows expressing the state space in a more compact manner, by representing the action's effect using a two-slice temporal Bayes net [9]. This means that we have a set of nodes (one for each KC) representing the state prior to the action, another set of nodes representing the state after the action, and directed arcs that indicate the casual influence of the two sets of nodes. This influence must be defined for each different possible action. The adopted modeling approach is linear, in the sense that each state variable is only affected by itself, as it is possible to observe in Figure 3.

Table 2: Percentage of correct answers in the Balanced BST subtopic.

	Correct Answers
Exercise 1	85%
Exercise 2	80%
Exercise 3	83%
Exercise 4	78%
Exercise 5	56%
Exercise 6	72%

Linked to the expressed influence is the specification of a Conditional Probability Table (CPT) that encodes the system's dynamics. In Figure 3 there are two examples of CPTs for the sc_{KC_4} action. The CPT models in a probabilistic way the change of the values of the state variables when content relative to KC_4 is shown.

In our model these probabilities were computed from the data obtained in our study, namely the percentage of correct answers in each exercise (Table 2). From this data it is possible to observe that *Exercise 5* has a specially low percentage of correctness. The two most likely candidate KCs that explain this situation are KC_3 and KC_4, since in the exercise there is a significant increase in the size of the tree to be analyzed by the students. The course instructor argues that most students will have more trouble with the application of the formula in KC_4 rather than knowing the height of a tree (KC_3). Therefore, it was considered that it is harder to acquire KC_4 than KC_3, and, thus, in the CPT of action sc_{KC_4} the probability of the variable s_{KC_4} changing to 1 is lower than the probability of a similar transition in s_{KC_3} for the CPT of the sc_{KC_3} action (the concrete values are 0.8 and 0.85 respectively). These same values for the KC_1 and KC_2 were both set to 0.9 probability. This was due to the fact that if these KCs were hard to acquire it would be noticed in the percentage of correct answers in the first three exercises.

In what respects the CPTs for the exercise action, if $s_{KC_i} = 0$ the corresponding state variable will deterministically continue in that same state. For the opposite case, our main concern was to enable the system to shift its dynamics such that the sc_{KC_4} action is not too much frequently tried, and allow other actions to be explored. In this context, when $s_{KC_3} = 1$ the probability of making a transition to $s_{KC_3} = 0$ is 0.85. This same type of transition for $KC_{1,2}$ is set to 0.9, which again is related with percentage of correct answers in the exercises. Finally, for KC_4, the probability is set to 0.95.

The last CPTs that need to be defined are for the observation model. If the students know all KCs and an exercise action is made, then a correct answer is observed with 0.9 probability (the student can always make a mistake). For the case that students do not know some KC, an incorrect answer is observed with 0.75 probability, which corresponds to a random guess, since there are four possible answers. When the action is of the type sc_{KC_i}, then the *none* observation is always obtained.

Although the number of possible state is just $2^4 = 16$, we still opted for the factored POMDP approach in order to assess its effectiveness, so that we can determine if it is an option for the full AVL domain, in which the number of possible states is actually a problem (there are 19 KCs).

6. EXPERIMENTS

In the following sections, the experimental setup that defines the learning environment for an ITS to act and the corresponding obtained results are described.

6.1 Experimental Setup

The scenario for the first experiments uses the study behaviors extracted in Section 4 to create tutorial situations where the ITS must perform the appropriate action. These situations correspond to the interactions students made until they provide a correct answer. If the student did not perform any review, and the ITS gives the student an exercise to solve, he/she will provide the correct answer. In case the student performed some review, he/she will only get the exercise correct after the appropriate show content actions are provided by the ITS, which corresponds to the actions that match the reviewed KCs. Using this setup it is possible to calculate the minimum number of actions that an ideal ITS would do.

In order to have a baseline comparison, a scripted ITS was designed to act in the defined learning environment. Such ITS always performs the same sequence of actions: it proposes an exercise to the student. If the student fails, a sc_{KC_i} action takes place, followed by another exercise. The order of the different actions was defined beforehand by the instructor, according to his own perception of what KCs usually present more difficulties of understanding and also by analyzing the data regarding the percentage of correct answers in the exercises (Table 2). This order is summarised as KC_4, KC_3, KC_2, KC_1. We then used the Symbolic Perseus software [20] to solve the POMDP obtained from the student model described in Section 5. The POMDP was initialized to a pre-defined initial belief state, in which the student is assumed to know all KCs except KC_4, reflecting the fact that we are analyzing students who at some point made a review and also because a significant percentage of incorrect answers to the exercises were caused by this KC.

6.2 Experimental Results

In this section, the results obtained using the previous experimental setup are reported. The comparison between the percentage of actions, regarding the optimal ITS, used by the baseline and POMDP ITSs is in Figure 4. In this context, the goal of the compared ITSs is to achieve a 100% score, meaning that they would have used an optimal number of actions. The results show that the POMDP consistently obtained a better performance than the baseline with the exception of *Student#8*. The last column of Figure 4 shows that the baseline spent, in total, twice more actions than the optimal strategy, whereas the POMDP model only used 140% of the actions - 28 less actions than the baseline, and 27 more than optimal. Also, the non-overlapping error bars indicate that these results are statically significant.

When inspecting the reviewed KCs it is possible to observe that the majority of them include all 4 KCs. This happens since the first slide contains the formal definitions of all KCs. In this situation, the POMDP model has a better teaching policy, because if the student still gives wrong answers after an action of showing content, it uses consecutive show content actions of the remaining KCs before trying another exercise, which proved to be more suitable than always alternating between exercise and show content actions.

Another important aspect to mention from the used data

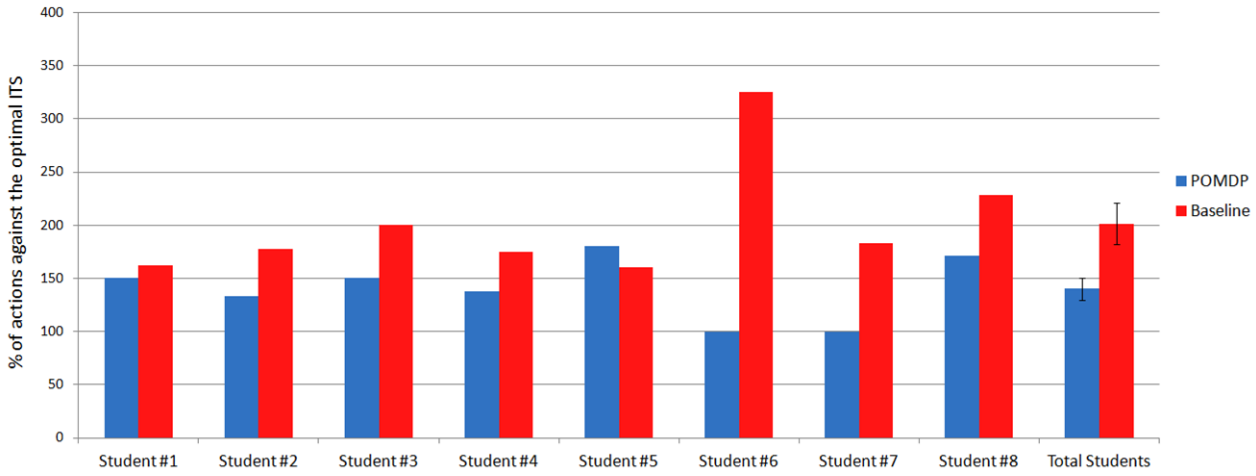

Figure 4: Performance comparison between the baseline and the POMDP ITS.

is that in order to achieve optimal performance it is necessary to take into many account different states trajectories, since students might have difficulties in distinct sets of KCs. In fact only two students showed the exact same behavior throughout the exercises of the analyzed subtopic. Some of these behaviors are particularly hard for scripted strategies to cope with, such as having students reviewing some KC to get the current exercise right, but in the next one students still review other KCs, even though all exercises require all KCs in order to be solved. For these cases the POMDP approach is advantageous since it automatically computes different teaching policies to be followed according the current belief-state of the student.

7. CONCLUSIONS AND FUTURE WORK

In this work we presented a methodology for studying students self-studies behaviors, with the ultimate goal of bootstrapping an ITS that can learn what is the appropriate tutorial action to take in each particular situation. This problem was tackled from a perspective of identifying what KCs the student has not mastered and guiding them to the appropriate learning materials.

The preliminary evaluation in one of the subtopics of the domain showed that the POMDP approach was more efficient than the baseline version. This result is important since it shows that a fix teaching policy, even when provided by a human expert, is not optimal and also that there are other sub-optimal policies, such as the one presented in this work, that can achieve better performance. The explanation for this is that the optimal teaching paths are different from student to student, and, thus, they have different levels of mastery of the KCs when starting to solve the exercises. This leads us to the conclusion that there are expectable performance gains to be obtained if the initial belief state of the model is not the same for all students, as it is currently. In order to calculate a more adequate initial belief state, one possibility is to use the information regarding the time students spent in each KC prior to the exercises.

Another important conclusion from this work is that it is indeed possible to alleviate, to some extend, the authoring effort required to design ITSs. This was done by using a POMDP model that automatically computes a teaching policy for every possible state and also provides a mechanism to update the estimation of what state the student is in according to the gathered observations. Obtaining such automation is important since it deals with a time consuming and hard task of the system's design. This is related with the fact that although the expert is able to provide an order of difficulty for a set of KCs, specifying some concrete teaching policy to follow when certain conditions are met is a much harder task. It should also be noted that just defining the KC difficulty order is not a trivial task.

As mentioned previously, the need of authoring work as not been eliminated completely. Namely, it is necessary to specify a KCs structure to back up the POMDP model. This structure can be complex and there are multiple possibilities for both the shape of the structure and what are the actual KCs. Even for our simple domain this type of doubt was certainly felt. In tackling this challenge, the use of Natural Language Processing might allow to obtain the KCs structure automatically. Another important improvement would be to make the CPTs estimation data driven, so that this process becomes automatic and possibly closer to optimal.

Regarding future work, it is necessary to address the problem of which learning materials should be shown to the student. This cannot be provided by our approach as multiple materials might relate to the same KC. A possible solution to this problem is to again use a POMDP model, similarly to what has already been done in this work. Finally, we may also perform the evaluation in the context of the full AVL topic in order to assess how our approach scales.

8. ACKNOWLEDGMENTS

The authors thank Maxine Eskenazi for the insights on the setup of the learning environment to perform our study. We also thank Hugo Rodrigues for his thorough beta testing of the AVL web application. Finally, we thank Ana Paiva and Francisco Santos for making possible to carry out the study with Técnico Lisboa University students. This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with both the reference UID/CEC/50021/2013 and the project under CMUP-

ERI/HCI/0051/2013, and also through the Carnegie Mellon Portugal Program under Grant SFRH/BD/51917/2012.

REFERENCES

[1] G. Adelson-Velsky and E. M. Landis. An Algorithm for the Organization of Information. *Doklady Akademii Nauk USSR*, 146(2):263–266, 1962.

[2] I. Arroyo, C. R. Beal, T. Murray, R. Walles, and B. P. Woolf. Web-Based intelligent multimedia tutoring for high stakes achievement tests. In *Proc. 7th Int. Conf. Intelligent Tutoring Systems*, pages 468–477, 2004.

[3] C. Boutilier and D. Poole. Computing optimal policies for Partially Observable Decision Processes using compact representations. In *Proc. 13th AAAI Conf. Artificial Intelligence*, pages 1168–1175, 1996.

[4] E. Brunskill and S. J. Russell. RAPID: A reachable anytime planner for imprecisely-sensed domains. In *Proc. 26th Conf. Uncertainty in Artificial Intelligence*, pages 83–92, 2012.

[5] M. Chi, K. VanLehn, and D. Litman. Do micro-level tutorial decisions matter: Applying reinforcement learning to induce pedagogical tutorial tactics. In *Proc. 10th Int. Conf. Intelligent Tutoring Systems*, pages 224-234, 2010.

[6] P. Cohen, C. Beal, and N. Adams. The design, deployment and evaluation of the AnimalWatch intelligent tutoring system. In *Proc. 18th Eur. Conf. Artificial Intelligence*, pp. 663–667, 2008.

[7] A. T. Corbett, K. R. Koedinger, and J. R. Anderson. Intelligent Tutoring Systems In M. Helander, T.K. Landauer, and P. Prabhu (Eds.), *Handbook of Human-Computer Interaction*, pages 849–874. Elsevier, 1997.

[8] L. Daubigney, M. Geist, and O. Pietquin. Model-free POMDP optimisation of tutoring systems with echo-state networks. In *Proc. 14th SIGDial Meeting on Discourse and Dialogue*, pages 102–106, 2013.

[9] T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150, Dec. 1989.

[10] K. Forbes-Riley and D. J. Litman. Investigating human tutor responses to student uncertainty for adaptive system development. In *Proc. 2nd Int. Conf. Affective Computing and Intelligent Interaction*, pages 678–689, 2007.

[11] A. C. Graesser, R. Vasile, and D. Sidney. *AutoTutor: learning through natural language dialogue that adapts to the cognitive and affective states of the learner.* In D.H. Robinson and G. Schraw (Eds.), *Recent Innovations in Educational Technology that Facilitate Student Learning*, pages 95–125. Information Age Publishing, 2008.

[12] J. I. Lee and E. Brunskill. The impact on individualizing student models on necessary practice opportunities. In *Proc. 5th Int. Conf. Educational Data Mining*, pages 118–125, 2012.

[13] B. Clement, P.-Y. Oudeyer, D. Roy, and M. Lopes. Online optimization of teaching sequences with multi-armed bandits. In *Proc. 7th Int. Conf. Educational Data Mining*, pages 269–272, 2013.

[14] K. N. Martin and I. Arroyo. AgentX: Using reinforcement learning to improve the effectiveness of intelligent tutoring systems. In *Proc. 7th Int. Conf. Intelligent Tutoring Systems*, pages 564–572, 2004.

[15] P. Martínez and A. García-Serrano. On the automatization of database conceptual modelling through linguistic engineering. In *Proc. 5th Int. Conf. Applications of Natural Language to Information Systems*, pages 276–287, 2000.

[16] W. Min, E. Y. Ha, J. Rowe, B. Mott, and J. Lester. Deep Learning-Based Goal Recognition in Open-Ended Digital Games. In *Proc. 10th Annual Conf. Artificial Intelligence and Interactive Digital Entertainment*, pages 37–43, 2014.

[17] T. Murray. Authoring intelligent tutoring systems: An analysis of the state of the art. *Int. J. Artificial Intelligence in Education*, (10):98–129, 1999.

[18] Z. A. Pardos, Y. Bergner, D. T. Seaton, and D. E. Pritchard. Adapting Bayesian Knowledge Tracing to a Massive Open Online Course in edX. In *Proc. 6th Int. Conf. Educational Data Mining*, pages 137–144, 2013.

[19] L. O. Pittsburgh Science for Learning Center. Knowledge component. Available: http://www.learnlab.org/research/wiki/index.php/Knowledge_component, 2011. Accessed: 2014-11-17.

[20] P. Poupart. *Exploiting Structure to Efficiently Solve Large Scale Partially Observable Markov Decision Processes*. PhD thesis, Univ. Toronto, 2005.

[21] J. Reye. Student modelling based on belief networks. *Int. J. Artificial Intelligence in Education*, 14(1):63–96, Jan. 2004.

[22] S. Ritter, J. R. Anderson, K. R. Koedinger, and A. Corbett. Cognitive Tutor: Applied research in mathematics education. *Psychonomic Bulletin & Review*, 14(2):249–255, 2007.

[23] R. Sedgewick and K. Wayne. *Algorithms, 4th Edition.* Addison-Wesley, 2011.

[24] G. Shani, J. Pineau, and R. Kaplow. A survey of point-based POMDP solvers. *Autonomous Agents and Multi-Agent Systems*, 27(1):1–51, 2013.

[25] S. P. Singh. Learning without state-estimation in Partially Observable Markovian Decision Processes. In *Proc. 11th Int. Conf. Machine Learning*, pages 284–292, 1994.

[26] J. R. Tetreault. Using reinforcement learning to build a better model of dialogue state. In *Proc. 11th Conf. European Association for Computational Linguistics*, 2006.

[27] M. Travis, L. Yun-En, L. Sergey, B. Emma, and P. Zoran. Offline policy evaluation across representations with applications to educational games. In *Proc. 13th Int. Conf. Autonomous Agents and Multiagents Systems*, pages 1077–1084, 2014.

[28] K. VanLehn. The behavior of tutoring systems. *Int. J. Artificial Intelligence in Education*, 16(3):227–265, 2006.

[29] K. Vanlehn, C. Lynch, K. Schulze, J. A. Shapiro, R. Shelby, L. Taylor, D. Treacy, A. Weinstein, and M. Wintersgill. The Andes Physics tutoring system: Lessons learned. *Int. J. Artificial Intelligence in Education*, 15(3):147–204, 2005.