Improving Learning & Reducing Time: A Constrained Action-Based Reinforcement Learning Approach

Shitian Shen, Markel Sanz Ausin, Behrooz Mostafavi, Min Chi
Department of Computer Science
North Carolina State University
Raleigh, NC
<sshen,msanzau,bzmostaf,mchi>@ncsu.edu

ABSTRACT

Constrained action-based decision-making is one of the most challenging decision-making problems. It refers to a scenario where an agent takes an action in an environment not only to maximize the expected cumulative reward, but where it is subject to certain action-based constraints; for example, an upper limit on the total number of certain actions being carried out. In this work, we construct a general data-driven framework called Constrained Action-based Partially Observable Markov Decision Process (CAPOMDP) to induce effective pedagogical policies. Specifically, we induce two types of policies: $CAPOMDP_{LG}$ using learning gain as reward with the goal of improving students' learning performance, and $CAPOMDP_{Time}$ using time as reward for reducing students' time on task. The effectiveness of $CAPOMDP_{LG}$ is compared against a random yet reasonable policy and the effectiveness of $CAPOMDP_{Time}$ is compared against both a Deep Reinforcement Learning induced policy and a random policy. Empirical results show that there is an Aptitude Treatment Interaction effect: students are split into High vs. Low based on their incoming competence; while no significant difference is found among the High incoming competence groups, for the Low groups, students following $CAPOMDP_{Time}$ indeed spent significantly less time than those using the two baseline policies and students following $CAPOMDP_{LG}$ significantly outperform their peers on both learning gain and learning efficiency.

KEYWORDS

Constrained Reinforcement Learning, POMDP, Intelligent Tutoring System

ACM Reference format:
Shitian Shen, Markel Sanz Ausin, Behrooz Mostafavi, Min Chi. 2018. Improving Learning & Reducing Time: A Constrained Action-Based Reinforcement Learning Approach. In *Proceedings of 26th Conference on User Modeling, Adaptation and Personalization, Singapore, July 8–11, 2018 (UMAP'18),* 9 pages.
https://doi.org/10.1145/XXXXXX.XXXXXX

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
UMAP'18, July 8–11, 2018, Singapore
© 2018 Copyright held by the owner/author(s). Publication rights licensed to Association for Computing Machinery.
ACM ISBN 978-1-4503-5589-6/18/07…$15.00
https://doi.org/10.1145/XXXXXX.XXXXXX

1 INTRODUCTION

Intelligent Tutoring Systems (ITSs), as one type of highly interactive e-learning environment, have been widely used in the educational domain. ITSs generally provide step-by-step adaptive support and contextualized feedback to individual learners at run-time [12, 25]. Specifically, ITSs often apply the *pedagogical strategies* to decide which action to take (e.g. give a hint, show an example) in the face of alternatives at each time step with the purpose of improving student learning. Reinforcement Learning (RL) is one of the most promising data-driven approaches to induce an effective pedagogical strategy (policy) in ITSs. Most of prior research [11, 19, 22] apply the RL approaches in *unconstrained* contexts, where the agent selects actions given a situation to maximize expected cumulative reward. Specifically, the agent chooses actions at each step based upon the current state alone, regardless of prior decisions.

In this work, we mainly focus on inducing pedagogical strategy in a *constrained action-based* RL (CARL) scenario, which involves the additional action-based constraints such as a maximum number of times that an agent may take a specific action. For example, in American football, when the referee makes a call against a team the coaches can challenge it but they can only do so 3 times per game. And if the challenge is rejected, they not only lose an opportunity but they may face an additional penalty. Similarly in law, when prosecutors decide to charge someone with a crime, they are committed to prove it under the relevant law and they have limited options to add or modify the charges later on. In both scenarios, the early decisions impose special constraints on the future actions. In other words, the available actions for an agent at any given situation are governed not only by the current state but also by prior decisions. Therefore, when deciding the next action, the agent should take the constraints into account.

Prior research on *constrained* RL has focused on inducing the optimal policy subject to constraints such as *safety* and *risk avoidance*. Systems that physically interact with humans, for example, need to satisfy the basic safety parameters or engage in risk avoidance [1]. Similarly robots that seek to reach a target position as quickly as possible should also avoid dangerous places (say a crater) that might render them irretrievable [14]. Prior researchers [2, 8, 9, 21] who have sought to address such *constrained* scenarios have typically specified an additional *cost* function which has a similar format to the reward and then imposing constraints on the values of the cost functions. However, such constraints are different from the action-based constraints on which our work is focused. So far as we know, no prior work has directly sought to address the action-based constraints in an interactive e-learning environment.

In order to solve the CARL problem, we propose a general framework called Constrained Action-based Partially-observable Markov Decision Processes (CAPOMDP). In particular, we apply this framework to transform CARL problems into normal RL problems by leveraging factored state representations to incorporate constraints into the state space itself (see Sec. 4.1). The CARL scenario we investigated is an ITS called Deep Thought (DT) [17], which contains the action-based constraints and one type of tutorial decision: whether to provide students with a *Worked Example (WE)* or to ask them to engage in *Problem Solving (PS)*. When providing a WE, DT presents an expert solution to a problem step by step so that the student sees both the answer and the solution procedure. During PS, students are required to complete the problem with tutor's support. In DT, the action-based constraints are: the last problem on each level must be done in PS, and prior to reaching that problem the students must complete at least one PS and one WE.

In applying CAPOMDP to DT, we explore two types of reward functions: learning gain and time. For the former, the goal is to maximize learning gain, while the goal for the latter is to reduce the amount of time spent on completing the entire tutor. Prior research use either learning gain or time but *not both*. In this work, we apply CAPOMDP to induce two pedagogical policies using learning gain and time as reward respectively, and then evaluate them through two empirical experiments. In Experiment 1 (Exp1), we compare the CAPOMDP policies induced by learning gain against a policy where the system *randomly* decides whether to present the next problem as WE or as PS. Because both PS and WE are always considered to be *reasonable* educational interventions in our learning context, we refer to such policy as a *random yet reasonable* policy or *random* in the following. In Experiment 2 (Exp2), we compare the CAPOMDP policies induced by time against a Deep Q-Network (DQN) induced policy and a random policy.

Our empirical results suggest that there is an Aptitude Treatment Interaction (ATI) effect [7]. Specifically, we find that students with high incoming competence are less sensitive to the induced policies in that they achieve a similar learning performance regardless of policies employed whereas students with low incoming competence are more sensitive in that their learning is highly dependant on the effectiveness of the policies. Furthermore, Exp1 shows that CAPOMDP using learning gains as reward significantly outperforms the random policy in both learning gain and learning efficiency, and Exp2 shows that CAPOMDP using time as reward helps low incoming competence students spend significantly less time than the two baseline policies. The post-hoc comparison suggests that CAPOMDP using learning gains as reward also beats the DQN induced policy in terms of learning efficiency.

2 RELATED WORK

2.1 Applying RL into Educational Domain

MDP Framework. Both *learning gain* (LG) and *time* were explored as the reward functions in prior work in applying MDP to ITSs. For LG, Chi et al. [19] applied a model-based RL method with LG as reward to induce pedagogical policies for improving the effectiveness of an ITS that teaches students college physics. However, they found that the RL policy did not outperform the random policy. Similarly, Shen et al. [24] designed the immediate and delayed rewards based on learning gain and implemented a model-based MDP framework on a rule-based ITS for deductive logic. They found that the RL policies were more effective than random baseline for a particular type of learners. In addition, Rowe et al. [23] investigated a MDP framework with normalized LG (NLG) as reward for tutorial planning in a game-based learning system. They found that students in the induced planner condition had significant different behavior patterns from the controlled group but no significant difference was found between the two groups on the post-test.

When using *time* as reward function, Iglesias et al. [11] applied online Q-learning to generate a policy in an ITS that teaches students database design, with the purpose of providing students with direct navigation support through the system's content and minimizing the time spent in the teaching process. Similarly, Beck et al. [3] investigated temporal difference learning to induce pedagogical policies that would minimize the time that students take to complete each problem in an ITS that teaches arithmetic to grade school students. For both works, they found that the policy group spent significantly less time than the non-policy peers.

POMDP Framework. Mandel et al. [15] applied POMDP using *LG* as reward to induce policies in an educational game for teaching students concepts related to refraction. Their results showed that the induced POMDP policies outperformed both random and expert-designed polices in both simulated and empirical evaluations. Similarly, Clement et al. [6] constructed a student model to track students' individual mastery of knowledge components, and treated the mastery as hidden state and LG as reward in POMDP for inducing instructional policies. Their results showed that the POMDP policies outperformed the theory-based policies in terms of students' knowledge levels on task.

Different from the above POMDP applications, Rafferty et al. [22] applied the POMDP framework using *time* as reward in the domain of alphabet arithmetic. They interpreted the hidden states of POMDP as the students' latent knowledge related to concept learning. Their empirical study showed that the POMDP policies significantly outperformed the random policy in terms of time in that the former spent significantly less time than the latter.

Deep RL Framework. Wang et al.[26] applied a deep RL framework for personalizing interactive narratives in an educational game. They designed the rewards based on NLG and found that in simulation studies the students with the Deep RL policy achieved a higher NLG score than with the linear RL agent. Furthermore, Narasimhan et al. [20] implemented a DQN approach in a text-based strategy game, where the state is represented by a Long Short-Term Memory (LSTM) network, and the Q value is approximated by a multi-layered neural network, and the reward is designed based on the performance of game quest. Using simulations, they found that the deep RL policy significantly outperformed the random policy in terms of quest completion.

In summary, much of the prior work on applying RL in ITSs used either LG or time as reward but not both. Furthermore, compared with MDP and POMDP, relatively little research has been done on applying Deep RL to the field of ITS nor has it directly empirically compared Deep RL with other RL frameworks. Last but not least, none of the prior research has investigated the impact of different reward functions on RL frameworks by considering the action-based constraints in interactive learning environments.

2.2 Aptitude Treatment Interaction (ATI) Effect

Prior research in instructional strategies [7] assert that for any type of instructional interventions, an ATI effect is often exhibited, which claims that the instructional interventions are more or less effective for the learners depending on their abilities or aptitudes. Chi & VanLehn [5] investigated the ATI effect in the domain of probability and physics and their results showed that the high incoming competence students can learn regardless of instructional interventions, while for students with low incoming competence, those who follow the effective instructional interventions learned significantly more than those following less effective interventions. In our prior work, it is consistently shown that for pedagogical decisions on WE vs. PS, certain learners are always less sensitive in that their learning is not affected, while others are more sensitive to variations in different policies. For example, Shen et al. [24] trained students in an ITS for logic proofs, then divided students into the Fast and Slow groups based on time, and found that the Slow groups are more sensitive to the pedagogical strategies while the Fast groups are less sensitive.

3 BACKGROUND

3.1 Problem Statement

In the original RL scenarios, the agent selects the optimal action at any given situation to maximize the expected cumulative reward (ECR), which is represented as formula (1). In prior work in solving the classic constrained RL scenarios, the constrained MDP [2] or the constrained POMDP [14] frameworks generally search for the optimal policy that maximizes ECR while staying below the upper *bound* of the expected cumulative *cost (ECC)*, which involves defining a *cost function* for each pair of state and action.

By contrast, in the *constrained action-based RL* (CARL) scenarios described here, the agent chooses an action from a set of alternatives to both maximize its expected cumulative reward while obeying the action-based constraints. For example, the constraints in our application limit the total number of times that PS and WE can be selected in a given level. Therefore, rather than defining a cost function for each pair of state and action as the classic constrained RL scenarios, we formalize CARL problems as:

$$\max_{\pi} E_{\pi}\left[\sum_{t=0}^{L}\gamma^t R(s_t, a_t)\right] \tag{1}$$

$$\text{s.t. } 0 < C_{\pi}(a) = \left[\sum_{t=0}^{L}\mathbb{I}(a_t^{\pi} = a)\right] < \widehat{c_a}, \forall a \in A \tag{2}$$

where formula (2) is the action-based constraint. More specifically, A is the set of all possible actions and L is the length of a trajectory; $\widehat{c_a}$ and 0 denote the upper and lower bounds on the number of times that action a can be selected; a_t^{π} indicates that action a is selected by policy π at time step t, and $\mathbb{I}(\cdot)$ is the indicator function in that it would return 1 if the expression in $(\cdot)$ is true and 0 otherwise. $C_{\pi}(a)$ represents the cost function (2) for the action-based constraints and only depends on the actions. For our application, if the action-based constraint is active at time t, then state s_t is treated as a terminal state, where the agent cannot take any more actions.

3.2 POMDP Framework

POMDP is defined as a tuple: $\langle$ *S, O, A, R,* P_s*,* P_o*, Prior, B* $\rangle$. S represents the set of hidden states $\{S_1, S_2 ..., S_K\}$ with the set size K; O is the set of observations with a wide range of features; A and R denote the set of actions and the reward function respectively. P_s denotes the hidden state transition probability: $P_s(s', s, a) = Pr(s'|s, a)$, and P_o is the emission probability: $P_o(o, s, a) = Pr(o|s, a)$. *Prior* denotes the prior probability distribution of hidden states. In addition, B denotes the belief state space, where each element $b_t(s) = Pr(s|o_{1:t}, a_{1:t})$ is the probability distribution of the hidden state s at each time step t after executing the action sequence $a_{1:t}$ and obtaining the observation sequence $o_{1:t}$. Specifically, we can estimate $b_t(s)$ as:

$$b_t(s) = \frac{1}{Z}\sum_{s'} b_{t-1}(s')P_s(s', s, a_t)P_o(o_t, s, a_t) \tag{3}$$

Where Z is the normalization value. The belief state at the first step is calculated by multiplying *Prior* with the emission probability P_o. In our work, we utilize Input-Output Hidden Markov Models (IOHMM) [4] to translate the observations into belief states. In this context the input and output denote the action and observation. More specifically, the belief state at each time step is calculated by following formula (3) via the forward-backward IOHMM algorithm.

4 CONSTRAINED ACTION-BASED POMDP

As an extension of the POMDP framework, CAPOMDP modifies the state representation and the reward function to incorporate action-based constraints, shown in the following sections.

4.1 Factored State Representation

The factored state representation is constructed by concatenating the belief state with a constrained state. Specifically, the belief state is defined in the POMDP framework. The constrained state at time step t is defined as $[c_1^t, c_2^t, ..., c_{|A|}^t]$, where each element is a constrained feature which counts the total number of times that the action was chosen up to the present time point. If an action a is selected at a particular time step, the value of the corresponding constrained feature c_a is incremented by 1. Thus, we can estimate c_a^t efficiently as:

$$c_a^t = \begin{cases} c_a^{t-1} + 1 & a^t = a \\ c_a^{t-1} & \text{else} \end{cases} \tag{4}$$

Consequently, the factored state at time t is represented as $S^t = \left[b_1^t, b_2^t, ..., b_K^t, c_1^t, c_2^t, ..., c_{|A|}^t\right]$. In other words, the factored state contains two independent components: the belief state and the constrained state. The former is used to model the learning process, while the latter only tracks the status of the actions and whether the selection of the action triggers the constraints. Therefore, the factored state transition can be decomposed into separate estimates of the transition for the belief state component via function (3) and the transition for the constrained state component via function (4).

Furthermore, we designate a factored state as *safe* if all of the elements in its constrained state component satisfy the constraint function (2). *Error* states are defined as any state where one or more elements of its constrained state component violate function (2).

Error states are treated as one type of terminal state since they are disallowed in the system, while safe states permit actions to be taken which can transit to other states.

4.2 Reward Function

Since the basic ITS prohibits any appearance of an error state, we need to assign a strong negative reward for any transition from a safe state to an error state and treat error states as terminal states. We still retain the original reward for transitions between safe states in the training corpus since these transitions impose no additional cost. We therefore define the new constrained reward function as:

$$R_c(s_t, a_t) = \begin{cases} R(s_t, a_t) & s_t \in S_{safe}, s_{t+1} \in S_{safe} \\ -\widehat{c} & s_t \in S_{safe}, s_{t+1} \in S_{error} \end{cases} \quad (5)$$

Where $-\widehat{c}$ indicates a strong negative value. $R_c(s_t, a_t)$ represents the reward function with constraints, and $R(s_t, a_t)$ denotes the real reward in the training corpus. However, our training corpus does not contain error states because the original system has hard-coded rules to avoid them. Thus, we are required to manually add transitions from the safe states to error states with strong negative rewards in training dataset as shown in formula (5).

4.3 CAPOMDP Policy Induction

We implement Least Squares Policy Iteration (LSPI) [13] on the factor state of CAPOMDP to induce the optimal policy, which consists of two steps: *policy evaluation* and *policy improvement*.

In the *policy evaluation* step, we approximate the Q-function $Q(s, a)$, the expected reward of taking action a at state s, using a linear model generalized as:

$$Q(s, a) = \sum_{i=0}^{|S|*|A|} w_i \phi_i(s, a) \quad (6)$$

Where $\phi_i(s, a)$ indicates the basic element in state s associated with the action a, and s is a factored state representation in CAPOMDP framework (see Sec 4.1). $|S|$ and $|A|$ denote the size of the state set and action set respectively. w_i is the parameter of the linear model and it also involves a constant item (when $i = 0$). Additionally, we have that the Q-function follows the Bellman equation:

$$Q^{\pi} = R + \gamma P \Pi_{\pi} Q^{\pi} \quad (7)$$

By integrating equation (6) and (7), Least Square Temporal Difference Q learning approach [13] estimates the parameter w as:

$$\begin{cases} w = H^{-1} f \\ H = \sum_{(s,a,s') \in D} \phi(s, a)[\phi(s, a) - \gamma \phi(s', \pi(s'))]^T \\ f = \sum_{(s,a,s') \in D} \phi(s, a) R(s, a) \end{cases} \quad (8)$$

Where D is the training corpus, and $\pi(s')$ denotes the action selected by current policy π given a state s'. H and f can be estimated from the training corpus.

In the *policy improvement* step, w is updated through the gradient decent approach toward to minimize the loss function, then LSPI checks whether w converges. If w does not converge, it goes back to the policy estimation step; otherwise, it terminates.

Figure 1: Interface of PS in ITS

5 EXPERIMENT SETUP

5.1 Intelligent Tutoring System: Deep Thought

Deep Thought (DT) is a data-driven ITS used in the undergraduate-level Discrete Mathematics (DM) course at North Carolina State University (NCSU). DT [18] provides students with a graph-based representation of logic proofs which allows students to solve problems by adding rule applications (represented as nodes). The system automatically verifies proofs and provides immediate feedback on logical errors. Every problem in DT can be presented in the form of either Worked Example (WE) or Problem Solving (PS). In WE, students are given a detailed example showing the expert solution for the problem or were shown the best step to take given their current solution state. In PS (shown in Figure 1), by contrast, students are tasked with solving the same problem using the ITS or completing an individual problem-solving step. By focusing on the pedagogical decisions of choosing WE vs. PS, which would allow us to strictly control the content to be *equivalent* for all students. In addition to the problems, DT has two hard-coded *action-based constraints* that are required by the class instructors: the last problem on each level must be done as PS, and prior to reaching that problem the students must complete at least one PS and one WE.

Procedure. The problems in DT are organized into seven strictly ordered levels and in each level students are required to complete 3–4 problems. In the **pre-test** (level 1), all participants receive the same set of PS problems and students performance in this level is used to measure their incoming competence. In the following five **training levels** 2–6, before the students proceed to a new problem, the system followed the corresponding RL-induced or random policies to decide whether to present it as PS or WE. The last question on each level is a PS without DT's help and thus functioned as a mini-test for evaluating students' knowledge on the concepts of that level. In the **post-test** (level 7), all participants also receive the same set of PS problems and their performance in this level is evaluated as the post-test score. In addition, we defined the **Normalized Learning Gain (NLG)** as:

$$NLG = \begin{cases} \frac{post - pre}{100 - pre} & post >= pre \\ \frac{post - pre}{pre} & post < pre \end{cases} \quad (9)$$

Therefore, we evaluate students performance based on 1) pre- and post-test scores, 2) NLG, 3) time and 4) **Learning Efficiency** ($LE \propto$

NLG/*Time*). In the following, it is important to note that due to class constraints the pre- and post-tests covered different concepts and were collected at different times: the pre-test occurred in a single session before the policies were employed, while the post-test scores were collected at the end of later levels. Therefore the two scores cannot be directly aligned.

5.2 Training Corpus

The training corpus for RL policy induction was collected from training 570 students in DT in the Fall 2014, 15, and 16 semesters of a Discrete Mathematics course. In these semesters, DT was programmed to make random decisions when selecting PS and WE. Note that, when collecting our original training data, DT had already implemented the action-based constraints requested by the class instructors. The average number of solved problems in the form of both PS and WE is 24.1 (SD=2.59). Furthermore, DT recorded the observation as a set of 133 state features, including 59 discrete and 74 continuous features, for representing the students' behaviors and learning environment. DT calculated level score based on the last problem in each of training levels 2–6. For simplicity reasons, the range of level scores is normalized to $[0, 100]$.

When inducing RL policies using learning gain as reward, we calculated the difference between the student's current and prior level scores. If students quit the tutor during the training, we assigned a strong negative reward of -300 on the last problem he/she attempted. When inducing RL policies using time as reward, we used the *negative* log of time as the reward for inducing the policy: that is, when training on DT, the less time a student spent on completing the entire training portion, the better. There is a significant correlation between the *negative* log of times and student post-test scores: $cor = 0.19$, $p = .006$.

5.3 Deep RL Policy

We apply the Deep Q-Network (DQN) algorithm, proposed by Mnih et al. [16], to construct a strong baseline policy. DQN uses a neural network to map state features to Q-values $Q(s, a)$ for each action. The neural network consists of three LSTM layers [10] with 1000 units each, followed by a fully connected layer with 2 output units, one for each action. We trained the network using the DQN algorithm on the training corpus until convergence of the loss function. When implementing the neural network in the ITS, the action selected for each student was the action with the highest Q-value, between $Q(s, PS)$ and $Q(s, WE)$.

5.4 Policy Execution

Due to the hard-coded constraints, whenever DT makes a tutorial decision, the baseline policy execution process will first check whether the selected action violates the hard-coded action-based constraints shown in Figure 2 (a). If the action is valid then it will be carried out; otherwise DT uses hard-coded rules to choose an alternative that satisfies the constraints. By contrast, the CAPOMDP policy execution is shown in Figure 2 (b). Since the action-based constraints are already incorporated into the policy, we therefore expect that the induced CAPOMDP policy will be fully carried out and that the hard-coded rules will not be violated.

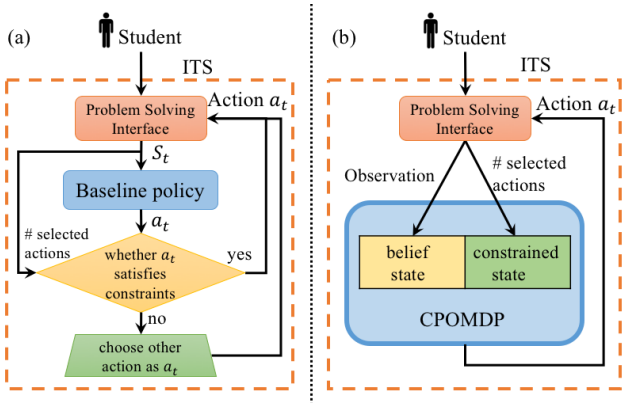


Figure 2: (a) Baseline policy execution with action-based constraints; (b) the CAPOMDP policy execution

5.5 Study Overview

In this work, we investigate two primary research questions: 1) whether using learning gain or time as reward makes the CAPOMDP framework induce a more effective pedagogical strategy; and 2) Can the CAPOMDP framework outperform the Deep Q-learning approach? We conducted two empirical experiments, involving the following four policies:

1. **CAPOMDP$_{\text{LG}}$:** CAPOMDP with learning gain as reward;
2. **CAPOMDP$_{\text{Time}}$:** CAPOMDP with time as reward;
3. **DQN$_{\text{LG}}$:** Deep Q-Network with learning gain as reward;
4. **Random:** Random yet reasonable decision (baseline).

Exp1 implements and compares the CAPOMDP$_{\text{LG}}$ and Random policies, and Exp2 applies the four policies including: CAPOMDP$_{\text{LG}}$, CAPOMDP$_{\text{Time}}$, DQN$_{\text{LG}}$ and Random. However, due to administration errors, very few students were randomly assigned to CAPOMDP$_{\text{LG}}$ and thus we will mainly focus on comparing the effectiveness of CAPOMDP$_{\text{Time}}$, DQN$_{\text{LG}}$ and Random in Exp2. Since all students are drawn from the same target population, we combine two CAPOMDP$_{\text{LG}}$ groups in Exp1 and Exp2 and also integrate the two Random groups in Exp1 and Exp2, and then conduct a post-hoc comparison across two experiments. In order to measure the Aptitude Treatment Interaction effect, we defined High and Low groups based on students incoming competence, the pre-test score. More specifically, we did a single median split of pre-test scores for all groups in both experiments since all students experienced an identical procedure. As shown in the following sections, this split reasonably reflects the incoming competence of the students.

6 EXPERIMENT 1 (EXP1)

6.1 Participants & Conditions

77 students enrolled in the DM course at NCSU in the Fall 2017 and were randomly assigned into two conditions: CAPOMDP$_{\text{LG}}$ ($N = 40$) and Random ($N = 37$). They were further divided into the High and Low groups using the median split on the pre-test scores as described above. Thus, we have: *CAPOMDP$_{LG}$-High* ($N = 25$),

Table 1: Students' learning performance in Experiment 1

Measure	*High*		*Low*	
	$CAPOMDP_{LG}$	*Random*	$CAPOMDP_{LG}$	*Random*
Pre-test	63.37(15.44)	66.02(15.01)	19.71(11.98)	14.87(11.15)
Post-test	54.36(20.34)	59.27(23.56)	57.13(16.77)	38.98(17.63)
NLG	-0.08(0.41)	-0.03(0.51)	**0.46(0.19)**	0.23(0.34)
Time	2.40(1.08)	3.15(1.49)	3.24(1.36)	3.83(1.37)
LE	0.02(0.64)	-0.01(0.67)	**0.49(0.31)**	0.21(0.28)

Note: Mean and SD of Time is in hours.

$CAPOMDP_{LG}$-*Low* ($N = 15$), *Random-High* ($N = 20$) and *Random-Low* ($N = 17$). While more students were assigned to $CAPOMDP_{LG}$-*High*, a χ^2 test shows no significant difference in the distribution of High vs.Low between conditions: $\chi^2(2, N = 77) = 1.5, p = 0.22$.

6.2 Results

Table 1 presents the mean and SD for students' corresponding learning performance and time in Exp1. One-way ANOVA tests show that no significant difference is found on the pre-test score either between the two conditions, or between $CAPOMDP_{LG}$-*High* and *Random-High*, or between $CAPOMDP_{LG}$-*Low* and *Random-Low*. As expected, the High groups ($M = 64.44$, $SD = 15.14$) score significantly higher than the Low groups ($M = 16.95$, $SD = 11.59$): $F(1, 75) = 231.2, p < .000$ on the pre-test score.

Post-Test: A two-way ANCOVA test using policy {$\text{CAPOMDP}_{\text{LG}}$, Random} and incoming competence {High, Low} as factors and the pre-test scores as covariate shows that there is a significant interaction effect on their post-test scores: $F(1, 72) = 5.78, p = .018$. A one-way ANCOVA shows that no significant difference is found either between $CAPOMDP_{LG}$ and Random, or between the High and Low groups. Additionally, one-way ANCOVA tests on policy using pre-test as covariate show that while there is no significant difference between the two High groups, a significant difference was found between the two Low groups: $CAPOMDP_{LG}$-*Low* achieves a significantly higher post-test score than *Random-Low*: $F(1, 32) = 8.65, p = .006$ (see Table 1).

NLG: Similarly, a two-way ANOVA test using policy and incoming competence as factors and the pre-test score as covariate yields a significant interaction effect on NLG: $F(1, 72) = 4.27, p = .018$. One-way ANOVAs shows that while there is no significant difference between $CAPOMDP_{LG}$ and Random, there is a significant between High and Low: $F(1, 75) = 4.16, p = .045$. The Low groups ($M = 0.33$, $SD = 0.30$) achieve a significantly higher NLG than the High groups ($M = -0.06$, $SD = 0.44$). Furthermore, while there is no significant difference between $CAPOMDP_{LG}$-*High* and *Random-High*, a one-way ANCOVA on policy using pre-test as covariate shows that $CAPOMDP_{LG}$-*Low* scores significantly higher NLG than *Random-Low*: $F(1, 33) = 5.46, p = .025$ (see Table 1).

Time: A two-way ANOVA test using condition and incoming competence as factors shows no significant interaction effect on total time that students spend in the tutor. One-way ANOVA tests show that there is no significant difference between $CAPOMDP_{LG}$ and Random, but the Low groups ($M = 3.57$, $SD = 1.28$) spend significantly more time than the High groups ($M = 2.70$, $SD = 1.30$): $F(1, 75) = 8.84, p = .004$. No significant difference is found either between the two High groups or between the two Low groups.

Learning Efficiency (LE): A two-way ANOVA test using policy and incoming competence as factors shows no significant interaction effect on LE, and a one-way ANOVA test also shows no significant difference between $CAPOMDP_{LG}$ and Random, but a significant difference is found between High and Low groups in that the Low groups ($M = 0.33$, $SD = 0.32$) score significantly higher than the High groups ($M = 0.005$, $SD = 0.64$): $F(1, 75) = 7.20, p = .009$. Despite this, while no significant difference is found between $CAPOMDP_{LG}$-*High* and *Random-High*, a one-way ANOVA test shows that $CAPOMDP_{LG}$-*Low* scores significantly higher LE than *Random-Low*: $F(1, 33) = 7.77, p = .009$.

6.3 Discussion

To summarize, we find significant difference between the High and the Low groups: the latter has significantly higher NLG, spends significantly longer time on DT, and achieves significantly higher LE than the High groups. More importantly, Exp1 exhibits an ATI effect: while no significant difference is found between the two High groups, significant differences are found between the two Low groups on post-test score, NLG and LE. In short, for the High incoming competence students it seems that both their learning performance and time on task is not impacted by the induced pedagogical strategies; for the low incoming competence students, however, by using LG as reward our CAPOMDP framework significantly benefits them more than the baseline Random policy on post-test, NLG and LE.

7 EXPERIMENT 2 (EXP2)

7.1 Participants & Conditions

139 students enrolled in the DM course at NCSU in Spring 2018 were randomly assigned into four conditions: $CAPOMDP_{LG}$ ($N = 12$), $CAPOMDP_{Time}$ ($N = 52$), DQN_{LG} ($N = 34$) and *Random* ($N = 41$). Due to administration errors, only 12 students were assigned to $CAPOMDP_{LG}$ and thus it is excluded in the following analysis. Similar to Exp1, students in Exp2 were divided into the High and Low groups using median split on the pre-test scores. Combining condition and incoming competence, we have a total of six groups: $CAPOMDP_{Time}$-*High* ($N = 34$), $CAPOMDP_{Time}$-*Low* ($N = 18$), DQN_{LG}-*High* ($N = 21$), DQN_{LG}-*Low* ($N = 13$), *Random-High* ($N = 26$) and *Random-Low* ($N = 15$). While it seems that High vs. Low is imbalanced, a χ^2 test shows no significant difference in the distribution of High vs. Low among the three conditions: $\chi^2(2, N = 139) = 0.12, p = 0.94$.

7.2 Results

Table 2 presents the mean (and SD) of students' corresponding learning performance and time in Exp2. A one-way ANOVA test shows no significant difference among the three conditions on their pre-test. As expected, the High groups ($M = 62.26$, $SD = 16.59$) score significantly higher pre-test than the Low groups ($M = 21.55$, $SD = 9.08$): $F(1, 137) = 240.6, p < .000$. Finally, no significant difference is found either among the three High groups or among the three Low groups.

Table 2: Students' learning performance in Experiment 2

Measure	*High*			*Low*		
	$CAPOMDP_{Time}$	DQN_{LG}	*Random*	$CAPOMDP_{Time}$	DQN_{LG}	*Random*
Pre-test	63.33(15.36)	63.72(15.45)	63.88(15.35)	23.26(9.07)	19.09(9.57)	21.63(8.79)
Post-test	55.99(22.01)	57.47(23.81)	64.46(19.11)	52.51(19.01)	45.65(21.84)	50.22(19.02)
NLG	-0.06(0.44)	-0.02(0.44)	0.08(0.42)	**0.37(0.26)**	0.27(0.39)	0.33(0.32)
Time	3.12(1.66)	3.13(1.58)	2.34(0.92)	**3.00(1.08)**	4.15(1.20)	3.88(1.38)
LE	0.04(0.55)	0.16(0.64)	0.13(0.58)	**0.39(0.34)**	0.21(0.27)	0.28(0.23)

Note: Mean and SD of Time is in hours; LE denotes the learning efficiency.

Post-test & NLG: Using policy {$CAPOMDP_{Time}$, DQN_{LG}, *Random*} and incoming competence {High, Low} as factors and the pre-test score as covariate, a two-way ANCOVA test shows there is no significant interaction effect on either their post-test scores or NLG. Additionally, one-way ANCOVA tests show that there is no significant difference either among conditions, or among the High groups, or among the Low groups on both post-test and NLG. Finally, while no significant difference is found between High and Low on post-test, the Low groups ($M = 0.31$, $SD = 0.29$) achieve significantly higher NLG than the High groups ($M = .08$, $SD = 0.55$): $F(1, 137) = 20.52, p < .000$. Therefore, much to our surprise, while DQN_{LG} is induced using learning gain as reward, it did not outperform Random; additionally, while $CAPOMDP_{Time}$ is induced using time as reward, it seems that it does not hurt students' learning performance as we are originally concerned about.

Time: To investigate whether $CAPOMDP_{Time}$ would indeed reduce student training time as expected, a two-way ANOVA test using condition and incoming competence as factors yields significant interaction effect on time: $F(2, 121) = 4.15, p = .018$. While one-way ANOVA tests show there is no significant difference among conditions, there is a significant difference between High and Low groups in that the Low groups ($M = 3.61$, $SD = 1.29$) spent significantly more time than the High groups ($M = 2.87$, $SD = 1.47$): $F(1, 125) = 8.10, p = .005$. More importantly, one-way ANOVA tests show that while there is no significant difference among the three High groups, there is a significant difference among the Low groups: $F(2, 43) = 3.90, p = .027$. Specifically, pairwise t-tests show that $CAPOMDP_{Time}$-*Low* spent significantly less time than either *Random-Low* or DQN_{LG}-*Low*: $p = .045$ and $p = .013$ respectively, and no significant difference was found between the latter two.

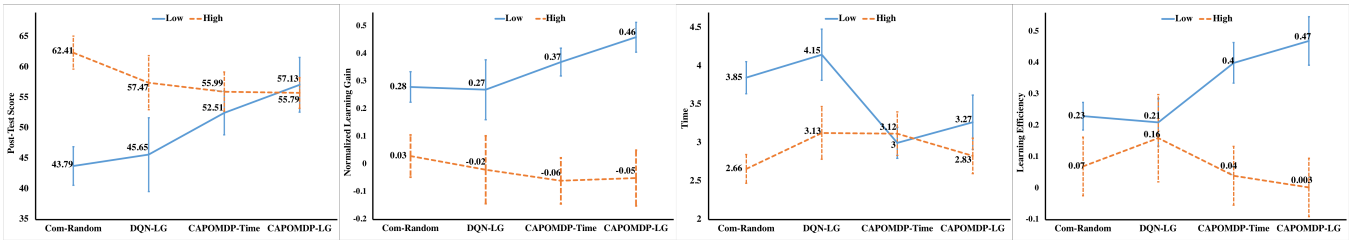
Learning Efficiency (LE): A two-way ANOVA test using condition and incoming competence as factors shows no significant interaction effect on LE. One-way ANOVAs indicate that there is no significant difference among the three conditions, but there is significant difference between High and Low in that the Low groups ($M = 3.61$, $SD = 1.29$) achieve significant higher LE than the High groups ($M = 0.10$, $SD = 0.57$): $F(1, 125) = 5.11, p = .025$. Although one-way ANOVA tests show that no significant difference is found among the High groups or among the Low groups, pairwise t-tests indicate that $CAPOMDP_{Time}$-*Low* scores marginally significantly higher LE than DQN_{LG}-*Low* ($p = .085$) (see Table 2).

7.3 Discussion

To summarize, in Exp2 we mainly focus on evaluating the effectiveness of the CAPOMDP framework using time as reward against the DQN using LG as reward and random group. Similar to Exp1, students are split into High vs. Low based on their pretest scores and the same patterns are found between the High and the Low groups: the latter had a significantly higher NLG, spent significantly longer time on DT, and achieved a significantly higher LE than their High peers. More importantly, while Exp1 exhibits an ATI effect on learning performance (post-test score, NLG and LE), Exp2 exhibits an ATI effect on time: while no significant difference is found among the three High groups, significant difference is found among the three Low groups in that students following $CAPOMDP_{Time}$ spent significantly less time than either DQN_{LG} or Random. In short, Exp2 shows that for the high incoming competence students, it seems that both their learning performance and time on task is not impacted by the induced pedagogical strategies; for the low incoming competence students, CAPOMDP using time as reward seemingly did not hurt their learning performance (post-test and NLG) and they indeed spent significantly less time than their peers under DQN_{LG} and Random. Much to our surprise, DQN_{LG} performs closely to Random. One of the possible reasons is that action-based constraints restrict the effectiveness of the DQN_{LG} policy. The DQN_{LG} and *Random* policies are only carried out 50.8% ($SD = 14.3\%$) and 51.1% ($SD = 11.3\%$) of the time respectively, while both the $CAPOMDP_{LG}$ and $CAPOMDP_{Time}$ policies can be fully followed.

8 POST-HOC COMPARISONS

In both Exp1 and Exp2, students were drawn from the same target population and all of them were enrolled in the experiments with the same method but in different semesters. By assigning students to each condition randomly, it provides the most rigorous test of our hypotheses. In this section, we conduct a post-hoc comparison across the two experiments in the hope that this wider view will shed some light on our main results. Especially while $CAPOMDP_{LG}$ outperformed Random in Exp1, it is not sure whether the same results would hold for Exp2 because we only have a small number of students assigned to $CAPOMDP_{LG}$ due to administration errors. Therefore, we want to combine the two Random groups into a single Random group and the two $CAPOMDP_{LG}$ into a single $CAPOMDP_{LG}$, and then compare their performance.



(a) Post-Test Score **(b) Normalized Learning Gain** **(c) Time** **(d) Learning Efficiency**

Figure 3: Students' learning performance across Experiment 1 and 2

Additionally, one-way ANOVA tests show that while there is no significant difference between two Random groups in Exp1 and Exp2 on pre-test: $F(1,76) = 2.76, p = 0.1$, NLG: $F(1,76) = 0.43, p = 0.52$, and LE: $F(1,76) = 0.46, p = 0.49$, there is a significant difference on time: $F(1,76) = 4.08, p = .046$ in that Random in Exp1 ($M = 3.51$, $SD = 1.34$) spent significantly more time than Random in Exp2 ($M = 2.90$, $SD = 1.32$), and there is a significant difference on post-test: $F(1,76) = 5.10, p = .027$ in that Random in Exp2 ($M = 59.25$, $SD = 20.08$) scored a significantly higher post-test than Random in Exp1 ($M = 48.31$, $SD = 22.71$). In short, Random in Exp2 performed better in post-test and spent less time than Random in Exp1. Therefore, by combining the two Random groups, we get a stronger baseline condition than Random in Exp1 alone.

We combine the two Random groups into a single Random group referred as *Com-Random* ($N = 78$) and the two $CAPOMDP_{LG}$ groups in Exp1 and Exp2 into a single $CAPOMDP_{LG}$ group ($N = 52$). Therefore, we have a total of four groups as described in section 5.5. One-way ANOVA tests show no significant difference among the four conditions on the pre-test score.

All students were further divided into High and Low using the same median split described above. Combining policy and incoming competence, we have a total of eight groups: *Random-High* ($N = 43$), *Random-Low* ($N = 35$), $CAPOMDP_{LG}$*-High* ($N = 34$), $CAPOMDP_{LG}$*-Low* ($N = 18$), $CAPOMDP_{Time}$*-High* ($N = 34$), $CAPOMDP_{Time}$*-Low* ($N = 18$), DQN_{LG}*-High* ($N = 21$), DQN_{LG}*-Low* ($N = 13$). A χ^2 test shows no significant difference in the distribution of High vs. Low among four policy groups: $\chi^2(3, N = 216) = 0.12, p = 0.57$.

Pre-test: As expected, the High groups ($M = 63.85$,$SD = 15.11$) score significantly higher than the Low groups ($M = 19.68$,$SD = 10.26$): $F(1,214) = 554, p < .000$ on pre-test. Except that, one-way ANOVA tests show that there is no significant difference either among $CAPOMDP_{LG}$, $CAPOMDP_{Time}$, DQN_{LG} and *Com-Random*, or among the four High groups, or among the four Low groups.

Post-test: Figure 3a presents the post-test scores of High vs. Low across the four policies. A two-way ANCOVA test using policy and incoming competence as factors and the pre-test score as covariate, yields no significant interaction effect on their post-test scores. Additionally, a two-way ANOVA test using policy and incoming competence as factors shows a significant interaction effect on their post-test scores: $F(3,208) = 2.81, p = .041$. Although one-way ANOVA tests show there is no significant difference among the High groups, there is a marginal significant difference among the Low groups: $F(3,80) = 2.29, p = .084$. Pairwise t-tests show that $CAPOMDP_{LG}$*-Low* scores the significantly higher post-test than *Random-Low*: ($p = .018$).

NLG: A two-way ANOVA test, using condition and incoming competence as factors yields no significant interaction effect on NLG. Additionally, a one-way ANOVA shows no significant difference among the four High groups, but a significant difference among the Low groups:$F(3,80) = 3.57, p = .017$. Pairwise t-tests show that $CAPOMDP_{LG}$*-Low* scores a significantly higher NLG than *Com-Random-Low*: $p = .043$ and a marginally significant higher NLG than DQN_{LG}*-Low*: $p = .094$, shown in Figure 3b.

Time: A two-way ANOVA test using policy and incoming competence as factors shows a marginally significant interaction effect on time: $F(3,208) = 2.50, p = .060$. Moreover, one-way ANOVA tests show that there is no significant difference among the four High groups, but there is a significant difference among the Low groups: $F(3,80) = 3.19, p = .027$. Specifically, pairwise t-tests show that $CAPOMDP_{Time}$*-Low* spends significantly less time than both *Com-Random-Low* and DQN_{LG}*-Low*: $p = .020$ and $p = .012$ respectively; $CAPOMDP_{LG}$*-Low* spends marginally significantly less time than DQN_{LG}*-Low*: $p = .053$, shown in Figure 3c.

Learning Efficiency (LE): A two-way ANOVA test using policy and incoming competence as factors shows no significant interaction effect on LE. Additionally, while one-way ANOVA tests indicate that there is no significant difference among the High groups, a significant difference is found among the Low groups: $F(3,80) = 3.57, p = .018$. Specifically, pairwise t-tests indicate that $CAPOMDP_{LG}$*-Low* scores significantly higher than both *Com-Random-Low* and DQN_{LG}*-Low*: $p = .007$ and $p = .016$ respectively; $CAPOMDP_{Time}$*-Low* achieves a marginally significantly higher LE than *Com-Random-Low* and DQN_{LG}*-Low*: $p = .065$ and $p = .083$ respectively, shown in Figure 3d.

9 CONCLUSION

We propose the CAPOMDP framework to deal with the action-based constraints in Deep Thought and we explored our CAPOMDP framework using both learning gain (LG) and time as rewards. Empirical results show that for the low incoming competence students, the $CAPOMDP_{LG}$ policy significantly outperforms the baseline random policy in terms of post-test score, NLG, and learning efficiency; the $CAPOMDP_{Time}$ policy significantly outperforms both DQN_{LG} and Random policies in terms of time. It seems that CAPOMDP indeed fulfills its promise for the low incoming students' learning in that it can improve their learning when using LG as reward and

reduce their time on task when using time as reward. However, for the high incoming students, both their learning performance and time are not impacted by either policies using LG as reward or those using time as reward.

Much to our surprise, DQN_{LG} performs close to Random. One of the possible reasons is that action-based constraints restrict the effectiveness of the DQN_{LG} policy. The DQN_{LG} and *Random* policies are only carried out 50.8% (SD = 14.3%) and 51.1% (SD = 11.3%) of the time respectively, while both the $CAPOMDP_{LG}$ and $CAPOMDP_{Time}$ policies can be fully followed. In future work, we will integrate the Deep Q-Network framework with the action-based constraints in our ITS. Moreover, we will induce the policy which can significantly improve LG as well as reduce the time, considering the learning gain and time as the objective simultaneously.

REFERENCES

[1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. 2017. Constrained policy optimization. *arXiv preprint arXiv:1705.10528* (2017).

[2] Eitan Altman. 1999. *Constrained Markov decision processes*. Vol. 7. CRC Press.

[3] Joseph Beck, Beverly Park Woolf, and Carole R. Beal. 2000. ADVISOR: A Machine Learning Architecture for Intelligent Tutor Construction. In *AAAI/IAAI*. 552–557.

[4] Yoshua Bengio and Paolo Frasconi. 1995. An input output HMM architecture. In *Advances in neural information processing systems*. 427–434.

[5] Min Chi and Kurt VanLehn. 2010. Meta-cognitive strategy instruction in intelligent tutoring systems: how, when, and why. *Journal of Educational Technology & Society* 13, 1 (2010), 25.

[6] Benjamin Clement, Pierre-Yves Oudeyer, and Manuel Lopes. 2016. A Comparison of Automatic Teaching Strategies for Heterogeneous Student Populations. In *EDM 16-9th International Conference on Educational Data Mining*.

[7] L. J. Cronbach and R. E. Snow. 1977. *Aptitudes and instructional methods: A handbook for research on interactions*. New York: Irvington.

[8] Dmitri Dolgov and Edmund Durfee. 2005. Stationary deterministic policies for constrained MDPs with multiple rewards, costs, and discount factors. *Ann Arbor* 1001 (2005), 48109.

[9] Javier Garcia and Fernando Fernández. 2012. Safe exploration of state and action spaces in reinforcement learning. *Journal of Artificial Intelligence Research* 45 (2012), 515–564.

[10] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[11] Ana Iglesias, Paloma Martínez, Ricardo Aler, and Fernando Fernández. 2009. Reinforcement learning of pedagogical policies in adaptive and intelligent educational systems. *Knowledge-Based Systems* 22, 4 (2009), 266–270. https://doi.org/DOI: 10.1016/j.knosys.2009.01.007 Artificial Intelligence (AI) in Blended Learning - (AI) in Blended Learning.

[12] Kenneth R Koedinger, John R Anderson, William H Hadley, and Mary A Mark. 1997. Intelligent tutoring goes to school in the big city. (1997).

[13] Michail G Lagoudakis and Ronald Parr. 2003. Least-squares policy iteration. *Journal of machine learning research* 4, Dec (2003), 1107–1149.

[14] Jongmin Lee, Youngsoo Jang, Pascal Poupart, and Kee-Eung Kim. 2017. Constrained Bayesian Reinforcement Learning via Approximate Linear Programming. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*.

[15] Travis Mandel, Yun-En Liu, Sergey Levine, Emma Brunskill, and Zoran Popovic. 2014. Offline policy evaluation across representations with applications to educational games. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*. 1077–1084.

[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[17] Behrooz Mostafavi and Tiffany Barnes. 2017. Evolution of an intelligent deductive logic tutor using data-driven elements. *International Journal of Artificial Intelligence in Education* 27, 1 (2017), 5–36.

[18] Zhongxiu Liu Mostafavi Behrooz and Tiffany Barnes. 2015. Data-driven Proficiency Profiling. In *Proc. of the 8th International Conference on Educational Data Mining*.

[19] M. Chi, Kurt VanLehn, Diane J. Litman, and Pamela W. Jordan. 2011. Empirically evaluating the application of reinforcement learning to the induction of effective and adaptive pedagogical strategies. *User Model. User-Adapt. Interact.* 21, 1-2 (2011), 137–180.

[20] Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. *arXiv preprint arXiv:1506.08941* (2015).

[21] Pascal Poupart, Aarti Malhotra, Pei Pei, Kee-Eung Kim, Bongseok Goh, and Michael Bowling. 2015. Approximate Linear Programming for Constrained Partially Observable Markov Decision Processes.. In *AAAI*. 3342–3348.

[22] Anna N Rafferty, Emma Brunskill, Thomas L Griffiths, and Patrick Shafto. 2016. Faster teaching via pomdp planning. *Cognitive science* 40, 6 (2016), 1290–1332.

[23] Jonathan P Rowe and James C Lester. 2015. Improving student problem solving in narrative-centered learning environments: A modular reinforcement learning framework. In *International Conference on Artificial Intelligence in Education*. Springer, 419–428.

[24] Shitian Shen and Min Chi. 2016. Reinforcement Learning: the Sooner the Better, or the Later the Better?. In *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization*. ACM, 37–44.

[25] Kurt Vanlehn. 2006. The behavior of tutoring systems. *International journal of artificial intelligence in education* 16, 3 (2006), 227–265.

[26] Pengcheng Wang, Jonathan Rowe, Wookhee Min, Bradford Mott, and James Lester. 2017. Interactive narrative personalization with deep reinforcement learning. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*.