Model and Evaluation of an Intelligent Tutoring System with a Reinforcement Learning Agent

Vitor Gouvêa[1], **Amanda C. N. de Oliveira**[2], **Daniel R. Figueiredo**[2],

[1] Instituto de Matemática – Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro – Brazil

[2]Programa de Engenharia de Sistemas e Computação (PESC)
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro – Brazil

vitor.martins@matematica.ufrj.br,

{amandacno, daniel}@cos.ufrj.br

Abstract. *The use of Reinforcement Learning (RL) in completing human tasks has exploded in recent years, with RL models often beating human experts and achieving record performance. A particularly interesting application of RL is in the Intelligent Tutoring Systems (ITS) problem of sequential learning. In this problem the ITS has to decide which questions to give to a student during an exercise session. In this paper, we model the sequential learning problem as a Markov Decision Process (MDP) environment and estimate an optimal policy using a model-free Monte Carlo algorithm from simulations. We compare the optimal policy to other trivial and intuitive policies and show that such a policy can achieve good performance, however it is limited by state representation.*

1. Introduction

Reinforcement learning (RL) has played an important role in some of the recent advances in AI [Guo et al. 2025], being especially useful in sequential tasks where there is clear reward structure, such as games or robotics [Kaelbling et al. 1996]. One of the areas where it has been readily applied is that of Intelligent Tutoring Systems (ITS) [Doroudi et al. 2019], an umbrella term for various computer-based teaching technologies and problems. One of the several problems that ITS aim to solve is that of instructional sequencing.

A typical instructional sequencing problem, as described by [Atkinson 1972], can be formulated as follows: the student is modeled as a stochastic environment with which the learning system interacts. The system selects instructional actions and observes the resulting outcomes as the student performs them, using them to select the next action. This kind of system-student interaction has the benefit of being highly responsive to any difficulties the student may present, as well as being personalized to the student's individual needs [Nie et al. 2023, Ruan et al. 2024].

From this formulation arise several natural questions: how should the system process the information it receives, how should this information be used to select future exercises, how can student learning be quantified, and how can the interaction between

a student and the system be modeled? In particular, we investigate whether by using Reinforcement Learning we can produce better educational outcomes than heuristic instructional policies.

To investigate these questions we propose a mathematical model of a student's interaction with an instructional sequencing environment, inspired by MathSpring [Arroyo et al. 2014], in which a student enters an exercise *session* and receives questions of varying difficulties, returning two outcomes: whether the student gave a correct answer, and whether the session will be terminated at that point, or if another question will be answered. We model this student-system interaction as a Reinforcement Learning problem and use Q-Learning [Watkins and Dayan 1992] to find the optimal policy. We find that our initial choice of state space is insufficient for optimality and that the resulting policy can underperform for certain types of students. We then show that including partial information about the student enables the system to achieve better performance.

The paper is structured as follows. Section 2 gives an overview of RL and the Q-Learning algorithm, as well as a short review of ITS and sequential learning. Section 3 details the designed system model, inclusing the student model and our Reinforcement Learning representation of the problem. Section 5 exposes our experiments and results. We end with a recap of the paper in Section 6.

2. Background and Related Works

2.1. Reinforcement Learning

The general Reinforcement Learning (RL) setup in the context of MDP's can be defined, as in [Sutton and Barto 2018], by an *environment*, and an *agent* that interacts with the environment. This system is typically defined by a *state space* S of all possible states that can happen, an *action space* $\mathcal{A}$ that are all available actions that our agent has at his disposal, and a *reward function* $R : S \times \mathcal{A} \to \mathbb{R}$ that represents the goals that the Agent has. The environment dynamics, or transition probabilities, are also assumed to be Markovian (i.e. $\mathbb{P}(s_{t+1} \mid a_t, s_t, \ldots, a_0, s_0) = \mathbb{P}(s_{t+1} \mid a_t, s_t)$).

The agent on the other hand follows a *policy* π that can be learned by a *learning algorithm*. A policy is a probability function over the action space $\mathcal{A}$, conditioned on the state space $\mathcal{S}$. There are several RL algorithms that can improve the performance of an agent's policy by interacting with the environment.

In most formulations of MDP's the learning algorithm of the *agent* tries to maximize either value functions $V^\pi(s)$, defined as:

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k} \middle| S_t = s \right] , \tag{1}$$

or the action-state value function $Q^\pi(s, a)$, defined as:

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k} \middle| S_t = s, A_t = a \right] , \tag{2}$$

where $s \in \mathcal{S}$, $a \in \mathcal{A}$, R_t is the reward at time t. Note the dependence on the policy π, since the transition probabilities depend on the actions taken, which are given by the policy.

A popular method for improving the policy by maximizing the state action value function is trough Q-Learning [Watkins and Dayan 1992], a model-free RL algorithm that estimates the optimal state-action value function $Q^*(s, a)$ directly through interaction with the environment. Rather than requiring knowledge of the transition probabilities, the algorithm updates its estimates using observed rewards and state transitions, gradually improving its approximation of the optimal policy. In our instructional sequencing problem, Q-Learning is used to learn which difficulty level should be presented to a student in each state of the tutoring environment.

2.2. Intelligent Tutoring Systems

The umbrella term Intelligent Tutoring Systems encompasses numerous computer-based techniques and approaches for teaching [Doroudi et al. 2019]. One of the earliest machine learning-based works in this direction introduced a "Theory of Instruction" [Atkinson 1972], a mathematical formulation for intelligent tutoring systems consisting of specified learning states, actions, rewards, and objectives, which naturally leads to an MDP where optimal policies (that is, choices that lead to high rewards) can be determined. In this scenario, the student is the random environment with which the tutoring system interacts: the system chooses learning actions and observes the outcome as the student executes the action, a problem known as instructional sequencing.

RL has been successfully applied to instructional sequencing and other problems within ITS [Doroudi et al. 2019, Riedmann 2025]. For example, [Atkinson 1972] proposes a simple markov-chain model for learning German vocabulary words which improved the learning of students with no prior knowledge of the language.

On the other hand, [Sen et al. 2018] train a neural network (NN) to imitate human learning and uses it to selects the best instructional policies for perceptual fluency in the identification of chemical molecules that induce desirable difficulties for perceptual learning, with a low accuracy during training but higher post-test accuracy.

Also, [Chi et al. 2010] solves MDP's with different reward functions using dynamic programming techniques and student interactions, showing that tutorial decisions (policies that determine the sequence of activities that are submitted to a student) make a difference in students' learning performance.

Similarly, [Ruan et al. 2024] models the student-agent interaction as an MDP an trains a NN based RL agent to give hints and helpful feedback to the student in a interactive interface aimed at children in grades 3-5.

3. System Modelling

We propose an instructional sequencing environment inspired by the MathSpring math tutor [Arroyo et al. 2014]. In our setting, each student is presented with exercises that differ only in their *difficulty*, ranging from 1 to D. In this environment, the student initially enters what we will refer to as an *exercise session*, hereafter shortened to simply *session*.

Upon being presented with an exercise, the student interacts with the tutoring system in two ways. First, the student answers the exercise, either correctly or incorrectly. Second, the student decides whether to terminate the session or continue and attempt additional exercises. This exercise session goes on indefinitely until the student exits the session. The length of a session, measured by the total number of questions answered, will be denoted by T, while $t \in \{1, T\}$ refers to a specific question that was answered.

3.1. Student Model

Since both forms of interaction are binary, we model the student's behavior when presented with an exercise of difficulty level d as a pair of Bernoulli random variables $(X_{V_t,F_t}, Y_{h,d})$, where $Y_{h,d}$ denotes whether the student correctly answered the exercise of difficulty d, and X_{V_t,F_t} denotes the student's decision to either remain in or terminate the session after t questions have been answered.

Note that the variable X_{V_t,F_t} depends on V_t and F_t, which denote, out of the t questions answered, the number that were answered correctly and incorrectly, respectively. This dependence is natural, since a student's propensity to terminate the session may vary according to their history of successes and failures throughout the session. We model this dependence through the function $q(V_t, F_t)$, defined in Eq. 3, which gives the probability that a student chooses to remain in the session at time t. This function is composed of two components, $e^{-\lambda_1 F_t}$ and $e^{-\lambda_2(V_t+F_t)}$, which we plot on Fig. 1. Note that the second component can be written in terms of the session length t, since it follows from their definition that $V_t + F_t = t$.

$$X_{V_t,F_t} \sim Bernoulli(q(V_t, F_t)), \quad \text{with} \quad q(V_t, F_t) = \min\{e^{-\lambda_1 F_t}, e^{-\lambda_2(V_t+F_t)}\} \tag{3}$$

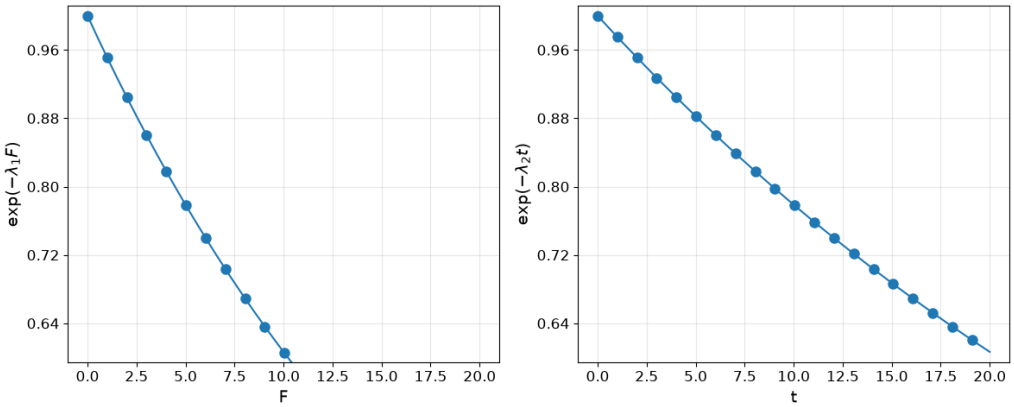


Figura 1. Components of the probability function $q(V_t, F_t)$ for $\lambda_1 = 0.05$ and $\lambda_2 = 0.025$. On the left, the first component is showcased, which decreases the probability of continuing the section with the increase in the number of errors. On the right, the second component decreases the probability of continuing the section with the increase in the total number of questions already answered.

As can be seen in Fig 1, each component of the probability $q(V, F)$ of staying in the session is decaying in their respective variables F_t and t, with the first component

decreasing faster then the second. This speed of decrease, parameterized by λ_1 and λ_2, can be interpreted as the student's *sensitivity* to incorrect answers and session length, respectively. Intuitively, this models a student becoming progressively fatigued throughout the session, with incorrect answers accelerating this effect.

The variable $Y_{h,d}$ on the other hand, does not depend on V_t and F_t, instead depending on the difficulty level of the exercise being solved, i.e. d, as well as an *ability* parameter h. The dependence on the difficulty level of the exercise is natural, since $Y_{h,d}$ represents the response of the student to the exercise. The ability parameter, on the other hand, was introduced to add heterogeneity to the student model and can be interpreted as the prior knowledge of each student on the particular subject. This dependence is modelled through the $p_{s,d}(h)$ functions defined in Eq 4 and visualized in Fig 2, which model the probability that a student of ability h correctly answers an exercise of difficulty d. The function also has an shape parameter α that controls the slope of the sigmoid function.

$$Y_{h,d} \sim \text{Bernoulli}(p_d(h))\,, \qquad p_d(h) = \frac{1}{1+e^{-\alpha(h-c_d)}}, \quad h \in [0,1]. \tag{4}$$

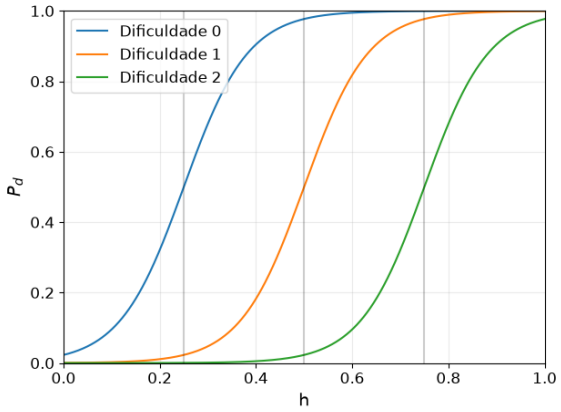


Figura 2. The probability p_d of correctly answering a question of difficulty level d as a function of the student's ability h, for $d = 1, 2, 3$, with $\alpha = 15$

The parameters c_d are the centers of the sigmoid functions $p_d(h)$, $d = 1, \ldots, D$ and are chosen so that they are centered at uniformly spaced points in the interval $[0, 1]$, more precisely, $c_d = d/(D+1)$.

As shown in Fig. 2, the probability that a student correctly answers an exercise of difficulty level d, denoted by $p_d(\cdot)$, increases with the student's ability. Additionally, the easiest difficulty level has the highest probability of being answered correctly, with most students exhibiting a high chance of success. In contrast, the hardest difficulty level has a generally low probability of being answered correctly, with only high-ability students having a substantial chance of success.

The curves shown in Fig. 1 and 2 correspond to the parameter values used throughout this paper. Specifically, we fix the parameters from Eq. 3 and Eq. 4 as $\lambda_1 = 0.05$, $\lambda_2 = 0.025$, $s = 15$, and $D = 3$. Since we have $D = 3$ difficulty levels, we may also refer to $d = 1, 2, 3$ as *easy*, *medium*, and *hard* exercises, respectively.

3.2. Environment

Having defined the student's behavior, we now specify the state-action-reward triplet $(\mathcal{S}, \mathcal{A}, R)$ through which the tutoring problem will be modeled.

We represent the state of the environment as $s_t = (l_{1t}, e_{1t}, l_{2t}, e_{2t}, \ldots, l_{Dt}, e_{Dt})$ where l_{dt} denotes the number of questions of difficulty d answered correctly by the student, and e_{dt} denotes the total number of questions of difficulty d attempted. Therefore, in our experiments the state will constitute of 6 elements: $s_t = (l_{1t}, e_{1t}, l_{2t}, e_{2t}, l_{3t}, e_{3t})$.

The state can be interpreted as a summary of the student's history, although it does not retain the order in which questions were answered. Notably, the student's ability is not included in the state representation, although it could be theoretically inferred from the state if the student interacted with the environment long enough. As we show in Section 5, representing the student's ability in the state has important implications for the performance of the resulting policy.

Having defined the state, we next specify the actions available to the agent. Given that the agent acts as a tutor in this setting, we define the action space as $\mathcal{A} = \{1, 2, \ldots, D\}$. That is, each action corresponds to selecting the difficulty level of the next question presented to the student. In our scenario, that means there are 3 possible actions to be taken by the agent.

To encourage the tutor to present challenging exercises, we define the reward as: $r_t(s_t, a_t) = I_t \cdot a_t$ where I_t is an indicator variable equal to 1 if the student's answer at time t is correct and 0 otherwise, and a_t denotes the difficulty level selected by the agent.

4. Experimental Setup

To estimate the optimal policy, we employ the Q-Learning algorithm mentioned in Section 2.1. Training is performed by generating Sn students with abilities drawn iid from $\mathcal{U}([0, 1])$ and then generating M trajectories for each student, updating the policy after each trajectory as summarized in Algorithm 1. Note that in algorithm 1 we define T as the lenght of the trajectory generated by the policy π, which is updated in a ϵ-greedy fashion.

In addition to the student model parameters from Section 3.1 and the training hyperparameters from Algorithm 1, we introduce two new parameters. The first is *max_episodes*, which limits the state-space size by forcefully terminating an episode once it reaches a given length (i.e., setting $X_{F_t,V_t} = 0$ when the episode length equals *max_episodes*, where X_{F_t,V_t} is defined in Section 3.1). The second is *min_episodes*, which delays the effect of correct and incorrect answers on the stopping probability $q(V, F)$ by only allowing the exponentials in $q(V, F) = \min\{e^{\lambda_1 F_t}, e^{\lambda_2 t}\}$ to take effect after the episode exceeds *min_episodes*. Until then, the student's stopping probability remains unaffected by the accumulated incorrect answers F_t and the elapsed time t.

The *max_episodes* parameter was introduced to ensure a finite state space, with its value chosen due to computational constraints. The *min_episodes* parameter was introduced to emulate situations in which students are required to answer a minimum number of questions before being allowed to terminate the session.

Algorithm 1 Training Loop

Initialize Sn, M, γ, ϵ, $Q = \mathbf{0}$, $N = \mathbf{0}$, $\pi = \epsilon$-greedy(Q)
for student $i = 1$ **to** Sn **do**
    Sample $h \sim \mathcal{U}(0, 1)$
    Create student with proficiency h
    for sample $m = 1$ **to** M **do**
        Generate episode by following π: $\{(S_t, A_t, R_t)\}_{t=0}^{T-1}$
        $G \leftarrow 0$
        for $t = T - 1$ **downto** 0 **do**
            $G \leftarrow \gamma G + R_t$
            $N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$
            $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{G - Q(S_t, A_t)}{N(S_t, A_t)}$
            $\pi(\cdot|S_t) \leftarrow \epsilon$-greedy$(Q(S_t, \cdot))$
        end for
    end for
end for
return $\pi_* = \arg\max_a Q(\cdot, a)$

Except when explicitly defined, the parameters used in the following sections are:

- Discount factor $\gamma = 1$
- Epsilon $\epsilon = 0.7$
- Number of students $Sn = 2000$
- Number of sessions per student *n_samples*$= 1000$
- Maximum session length *max_episodes*$= 20$
- Minimum session length *min_episodes*$= 0$
- Total number of difficulties $D = 3$

We also define 5 new policies to contrast against the optimal one found in the training phase:

- **Simple Policy**: If the student got a question of difficulty d wrong, offers a question of difficulty $d - 1$. If the student got it right, offers a question of difficulty $d + 1$. If the question that was correctly answered was the maximum difficulty one (i.e. 3), the policy offers 3 again, and analogously for the answering wrong a lowest difficulty question (i.e. 1).
- **Uniform Policy**: Gives all 3 questions with equal probability, independently of the state.
- **Easy Policy**: Gives only questions of difficulty 1
- **Medium Policy**: Gives only questions of difficulty 2
- **Hard Policy**: Gives only questions of difficulty 3

The evaluation loop was the same as the training one, with the main differences being that we do not update the policy, and that each h was picked as $h = i/Sn, i = 1, \ldots, Sn$. This was done to have a more controlled view of the each policies behaviour. In the following Sections we analyze the mean return and duration of exercise sessions over *n_samples* samples. The parameters used in evaluation were the same as training.

5. Experimental Results

5.1. Trivial Policies

We begin by analyzing the performance of the *trivial policies*, namely the Easy, Medium, and Hard policies defined in Section 4.

In Fig 3 we plot the mean return of each policy as a function of the student's ability, that is, the average return G that is simply defined as the sum of the rewards over a session. As we can see, higher ability often implies higher average return, since student's with high ability have a higher chance of answering questions correctly. Also note that each policies return saturates at different levels, with the saturation point being related to the average episode duration seen in Fig 4. Another thing to note is that all policies achieve the same return for student's with ability close to 0, since these student's have an extremely low probability of getting questions right.

The average duration as a function of the student's ability, seen in Fig 4, exhibits similar, albeit slightly different behaviour, with all policies saturating at the same point, but at different ability levels. Recalling the interpretation of the student model in Sec 3.1, Fig 4 shows that, by giving exercises of a higher ability level then the student is able to answer, said student quickly gets frustrated and exits the session, while giving him appropriate exercises can retain his attention for longer.

Note that there is a trade-off between return and duration. The easy policy, for example, has higher average duration, particularly for lower ability students, while also having lower return for medium to high ability students. This trade-off between average return and average duration, where higher difficulty exercises may obtain a higher return but may also frustrate the student and end the session earlier, diminishing the total return, is exactly what the policy found by the agent must optimize.

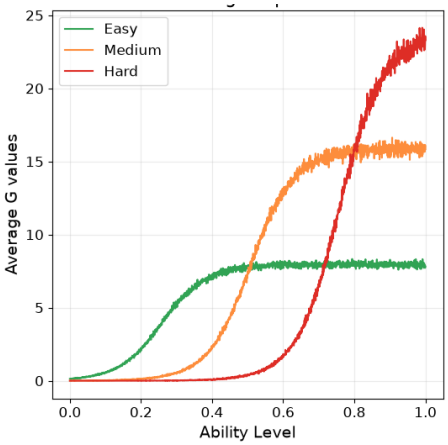


Figura 3. Average session return (G) for the trivial policies as a function of the student's ability level.

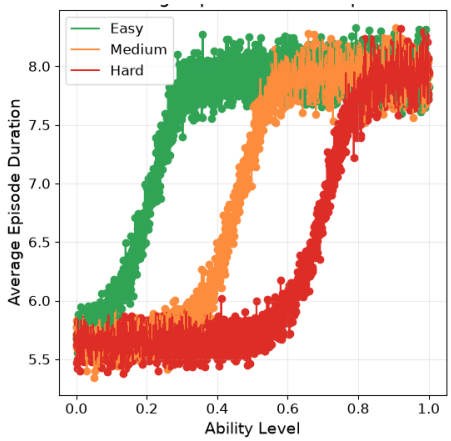


Figura 4. Average session duration for the trivial policies as a function of the student's ability level.

Looking at Figs 3 and 4, we see that each policy appears to perform best on a specific subset of students, showcasing that student ability plays an important role in

determining which policy is most effective.

Although these policies are useful for understanding the student model and tutoring environment, and even achieve good performance for their respective range of students' abilities, their effectiveness depends on the ability parameter h. Is it possible to outperform on average these trivial policies without any information about the student's ability?

5.2. Non-Trivial Policies

In order to answer this question, we evaluate the simple and random policies defined in Section 4, as well as the optimal policy found by our agent.

Figure 5 shows that the policy learned by the agent achieves a higher average return than the simple and random policies for students with low to medium ability, while the simple policy performs better for students with high ability. We can also see that both the optimal and simple policy approximate the returns of the hard policy for higher ability students. Figure 6 shows that the learned policy also achieves a higher average duration for students with low to medium ability. With all policies saturating at the same value as in Fig 4.

At first glance, these results appear contradictory, since the learned policy is expected to outperform all other policies in terms of average return. Nevertheless, this result makes sense when we realize that the simple policy has *more information* than the optimal policy. Recall its definition from Section 4: the simple policy has access to both the current state s_t and the previous state s_{t-1}, since it knows which type of question was answered and whether the student was correct. In contrast, the agent only has access to the current state s_t, defined in Section 3.2.

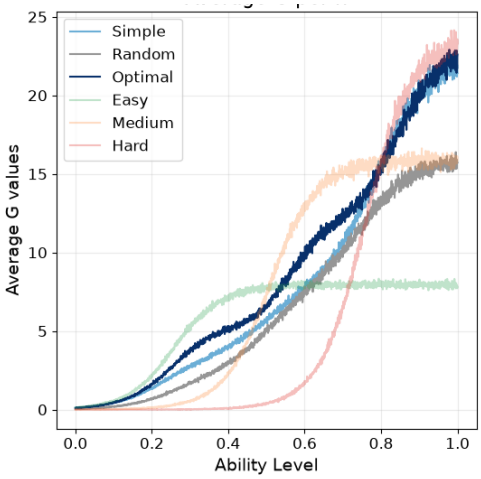


Figura 5. Mean session return for the trivial and non-trivial policies as a function of student ability.

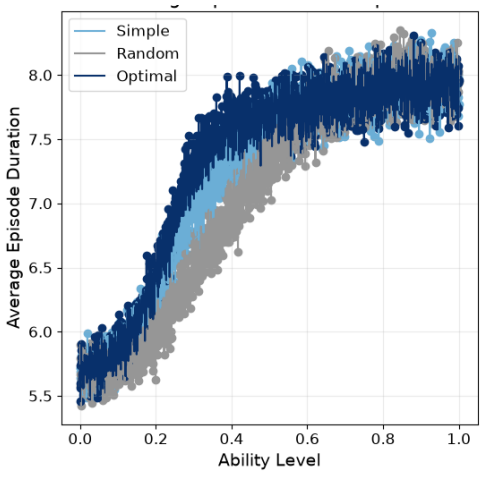


Figura 6. Average session duration for trivial and non-trivial policies as a function of the student's ability level.

Up until now, we have been concerned with the mean return and duration conditioned on a given student, i.e. $\mathbb{E}[G \mid h]$ and $\mathbb{E}[T \mid h]$, respectively. This allows for a

more precise evaluation of our policies, letting us see how each performs for each type of student. However, it is also informative to consider the unconditional averages $\mathbb{E}[G]$ and $\mathbb{E}[T]$. Using the tower property of expectations [Bertsekas and Tsitsiklis 2008], we calculate these marginalized averages directly from the data in Figures 5 and 6, showcasing them on Table 1. These results indicate that the learned policy achieves the highest marginalized return and the second-highest marginalized duration, being surpassed only by the easy policy in terms of duration.

Tabela 1. Policy Performance Comparison

Policy	Unconditional Mean Return	Unconditional Mean Duration
Simple	7.93	7.24
Random	6.43	7.07
Optimal	8.54	7.34
Easy	5.86	7.49
Medium	7.78	6.92
Hard	5.76	6.34
Extended Optimal	9.65	7.35

Despite achieving the best overall performance in terms of unconditional return, the policy learned by the agent still underperforms for students with high ability, with the trivial policies outperforming it within the ability ranges for which they are naturally suited, as discussed in Section 5.1. Looking at Fig 6 and 4 we see that most policies achieve a maximum mean duration of 8, which could indicate that the trajectories are too short and, that by artificially increasing the size of the sessions, the Agent would have a better estimate of h from empirical experience. With that in mind we estimate the unconditional mean and duration for the optimal and simple policy for various values of the *min_length* parameter defined in Sec 4. Results are shown in Table 2. Such results suggests that the state representation defined in Section 3.2 may be insufficient to achieve optimal performance across all ability levels.

5.3. Adding ability information

In order to investigate whether it is the lack of information on the student's ability that is hindering the performance of the agent, we add a discretization of the ability parameter in the following way: we segment the [0,1] interval into k subintervals of equal length and augment the state with a new variable $\hat{h}$, which encodes in which of the k subintervals the student's ability is contained.

As we can see in Figures 7 and 8, when choosing $k = 4$, the added information

Tabela 2. Impact of *min_length* in Unconditioned Return and Duration

Min Length	Return (π^*)	Duration (π^*)	Return (π_{simple})	Duration (π_{simple})
0	8.54	7.34	7.93	7.24
5	999	999	999	999
10	19.26	16.67	17.61	16.61
15	22.39	19.44	17.61	16.61

about the student's ability allows the optimal policy to not only achieve better results, but rival the trivial policies on their respective ranges. This can also be seen in Table 1.

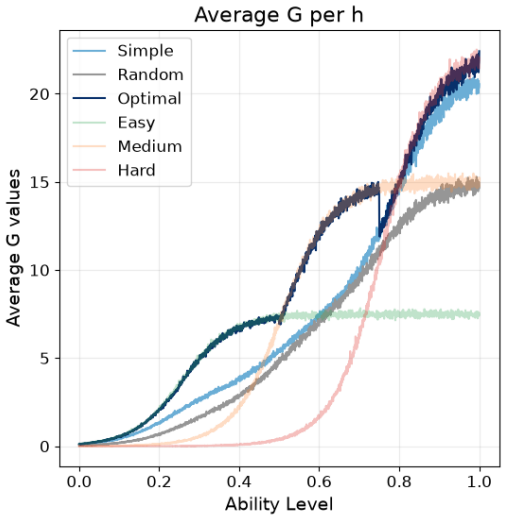


Figura 7. Average Return with Ability Information

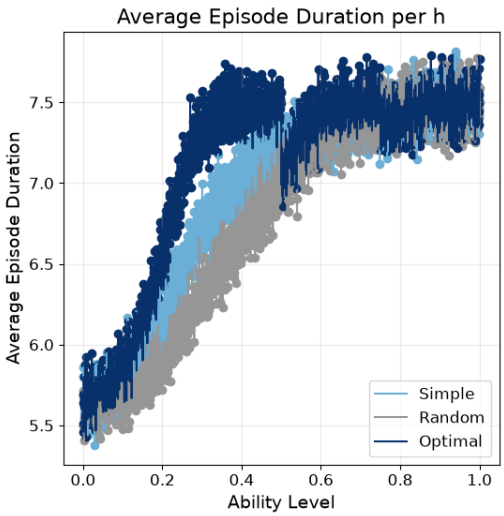


Figura 8. Average Duration with Ability Information

Note that there is a pronounced discontinuity in the average return around the ability threshold $(h = 0.75)$, caused by the discretization of ability levels. Students just below this threshold are assigned to $(\hat{h} = 3)$, a category dominated by medium-ability students, leading the policy to favor medium-difficulty questions. Once $(h > 0.75)$, students are assigned to $(\hat{h} = 4)$, where the policy shifts toward hard questions. However, students with abilities just above (0.75) are still closer to medium-ability learners, making these harder questions suboptimal and causing a temporary drop in performance.

This phenomenon is not similarly observed for the other ability thresholds because $\hat{h} = 1$ and $\hat{h} = 2$ correspond both to lower ability students that thrive with easy questions, and because the threshold between lower and medium ability coincides with the used ability threshold of 0.5.

These results confirm that the state representation defined in Section 3.2 contain insufficient information for the Agent to estimate an optimal policy, with information about the ability of the student being needed in order to achieve uniformly better performance.

6. Conclusion

In this paper, we addressed the instructional sequencing problem in ITS by proposing a tutoring environment and a probabilistic model for the student's interaction with the environment. A student has an ability h that determines the probability of correctly answering questions of varying difficulty levels, with session duration depending on both incorrect answers and time spent. We model the student-system interaction as an MDP and estimate a policy using Q-Learning by training on simulated student interactions.

Our experimental results show that the learned policy achieves competitive average performance compared to heuristic policies, but fails to outperform trivial policies

(e.g., always asking easy questions) for certain students. This limitation is attributed to the partial information contained in the system, and we show that adding information about the ability parameter can drastically improve performance.

These results demonstrate the effectiveness of RL for the instructional sequencing problem, where optimal tutoring policies consistently achieved higher overall returns in comparison to heuristic baselines, showing that data-driven instructional strategies can outperform manually designed rules.

Referências

Arroyo, I., Woolf, B. P., Burleson, W., Muldner, K., Rai, D., and Tai, M. (2014). A multimedia adaptive tutoring system for mathematics that addresses cognition, metacognition and affect. *International Journal of Artificial Intelligence in Education*, 24(4):387–426.

Atkinson, R. C. (1972). Optimizing the learning of a second-language vocabulary. *Journal of Experimental Psychology*, 96:124–129.

Bertsekas, D. P. and Tsitsiklis, J. N. (2008). *Introduction to Probability*. Athena Scientific, Belmont, MA, 2 edition.

Chi, M., VanLehn, K., and Litman, D. (2010). Do micro-level tutorial decisions matter: Applying reinforcement learning to induce pedagogical tutorial tactics. In *Intelligent Tutoring Systems*, pages 224–234.

Doroudi, S., Aleven, V., and Brunskill, E. (2019). Where's the reward?: A review of reinforcement learning for instructional sequencing. *International Journal of Artificial Intelligence in Education*, 29(4):568–620.

Guo, D., Yang, D., Zhang, H., et al. (2025). Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645:633–638.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285.

Nie, A., Reuel, A. K., and Brunskill, E. (2023). Understanding the impact of reinforcement learning personalization on subgroups of students in math tutoring. In *Artificial Intelligence in Education.*, volume 1831, pages 939–946. Springer.

Riedmann, A., S. P. . L. B. (2025). Reinforcement learning in education: A systematic literature review. *Int J Artif Intell Educ 35*, page 2669–2723.

Ruan, S., Nie, A., Steenbergen, W., He, J., Zhang, J. Q., Guo, M., Liu, Y., Nguyen, K. D., Wang, C. Y., Ying, R., Landay, J. A., and Brunskill, E. (2024). Reinforcement learning tutor better supported lower performers in a math task. *Machine Learning*, 113:3023–3048.

Sen, A., Patel, P., Rau, M. A., Mason, B., Nowak, R., Rogers, T. T., and 0001, X. Z. (2018). Machine beats human at sequencing visuals for perceptual-fluency practice. In *Proceedings of the 11th International Conference on Educational Data Mining*.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT Press, 2 edition.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3):279–292.