Reinforcement Learning for the Adaptive Scheduling of Educational Activities

Jonathan Bassen[1], **Bharathan Balaji**[2], **Michael Schaarschmidt**[3], **Candace Thille**[2], **Jay Painter**[2], **Dawn Zimmaro**[2], **Alex Games**[2], **Ethan Fast**[1], **John C Mitchell**[1]
Stanford University[1], Amazon[2], University of Cambridge[3]
jbassen@stanford.edu, bhabalaj@amazon.com

ABSTRACT
Adaptive instruction for online education can increase learning gains and decrease the work required of learners, instructors, and course designers. Reinforcement Learning (RL) is a promising tool for developing instructional policies, as RL models can learn complex relationships between course activities, learner actions, and educational outcomes. This paper demonstrates the first RL model to schedule educational activities in real time for a large online course through active learning. Our model learns to assign a sequence of course activities while maximizing learning gains and minimizing the number of items assigned. Using a controlled experiment with over 1,000 learners, we investigate how this scheduling policy affects learning gains, dropout rates, and qualitative learner feedback. We show that our model produces better learning gains using fewer educational activities than a linear assignment condition, and produces similar learning gains to a self-directed condition using fewer educational activities and with lower dropout rates.

Author Keywords
Online education; adaptive learning; reinforcement learning.

CCS Concepts
•**Applied computing** → Computer-assisted instruction;

INTRODUCTION
Designing and improving online courses is a daunting task. Today, instructors usually organize course materials through some combination of best practice and intuition. But how do we choose which materials are *really* best? And in what order should we present them? These questions almost never have clear answers. For example, consider an online course on functional programming. We could begin the course with either lambda calculus or the basic concept of functions. We could present the concepts with a written explanation or a short video. It is not obvious which combination is better. Even

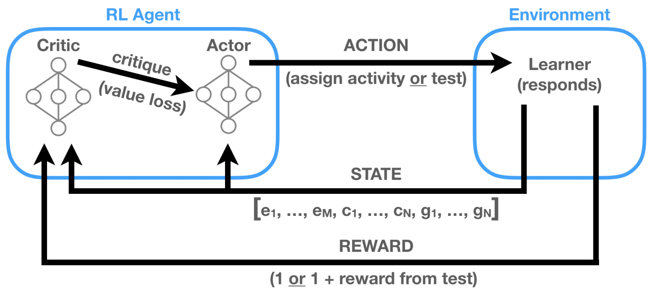
RL Agent
Environment
Critic
critique
(value loss)
Actor
ACTION
(assign activity or test)
Learner
(responds)
STATE
[e1, ..., eM, c1, ..., cN, g1, ..., gN]
REWARD
(1 or 1 + reward from test)

Figure 1. Reinforcement Scheduling uses an RL agent to assign activities in an online course. Over time, this method learns latent relationships between actions (assignments), states (learner traces) and rewards (outcomes) in order to schedule better and fewer activities to these learners. When a learner completes a pre-test or educational activity, their trace is updated in the form of a state vector with pre-test scores (e_m), assignment completions (c_n), and assignment scores (g_n).

more challenging, learners may perform better with different routes depending on their background and knowledge.

We refine these questions to focus on a specific goal for course design: *given a set of course materials, how can we assign each learner the smallest number of activities that maximize their learning gains?* Assigning fewer materials can help us identify which materials work best, both globally and for subpopulations of learners. By assigning personalized materials, we can provide those best tuned to a learner's trace and knowledge state. And by maximizing learning gains for each learner, we ensure that a course is effective. However, evaluating the efficacy of individual educational activities is challenging for a model, and using such a model to assign activities is even more so [17]. No prior work we know of has demonstrated that an adaptive assignment policy can be applied at scale in an online course, optimized in real time against learning gains.

To achieve this goal we introduce *reinforcement scheduling* (RS): a novel RL algorithm for assigning course materials based on a learner's starting knowledge state and history of interaction with course materials. Our model can learn to make assignment decisions in real time given feedback about how these decisions impact downstream post-test scores. Concretely, our model represents a learner's a priori knowledge through a pre-test score, and their interactions with the course that encodes completed problems and whether they were answered correctly. From this feature space, the model assigns the next activity to a learner. Using proximal policy optimization (PPO) [33], the model learns over time by optimizing a function that rewards for post-test based learning gains and penalizes for the number of materials assigned (Figure 1).

We evaluate RS by integrating it into an online linear algebra course taken by over 1000 learners. Existing open source platforms do not support the adaptive scheduling of educational activities, so we created a new online learning platform in order to deploy the course. We evaluated the impact of our model through a randomized controlled trial that divided learners between reinforcement scheduling and two control conditions. Through this experiment, we aimed to answer four research questions:

R1: How does reinforcement scheduling affect learning gains, the number of activities completed, and dropout?

R2: Do early participants suffer from a worse assignment policy under reinforcement scheduling?

R3: What can instructors and course designers learn from reinforcement scheduling?

R4: What are the qualitative experiences of learners under reinforcement scheduling?

In our investigation of these questions we find that RS largely succeeds in meeting our design goals while inspiring positive learner feedback. In particular, we found that learners who completed the course under RS demonstrated larger learning gains than the linear control condition ($p < .05$) and were also assigned fewer activities ($p < .01$). Further, RS led to lower dropout rates than the self-directed control condition ($p < .05$), with no significant difference to the linear condition. Finally, learners generally perceived the RS condition as adaptive and expressed satisfaction with the course.

In summary, this paper presents the first RL model to schedule educational activities in real time for a large online course through active learning. We show that when considering learning gains, the number of assignments, and learner dropout as a whole, our model performed favorably against two baseline assignment strategies. More broadly, our work gestures towards a future where online courses continuously improve themselves with reduced time and attention from instructors.

RELATED WORK

Reinforcement scheduling learns how to assign sequences of educational activities to online learners without skill labels or existing course data. We begin this section with some background information on skill labels and their use for course development. We then describe how our model builds upon prior work for adaptively assigning course materials. Finally, we discuss other methods for automating instructional practices and improving learning outcomes in online courses.

Skill Labels and Course Design

Skill labels are required inputs to many learner models and automated assignment algorithms, and they often play a key role in course design [18]. In this context, skills equate to the smallest units of actionable human knowledge [29]. Many course designers use skill labels to tag and organize their course materials. This helps them make sure the course materials cover all the desired skills, and don't require skills outside the scope of the course.

Unfortunately, determining the skills in a course and tagging course materials with skill labels is largely a manual and instructor-centric process. Cognitive task analysis (CTA) is the traditional mechanism for identifying skills, where individuals with a broad range of expertise in the field are asked to talk through their thought processes while they solve relevant problems. Each unique step applied within these problem solving processes is considered a separate skill [29]. Once a skill-labeled course has been launched, Learning Curve Analysis (LCA) can be used to detect missing and mislabeled skills from learner trace data [26, 5]; this technique requires an instructor to visualize learners' responses to single-skill problems over time, investigate any instances where performance does not increase monotonically.

Several techniques have emerged to automate the collection of these skill labels. For example, Q-matrices can learn to differentiate between different skills without skill labels [35]. Unfortunately, Q-matrices still require human determination of skill granularity, and cannot be used to directly label educational activities. Even the best skill mappings will always fall short of capturing the full complexity of human learning, which operates largely by analogical reasoning [9].

As we discuss in the following section, most learner models and automated assignment algorithms require skill labels and so lead to a potentially time consuming and expensive setup process. In contrast, our work aims for course materials to be organized at a higher level, such that course designers do not need to manually curate detailed skill labels to automate scheduling policies. By working from a model that learns to make effective assignments of materials without information about skill labels, we can make these tools accessible to the majority of instructors who do not use these precise labels when designing their curricula.

Adaptive Scheduling of Educational Activities

There is a long history of models that adaptively prescribe educational materials to online learners. Many intelligent tutoring systems have used scheduling models that improved learning outcomes and reduced the work of instructors [36, 12, 28]. However, historical learner data and model-training expertise are typically required to deploy these models. New courses must rely on conservative models, under-trained models, or self-directed learner navigation as a temporary scheduling strategy. The scheduling models for long-running courses need to be periodically retrained, especially in cases where the learner population changes over time.

The most common scheduling models require skill maps and combine these maps with learner knowledge models to make assignment decisions. Bayesian Knowledge Tracing (BKT) trains a Hidden Markov Model to predict a learner's binary knowledge state for each skill [5]; when answering problems, learners have some fixed probability of guessing when they don't know the required skills or slipping when they do. Similarly, IRT-Integrated Knowledge Tracing (IIKT) reformulates the BKT model so that learner ability exists on a spectrum between 0 and 1 and adds a parameter for the difficulty of each item [15]. These models require careful curation of skill labels, since they are sensitive to incorrect labels [18]; even so, they tend to perform worse in scenarios with multiple skills per educational activity [27]. In contrast to these methods, reinforcement scheduling does not require skill maps or assume that all activities covering a skill are equally effective.

Other scheduling models can be trained without explicit skill labels. For example, Deep Knowledge Tracing (DKT) uses an LSTM trained on binary learner traces to predict future responses [27]; though DKT can be used to schedule assignments, it cannot be used to actively optimize for the efficacy of assignments. Similarly, other methods have explored a set of parameterized instructional policies but cannot easily be used to evaluate individual educational activities [21, 22]. Contextual Bandits have been used to decide which instructional action is best for a given situation, but are unable to reason about delayed effects such as a post-test score [34].

RL has seen surprisingly little use in online education. RL models trained on historical data have been used to evaluate competing instructional policies for online education, demonstrating how ineffective educational activities can be eliminated, entirely [24]. However, just as with knowledge tracing, such evaluation requires historical data to train on. Even with hundreds of thousands of learner traces, RL will have difficulty learning long sequences of actions [7]. Q-Learning has been used to decide what feedback is given to learners for individual problems in an online course, but this model was limited to two simple binary decisions [14]; as we discuss later, Q-learning appears too sample inefficient to use for an activity-centered scheduling policy.

One big challenge when it comes to scheduling educational activities is the absence of platforms that support model-driven adaptive navigation. Several platforms support a fixed form of instructional model [36, 12, 28], but we could not identify any platforms that allowed for a new model to be plugged in and direct the scheduling of educational activities. This motivated us to create a new platform that would support adaptive scheduling through an API. Our platform works not just for our implementation of reinforcement scheduling, but for any model that conforms to the platform's specifications.

Other Methods for Automating Instruction

A broad class of prior work has proposed solutions for other instructional processes. For example, methods have been introduced to automate experimentation over instructional policies [21, 41, 12, 23]. Other work has explored how mindset interventions and automated dropout interventions improve learner outcomes [8, 39]. As a very different form of automation, learnersourcing has been used to manage hard-to-scale instructor tasks such as grading [19], providing feedback [20], giving explanations [40] and hints [10], and even creating new educational materials [25].

REINFORCEMENT SCHEDULING

Reinforcement scheduling is a novel method for adaptive activity assignment that leverages reinforcement learning and requires no pre-existing course data or skill labels. In this section, we describe the details underlying our method and the many considerations we brought to its design. We further discuss the learning platform we built to support reinforcement scheduling, and the course we created for its deployment.

Model Development

Our primary objective for reinforcement scheduling was to maximize learning gains while reducing the time spent on educational activities. We also adopted three design principles that encourage generalizability: 1) the model should support courses with different educational activities and topics, 2) the features should be meaningful, consistent, and non-redundant, and 3) there should be no requirement for human labels. To achieve these goals, we adopted a RL-based model: a good choice to make decisions and learn from their outcomes continuously and in real time.

As is standard practice when developing RL models, we are principally concerned with the choice of optimization strategy, action space, state space and reward function. The action space provides the set of possible decisions that can be taken by the RL agent at a given point in time. The state space defines the "state of the world" that is visible to an RL agent; in our case, this is a compact representation of a learner's trace. The reward function assigns value to the steps and outcomes that the RL agent arrives at. Finally, the optimization strategy determines the gradient chasing method that is used to improve the RL agent's decisions over time. What follows is a detailed explanation of our scheduling model, and the design process that led to its formation.

Action space

Under our navigational strategy, the only action that the RL agent takes is to prescribe the next educational activity or the post-test. Because the post-test is always the final thing to be scheduled in the course, its assignment leads to a terminal state. The action space is defined by a single binary vector with a position for each of the N activities and the post-test (at position $N+1$).

$$A_i = [a_1, ..., a_N, a_{N+1}]$$

We put two limitations on the actions taken by the RL agent. First, we prevented repeat assignments of the same educational activity to the same learner; repeats seemed unnecessary in a course with only 12 educational activities, and we wanted to ensure that the course length was no longer than 12 activities. Second, we prevented the RL agent from assigning the post-test to a learner until it had already given them at least one educational activity; this guarded against exploration of a useless trajectory, and the learner frustration that was likely to accompany it. To suppress a set of actions, we sampled from a regularized distribution of the remaining allowable action probabilities.

State space

Since test scores and educational activity scores help the RL agent make inferences about learning gains, this data formed the core of our state space. Pre-test problems are always completed in the same order before the learner moves on to any educational activities, so the pre-test scores can be concisely represented by a binary vector of length M. Educational activities can be delivered in any order, so the state space needs to communicate both the set of activities that a learner has completed and the associated scores.

We evaluated different state space designs using learner simulators. To encapsulate a learner's current record of scores and activity completions, we explored vectorized state spaces to feed to the RL agent. Inspired by DKT [27], we considered using concatenated one-hot encoding for each combination of educational activity and score within a learner trace; however, our simulations revealed that this large state space made RL training too sample inefficient for active learning. Instead, we found that a compact state space, which ignores the ordering of previous activities, was much faster and still able to make effective decisions. This learner state space (S) at step i is a concatenation of the M pre-test scores (E), a vector indicating which of the N educational activities have been completed so far (C), and a vector that holds the grade for each of the N activities once they have been attempted (G).

$$S_i = [e_1, ..., e_M, c_1, ..., c_N, g_1, ..., g_N]$$

Several additional features were evaluated, but ultimately eliminated for lack of utility. The post-test scores were left out of the state space because the optimization strategy we chose does not differentiate between terminal states; in courses with a fixed post-test, the post-test scores are the only part of the environmental state that change between the penultimate step and the end of an episode. The time spent by a learner on each activity was also left out, since the time spent on a page has proved to be a poor proxy for time spent on task [16]. Because all the educational activities were designed to take the same amount of time, we decided that the count of these activities would be a sufficient approximation.

Reward function

In line with our high-level design goals, we constructed a step-wise reward function that prioritized learner test performance improvements and penalized the inclusion of extra activities. The total reward (R) for a learner "episode" (a complete trajectory through the course) comes from the sum of these individual steps. It sums up individual improvements between the pre-test scores (E) and the post-test scores (O), and imposes a penalty (ψ) in proportion to the number of educational activities in the episode (H).

$$\begin{aligned} R_i &= 1 \text{ [after educational activity is assigned]} \\ &= 1 + \sum_{p=1}^{M} \Big[max(0, o_p - e_p) \Big] - (1+\psi) * H \\ &\qquad \text{[after post-test is assigned]} \end{aligned}$$

We chose R based on the speed and consistency with which our RL agent learned a good policy on simulated learners. We found that adding a small immediate reward for each educational activity assignment—and delaying the penalty for all educational activities—encouraged exploration of longer and more diverse course paths. This is counterbalanced by the penalty ψ. Similarly, we found that adding 1 to the post-test assignment reward prevented the RL agent from assigning activities too preferentially. The pair-wise increases between the pre-test and post-test scores allowed the agent to ignore cases where a learner guessed how to solve a problem correctly on the initial test or slipped up on a problem on the post-test; this improved both the speed and consistency of the RL agent's learning.

The educational activity assignment penalty (ψ) is a proxy for the minimum marginal test improvement that must be expected from an activity to justify its scheduling. ψ typically takes a value between 0 and 1. Course designers can think of this parameter as the proportion of learners that should see a one point increase from the pre-test to the post-test if you include this problem, to make it worth including.

The number and difficulty of test problems inform the magnitude of ψ. A course with half the test problems would halve the potential improvement going from the pre-test to the post-test; tests with easier problems should generally increase the value of ψ as a one-point increase becomes less meaningful. If this value is too low, learners are likely to waste more time on activities that don't support much learning. Conversely, if this value is too large, some useful activities will not be assigned to learners who would benefit from them. After experimenting with many parameter values for our short course with 6 test problems and 12 educational activities, we settled on a value of 0.2 for ψ; under most simulated conditions, this allowed for substantial reductions in the number of items assigned without sacrificing test improvement.

Policy Optimization

Actively-learned RL learns to make decisions in real time based on interactions with an environment, but this presents challenges for its use in online education. RL algorithms tend to have poor sample efficiency, often requiring hundreds of episodes to learn simple policies, or hundreds of thousands of episodes to learn more complicated ones [37]. Such models take too long to learn in courses with only hundreds or thousands of learners. We overcome this challenge for RS by using proximal policy optimization (PPO), a policy gradient method that leverages deep neural networks to more efficiently learn a scheduling policy [33]. Use of neural networks reduces the dimensionality of the state/action space and so also reduces number of samples an algorithm needs to converge.

PPO is an actor-critic algorithm, where the actor decides which action to take based on the environment state and the critic estimates how much reward we are expected to accrue from the given state. The actor and critic are represented by a neural network each, and use stochastic gradient descent (SGD) optimization [2] to converge to a good policy. An RL agent has the choice of selecting the best action given the past information or trying a different action that might yield higher rewards. This is referred to as the exploration-exploitation trade-off. In PPO, the actor network outputs the probability of taking each action and samples the recommended action from this distribution. As training proceeds, the actor will increase the probabilities for actions that give higher rewards.

The actor network maximizes the following loss function:

$$J_\theta = \mathbb{E}_t \left[min\left(\rho(\theta)A(s,a), clip(\rho(\theta), 1-\varepsilon, 1+\varepsilon)A(s,a)\right)\right]$$

$$\rho(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}$$

$$A(s_t,a_t) = R(s_t,a_t) + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$$

J_θ maximizes the probability of the actions that maximize expected cumulative discounted reward, where the expectation $\mathbb{E}_t$ is taken over the data collected. γ is used to discount future rewards to encourage the agent to accumulate rewards quickly, e.g., a game won in 5 steps is better than winning it in 10 steps.

π_θ is the probability of taking action a given state s and is given by the actor network with weights θ. The actor that chooses the action lags slightly behind the network updates, hence an importance sampling correction is applied with $\pi_{\theta_{old}}$. This correction reduces bias in the data by giving higher importance to actions which were taken with low probability and vice-versa. The importance sampling ratio ρ is clipped to be within $1 \pm \varepsilon$ to avoid large gradient updates that de-stabilize training. $A(s,a)$ computes the relative benefit of taking action a given state s. $A(s,a)$ is computed using $V(s,a)$, which is the predicted cumulative discounted rewards from the current state s. $V(s,a)$ is given by the critic network with weights ϕ.

The loss function of the critic network uses a recursive formulation, where it minimizes the mean squared error between the predicted value and the actual value given by the rewards $R(s,a)$ received when interacting with the environment:

$$L_\phi = \frac{1}{2} \sum_t V_\phi(s_t) - \left(R(s_t,a_t) + \gamma V_\phi(s_{t+1})\right)^2$$

We implemented our PPO agent using RLGraph, a framework for developing RL agents using modular computation graphs [32]. When we attempted Q-learning for activity scheduling, it took our agent tens of thousands of time steps to improve its decision making compared to hundreds with PPO [38].

Simulated Learners

Before unleashing any RL model on a course full of online learners, we needed to test its performance and reliability and validate our choices of hyperparameters. In situations where learner trace data is available, researchers have advocated for the validation of scheduling models with a "robust evaluation matrix" of learner simulators trained on this data [6]; the idea is to use a few learning models with different behaviors to make sure that your scheduling model isn't over-fitting for any assumed learner behaviors. Because we were launching a new course that would use active learning for assignments, we couldn't leverage historical learner data when testing out our scheduling model. Instead, we created two classes of simulated learners — based on BKT [5] and IIKT [15] — to imitate many possible scenarios for online learners engaging with our educational activities.

BKT assumes that learner knowledge state is binary for each of the skills in a course, that all problems involving the same skill elicit identical response patterns, and that learned skills are not forgotten. Learners who possess a skill have a static probability of slipping when they answer a problem that requires that skill. Learners who don't possess a skill have a different static probability of guessing correctly when they answer a problem that requires that skill. For problems that require many skills, BKT assumes that all the required skills must be known by a learner in order for them to solve a problem. Any time a learner encounters a problem with a given skill, there is also some probability that they will learn that skill and have their knowledge state flip from 0 to 1.

IRT-integrated knowledge tracing combines the IRT model for learner knowledge and performance with the BKT model for probabilistic learning gains [30]. The IRT model assumes that learner knowledge is skill-based, but that learner ability exists on a continuous spectrum. According to this model, learner performance increases following a logistic curve that tracks the ability to answer problems involving a skill. Unlike with BKT, IRT-integrated knowledge tracing can assign different difficulty parameters for problems involving the same skills. To calculate the probability of answering a problem correctly that involves multiple skills, it is also possible to take the average of the probabilities for all of these skills.

To develop and test our model, we attempted many different simulated scenarios, with randomly seeded knowledge states across the population of simulated learners. We first tested and optimized our RL model using the a BKT simulator. Once we were satisfied with its performance on a variety of possible parameters for this simulator, we began testing with the IRT-integrated knowledge tracing simulator. Unsurprisingly, our RL agent performed significantly worse when the answering pattern looked less distinct between learners with high knowledge of a skill and low knowledge of a skill; this made it harder to predict how a learner would perform on the post-test. As a final test, we adjusted both models so that the probability of learning varied not just by skill, but by individual item; we found that the RL model was able to successfully prioritize items that produced stronger learning gains. All of these learner simulators were developed using the OpenAI Gym's API for reinforcement learning environments [3].

Course Design

Integrating RL assignments into a course requires careful thought. To ensure that our course would work with RL, we recruited a large pool of learners for the model to learn from, designed appropriate educational activities, and employed a pre-test and post-test to measure learning gains.

Design Philosophy

Education theory supports competing strategies for sequencing course materials. Sometimes new material should be introduced using many connected examples [4], but other times learner knowledge should be built up from foundational skills [13]. Similarly, sometimes it is better to connect a few educational activities focused on the same skill before moving onto a new skill, but other times it is better to mix educational activities that cover separate skills [31]. With these different strategies in mind, we wanted to provide the RL agent with the flexibility to sequence activities however it saw fit. Each of our educational activities was self-contained and could be displayed on a single page without referencing other activities.

Course Overview

To maximize recruitment of online learners, we ran a course on elementary linear algebra. This topic is recommended as a prerequisite for many popular courses in software engineering, machine learning and artificial intelligence, and so was likely to have a willing population of learners. To lower the time cost of participation, we designed our course so that it could be completed in 90 minutes or less. The course was available to Amazon employees in English-speaking regions, where participants were recruited through an email campaign over the course of several months.

Given our goals for course length, we taught three basic skills from linear algebra. We created a pool of twelve auto-graded educational activities to match this curriculum — each of which required a multiple-select response, and was designed to take five minutes. Four types of educational activities were created for each skill: video explanations, written descriptions, worked examples and assessment questions. To measure learning gains from the beginning of the course to the end, we used a six-problem test covering the three course skills as an identical pre-test and post-test. In between, the RL agent would learn to assign different educational activities to learners, exploring the set of possible trajectories, and exploiting trajectories that had shown good results for similar learners in the past.

To provide the RL model with a consistent real-time feature set, each test problem and educational activity recorded a binary score of 1 or 0 after the learner responded to it. Activities that did not pose a problem for learners to solve, instead asked them: "Did you understand the content presented above?" Learners would select whether they "fully understood" or "did not fully understand", yielding scores of 1 or 0, respectively. Assessment questions provided feedback based on learners' responses. Conversely, since test problems were meant to be purely evaluative, learners did not see test scores or feedback until completing the post-test.

Pilot study

To check if our course would be appropriately difficult—covering things that learners didn't already know, but that they could learn in the span of a short course—we recruited twenty-four learners from our target population to take part in a pilot study. Rather than use an untrained RL agent to assign educational activities between the pre-test and post-test, we had all the participants go through the activities in whatever order they saw fit. This pilot also allowed us to identify points of confusion within our course materials and interfaces, which we were able to fix before the course launch.

Platform Development

To deliver a short course with RL assignments, we needed to measure learning gains from the beginning to the end of the course, flexibly assign learners to different course materials, and observe learners' responses to the assigned activities. We were unable to find an online learning platform that could connect with a scheduling algorithm to deliver real-time adaptive assignments to learners. Instead, we constructed a custom platform built for RL assignments that supported embedded videos, custom images, multiple-select responses and feedback messaging for our pool of educational activities.

Our learning platform exposed an API for learner traces and adaptive assignments, allowing it to interface with an arbitrary scheduling algorithm that conformed to its specifications. With each new learner response, their binary score would be added to their trace, and the scheduling algorithm would use this new information to instantaneously prescribe the next educational activity. Inspired by OARS [1], learner traces were composed of response "events" with: a course ID, activity ID, anonymous learner ID, response selection, score, and timestamp.

To enforce the constraints of the RL model, our platform prevented users from viewing more than one educational activity at a time or going back to previous educational activities (when not self-directing). Learners would complete the pre-test one problem at a time, get directed through a sequence of educational activities, and then get assigned to complete the post-test one item at a time. However, we created another version of the platform with an activity menu for self-directed learner navigation through the course activities. We used this alternative platform in one of our experimental control conditions.

EVALUATION

Can reinforcement scheduling assign learners fewer activities with increased learning gains? In this section, we evaluate the behavior and usefulness of our method by analyzing its change in performance over time and comparing it against two control conditions. We frame our analyses and results through the lens of our four research questions.

Experimental Design

To better understand the impact of our reinforced scheduling method on learners, we compared our method against two baselines in a randomized controlled trial. The first baseline condition, *linear assignment*, asked learners to complete all twelve of the educational activities in an order prescribed by an independent instructional designer. The second condition, *self-directed assignment*, asked users to choose their own path through the materials. In each condition, learners were first given a pre-test to evaluate their starting knowledge state and then a post-test to evaluate their knowledge state having completed the course. We used the same test for both the pre-test and post-test. Notably, learners did not receive feedback on any of their test solutions until the completion of the post-test. We measured learning gains as the difference in a learner's score between post-test and pre-test. We also collected learner traces throughout the course: how many and which activities a learner completed, the correctness of those activities, and whether and when a learner dropped out.

To run the trial, we opened recruitment for our "Introduction to Linear Algebra" course through an internal email campaign. We randomly assigned 95% of participants to the reinforcement scheduling condition, 2.5% to the linear assignment condition, and 2.5% to the self-navigation condition. We made

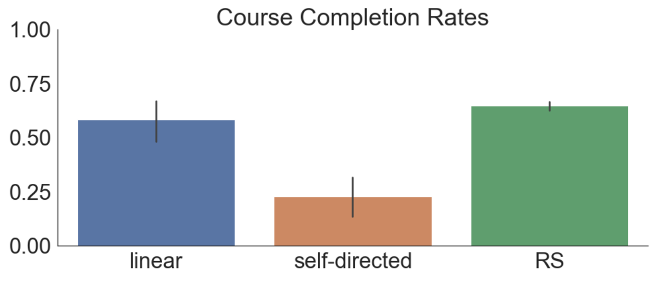

Figure 2. The RS and linear navigation conditions saw significantly higher completion rates than self-directed navigation (p < .05).

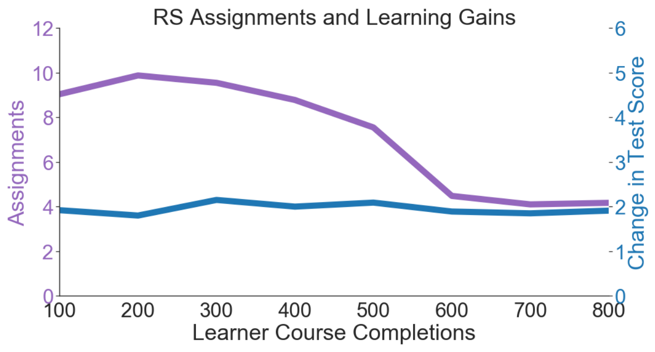

Figure 3. RS learned to assign fewer educational activities, on average over time, without decreasing the average learning gains.

the conditions as similar as possible to avoid introducing confounds. In each condition, learners received short instructions that explained the course setup they would experience: we explained the goals of the course, informed learners that they were part of a learning experiment, and explained how navigation would work for them. Finally, in order to receive a certificate of completion, all learners were required to complete a post-course survey.

In summary, 1987 people enrolled in the course. Of those randomly assigned to each condition: 1830 completed enrollment in the reinforced scheduling condition, 91 in the linear condition, and 66 in the self-navigation condition.

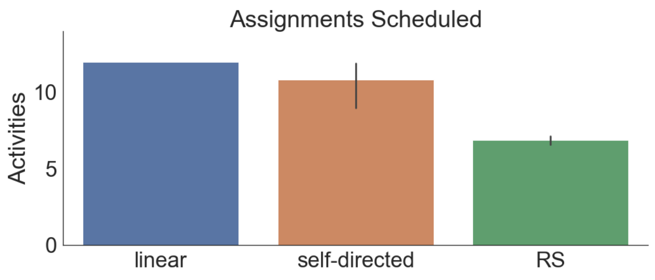

Figure 4. Learners in the RS condition saw significantly fewer educational activities than those in the two other conditions (p < .01). As presented in Figure 3, the number of educational activities assigned by RS decreased over time; the activities reported here are an average for all learners.

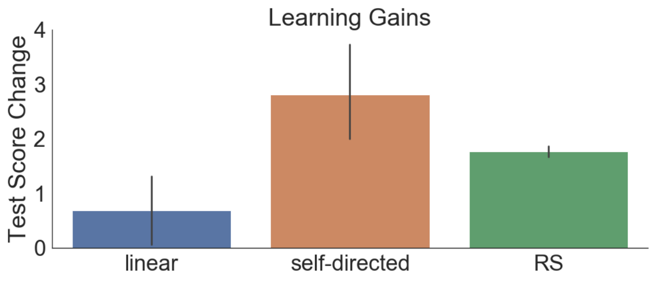

Figure 5. The learners in the RL and self-directed navigation conditions saw significantly better test score improvements than the learners following linear navigation (p < .05).

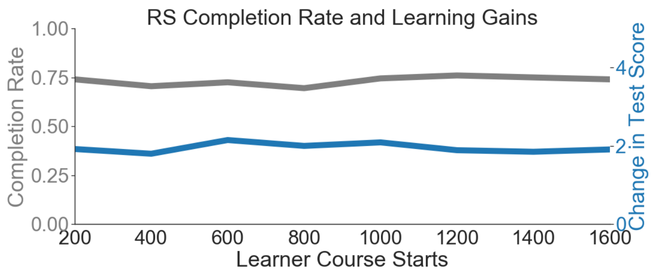

Figure 6. Both the learner completion rate and test improvements held steady under RS. Early participants in the course didn't fare any worse than the ones who joined later on.

R1: Comparing RS and Control Conditions

Our first analysis addresses how reinforcement scheduling affects learning gains, the number of activities learners complete, and course dropout (R1). We answered this question by comparing our method against two control conditions.

Method

We first analyzed the data produced by our experiment to collect several measures. We computed the number of activities completed by each learner from traces of learner interactions with the system. To compute dropout rates, we counted the number of learners who completed the course pre-test but not the post-test. We computed learning gains by subtracting each user's pre-test score from their post-test score.

We used t-tests to compare the number of activities completed and learning gains across conditions. To compare course dropout rates between conditions, we used chi-squared tests. For each analysis, we corrected for the fact that we performed multiple comparisons with the Bonferroni method.

Results

We depict comparison in dropout across conditions in Figure 2. The linear condition and RS condition yielded significantly higher course completion rates than the self-directed navigation condition ($p < .05$ in each case). There was no significant difference between the dropout rate for learners in the RL and the linear navigation conditions ($p > .05$).

Figure 4 shows a comparison of the number of activities completed on average by learners in each condition. Learners in the RS condition engaged with significantly fewer educational activities than those following linear and self-directed navigation ($p < .01$ in each case). As shown in Figure 3, the average number of educational activities assigned by the RL agent decreased from a maximum of approximately ten to an average of about four activities, delivering a greater reduction for later participants in the RL-managed course. The difference in activities between the RS condition and the self-directed navigation condition is even greater, considering that we did not double-count the return visits to past educational activities that were taken by the majority of self-directed learners.

Figure 5 shows the learning gains that we observed in each of our experimental conditions. Learners in the self-directed and RL conditions saw significantly higher test score increases than those in the linear navigation condition ($p < .05$ in each case). However, there was no significant difference between learning gains in the self-directed and RS conditions ($p > .05$).

Discussion

Collectively, these analyses show that learners in the RS condition completed fewer activities than those assigned to the linear and self-navigation conditions, demonstrated larger learning gains than learners in the linear navigation condition, and completed the course at higher rates than learners in the self-navigation condition.

We were surprised by the significantly higher dropout rate for self-directed learners, and it is possible that this condition's strong learning gains—equivalent to the RS condition—are at least partially due to a loss of less-motivated learners. Conversely, it is also surprising that learners in the linear condition demonstrated high completion rates, at the same level as the RS condition. When investigating further, we found that most dropout occurred shortly after the pre-test in these conditions, suggesting that the number of required educational activities didn't have a large effect on the dropout rate for our short course. Further, learner dropout was not strongly correlated with pre-test score in any of the conditions. This suggests that learners are more likely to complete a course where they don't have to decide what they will do next.

R2: The Cold Start Problem

Our second analysis addresses the concern—known as the "cold start" problem—that an RL agent might produce lower learning gains and course completion rates in early learners, before the model has been exposed to much training.

Method

To compute changes in measures as the course progressed, we partitioned learner data from the RS condition into groups of one hundred learners, beginning with the first hundred that entered the course, then the next hundred, and so on. (While larger partitions give a smoother curve, we repeated this method with smaller partition sizes and got similar results). We computed average completion rates and learning gains for each group in the partition, then plotted these results over time.

Results

Figure 6 shows that the learning gains and completion rates remain steady throughout the course. Though early participants in the RL condition have to complete more educational activities than later participants directed by RL, they still spent time on fewer total educational activities than participants following linear and self-directed navigation.

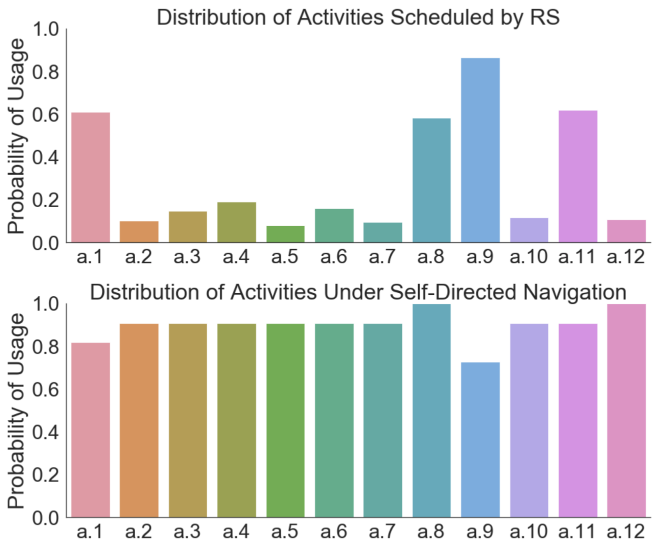

Figure 7. Under RS, learners were most likely to see educational activities 1, 9, 8 and 11. Self-directed learners engaged with every activity in the course at a similar frequency. RS activity distributions significantly deviate from a uniform distribution under a chi-squared test (p<.01).

Discussion

This result suggests that we can work past a common criticism of RL for human-in-the-loop applications; that RL agents may disproportionately harm early users before a fresh model has a chance to learn a decent policy [24]. Earlier versions of our model tested with simulated learners were less successful in this regard, and we eliminated this problem by making our reward function more conservative. First, we delayed the penalty for additional educational activities. Second, we provided a positive immediate reward for assignments, which biased our RL agent's early exploration towards longer paths.

R3: RS Scheduling Patterns

Our third analysis focuses on identifying useful patterns in the scheduling policy that RS learned for our course. We identify patterns in the assignments RS chose for different kinds of learners, and explore what instructors and course designers might infer from how our model learned to assign activities.

Method

We first collected data from the last 200 learners assigned activities by RS, as later participants were directed by the longest-trained policy and so are more representative of the trained model's decision making. Next, we looked at the assignment frequency distribution for educational activities to determine which were assigned most frequently, using a chi-squared test and odds ratios to determine which (if any) activities were assigned most frequently. We then investigated the relationship between pre-test scores and the number of resulting activity assignments with a linear regression model.

Results

Figure 7 shows significant differences in how often activities were assigned for the last 200 learners ($p < .01$). Learners in the RS condition were most likely to see activities 1 (1.99 odds), 8 (1.90 odds), 9 (2.82 odds) and 11 (2.019 odds). (These were "Defining A Vector: Video", "Vector Addition: Assessment Question", "Vector Norm L1: Video" and "Vector Norm L1: Worked Example".) Learners were least likely to see activity 5 ("Vector Addition: Video") with an odds ratio of 0.27. The four most commonly assigned educational activities included all activity types except for a written explanation. They also covered each of the three course skills, as designated by the course designer, but with a duplication of the third skill.

Figure 8 shows that as pre-test score increases, the number of educational activities assigned by RS decreases. Linear regression showed a 0.60 decline in the number of activities assigned for each one-point increase in pre-test score.

Discussion

Put together, these analyses show that RS adapted to different learner states. Learners in the RS condition completed fewer activities than those assigned to the linear and self-navigation conditions, demonstrated larger learning gains than learners in the linear navigation condition, and completed the course at higher rates than learners in the self-navigation condition.

Figure 7 illustrates that learners in the RL condition didn't just engage with fewer educational activities than self-directed learners — the RL learners were most likely to see activities 1, 8, 9 and 11. From this observation, an instructional designer might infer one of three things: 1) These four activities have the largest impact on test score improvements for learners in the course. 2) The scores from these activities are the most informative for the RL agent's future scheduling decisions. 3) Both explanations are simultaneously true. A logistic regression of the decisions made by the RL agent based on learner scores supports the first hypothesis. Learner scores did influence future assignments, but the influence of scores from these four activities wasn't stronger than the influence of the scores from the less-assigned educational activities.

If we accept that these four educational activities were best at increasing post-test scores, instructors and instructional designers might try to develop more activities that resembled these successful ones. Since no written explanation was included among the most useful activities, it is possible that this type of activity may be less useful; but more data would be required to make this judgement. Although this lack of certainty might be discouraging for a course designer trying to optimize the activities available, it lends support to the use of RL to evaluate individual activities — independent of their types.

Course developers might also consider whether the available educational activities were sufficient to produce strong learning outcomes. Comparing the linear condition to the RL condition, it's clear that the total number of assignments were not the only factor influencing test improvements. However, the *order* of the educational activities appear to have played a role on the size of those improvements. Following this experiment, it remains unclear whether *repetition* of the available activities or *self-directed navigation* through activities may have an effect on learning outcomes.

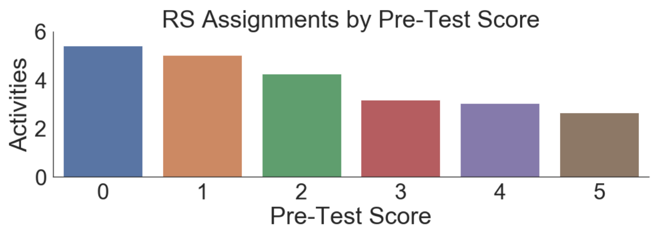

Figure 8. This bar chart shows the number of activities assigned to the last 200 RS learners, based on their pre-test scores. As learners performed better on the pre-test, RS assigned fewer activities.

R4: Qualitative Feedback From Learners

Method

We conducted an exit survey at the end of the course. The survey consisted of questions related to demographics, course quality, and textual feedback. We surveyed course quality across several measures using a Likert scale and analyzed learner textual feedback for broader themes.

Results

The mean score for *overall experience* was 5.53/7, with 32% of the users extremely satisfied and 49% moderately satisfied. Learners scored the effectiveness of the *number of activities* as 3.34/5, the effectiveness of *ordering of activities* as 3.38/5 and the effectiveness of the *selection of activities* as 3.46/5. 62% of the learners said the number of activities were "Just Right", while 24% said it was "Too Few" and 16% said it was "Too Many". If we only consider the last 200 learners, the "Too Many" activities category dropped to 7%.

Discussion

One clear theme that emerged from learner feedback is that learners generally liked the course and were intrigued by the possibility of using RL for adaptive assignment. For example: "I loved this" and "[I liked] the option to let me make mistakes, then teach me and they evaluate me again" and "[being assigned] lessons that seemed tailored to the specific subjects I was unsure about was quite interesting."

In general, many learners noticed and appreciated how the course adapted to their knowledge state. For example: "I was exposed to only those lessons which I needed" and "The lessons selected are very relevant to the questions that you answered wrongly." There were cases, however, in which the learners did not experience the course as adaptive. One learner said: "I LIKE the idea of [adaptivity]. But given that [I think that] I got all the pre-test questions right, it didn't seem to adapt at all." Such cases are to be expected in a RL algorithm which must occasionally explore new paths.

One implication of using an adaptive agent is that learners who know most of the material will not be assigned many activities. For example, one learner said: "Nice to not have to sit through everything (some of which I already know) and end up tuning out, missing the parts that I don't know/need review."

However, such a small number of activity assignments was not always perceived as helpful. Other learners said: "were there only two short videos?" and "I got bored because there was only one little thing I had to learn." This feedback became more prevalent as the course progressed.

We noticed that some learners expected to have one activity per skill, especially when they answered the relevant questions correctly on the pre-test. When the agent scheduled multiple activities per skill, some learners said: "some lessons/concepts appears twice" and "The order of lessons seemed a little haphazard, as some material that was presented earlier was repeated later." This sort of feedback declined over time and was absent in the last 200 learners.

Finally, we observed a high demand for similar courses. "Uh, please, PLEASE, do more of these. Sign me up." One learner told us. Another said: "Keep up with the good work, this can disrupt online education."

LIMITATIONS AND FUTURE WORK

In this section we discuss the limitations of our method and studies, many of which we aim to address in future work.

Generalizability

A key concern in designing our method was that it generalize to other courses and course materials. RS can be configured to schedule educational content for courses that use tests to evaluate learning gains and activities that can stand alone without cross-reference. And unlike most existing techniques, RS does not require carefully chosen skill labels to learn its policy. But, our method does not support courses where the educational activities involve complex or qualitative evaluation metrics, or where these activities are used to assess learners along the way. RS is not a good fit for courses that do not have many hundreds of learners, as the algorithm requires sufficient data to converge to a reasonable scheduling policy. The data requirements grow with the number of skills and activities in the course. Finally, we did not develop our method within the framework of mastery learning—where learners demonstrate consistent ability to apply a set of skill before they are able to move onto additional skills.

There are, however, solutions to some of these limitations. We can reduce the data required by mixing self-navigation and historical data. Use of improved algorithms like soft actor critic [11] will reduce the sample complexity further and generalize better to new learners and courses. Mastery learning can be emulated by including enough problems on the pre-test and post-test where the RL agent will be incentivized to increase their mastery, or by requiring learners to repeat the course until they have demonstrated sufficient mastery.

Scale of Data

While we integrated reinforcement scheduling into a large online course—the model interacted tens of thousands of times with nearly two thousand learners—bringing an order of magnitude more learners to the platform would present many new opportunities for analysis. In particular, it is not apparent that RS has converged on a final policy, and so it might continue to improve on its adaptive assignments. One possible outcome would be the elimination of entirely useless educational activities. With many thousands of learners, we would have the opportunity to optimize a policy over a wider range of skills and content, and we might observe more interesting ordering effects. If RS were run over a longer period of time for a course with changing learner behaviors, it would be interesting to see how well it adapts to these changes. Since there were not enough learners in the control conditions to compare activity assignments for specific pre-test responses and pre-test scores, across conditions, we would like to run an even larger randomized controlled experiment on RS with enough learners to compare the paths taken by these subpopulations.

Future Modifications of RS

There are three main adjustments to RS that we would consider for a re-run of our course. First, we would like to try eliminating the penalty for additional educational activity assignments from our reward function. The linear condition from our experiment shows that completing more activities does not always cause more learning, and RS might learn to reduce the number of assignments without an explicit penalty. If over-practice is actually a problem, its negative effects on test scores might be sufficient to prevent unnecessary activities from getting assigned. Next, it would be interesting to see how the RL model behaves differently if we cap the number of assignments, but allow the RL agent to make repeat assignments. This would require a larger state space for our model and make the RL agent less sample efficient, but learners might benefit from these repeats in ways we hadn't expected. Finally, we might add a term to our reward function that penalizes the loss of learners — to see if the RL agent can keep learners engaged for longer and prevent some instances of dropout.

CONCLUSION

This paper presents the first adaptive scheduling model to assign educational activities at scale in an online course through active learning. When balancing learning gains, the number of assigned items and learner dropout, our model performs favorably against two baseline assignment strategies: assigning all activities in an linear order defined by a course designer, and self-directed learner navigation. More broadly, our work gestures towards a future where online courses continuously improve themselves without the dedicated time and attention of instructors.

ACKNOWLEDGEMENTS

This work was supported by NSF grant 1176927-100-QCADX and Amazon. We thank our colleagues at Amazon's Machine Learning University for their generous support with recruitment, instruction, learning materials, and use of classroom space. We also thank our reviewers for their helpful feedback.

REFERENCES

[1] Jonathan Bassen, Iris Howley, Ethan Fast, John Mitchell, and Candace Thille. 2018. OARS: exploring instructor analytics for online learning. In *Proceedings of the Fifth Annual ACM Conference on Learning at Scale*. ACM, 55.

[2] Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*. Springer, 177–186.

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *arXiv preprint arXiv:1606.01540* (2016).

[4] Catherine C Chase, Jonathan T Shemwell, and Daniel L Schwartz. 2010. Explaining across contrasting cases for deep understanding in science: An example using interactive simulations. In *Proceedings of the 9th International Conference of the Learning Sciences-Volume 1*. International Society of the Learning Sciences, 153–160.

[5] Albert T. Corbett and John R. Anderson. 1994. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction* 4, 4 (01 Dec 1994), 253–278. DOI:http://dx.doi.org/10.1007/BF01099821

[6] Shayan Doroudi, Vincent Aleven, and Emma Brunskill. 2017. Robust Evaluation Matrix: Towards a More Principled Offline Exploration of Instructional Policies. In *Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale (L@S '17)*. ACM, New York, NY, USA, 3–12. DOI: http://dx.doi.org/10.1145/3051457.3051463

[7] Miroslav Dudík, John Langford, and Lihong Li. 2011. Doubly robust policy evaluation and learning. *arXiv preprint arXiv:1103.4601* (2011).

[8] Andrew J Elliot and Carol S Dweck. 2013. *Handbook of competence and motivation*. Guilford Publications.

[9] Dedre Gentner. 1983. Structure-Mapping: A Theoretical Framework for Analogy. *Cognitive science* 7, 2 (1983), 155–170.

[10] Elena L. Glassman, Aaron Lin, Carrie J. Cai, and Robert C. Miller. 2016. Learnersourcing Personalized Hints. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing (CSCW '16)*. ACM, New York, NY, USA, 1626–1636. DOI: http://dx.doi.org/10.1145/2818048.2820011

[11] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Jennifer Dy and Andreas Krause (Eds.), Vol. 80. PMLR, StockholmsmÃd'ssan, Stockholm Sweden, 1861–1870. http://proceedings.mlr.press/v80/haarnoja18b.html

[12] Neil T Heffernan and Cristina Lindquist Heffernan. 2014. The ASSISTments Ecosystem: Building a platform that brings scientists and teachers together for minimally invasive research on human learning and teaching. *International Journal of Artificial Intelligence in Education* 24, 4 (2014), 470–497.

[13] Neil T Heffernan and Kenneth R Koedinger. 1998. A developmental model for algebra symbolization: The results of a difficulty factors assessment. In *Proceedings of the twentieth annual conference of the cognitive science society*. Hillsdale, NJ, 484–489.

[14] 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.

[15] Mohammad Khajah, Yun Huang, José P. González-Brenes, Michael C. Mozer, and Peter Brusilovsky. 2014. Integrating Knowledge Tracing and Item Response Theory: A Tale of Two Frameworks. In *UMAP Workshops*.

[16] Juho Kim, Philip J. Guo, Daniel T. Seaton, Piotr Mitros, Krzysztof Z. Gajos, and Robert C. Miller. 2014. Understanding In-video Dropouts and Interaction Peaks Inonline Lecture Videos. In *Proceedings of the First ACM Conference on Learning @ Scale Conference (L@S '14)*. ACM, New York, NY, USA, 31–40. DOI: http://dx.doi.org/10.1145/2556325.2566237

[17] Kenneth Koedinger, Philip I Pavlik Jr, John Stamper, Tristan Nixon, and Steven Ritter. 2010. Avoiding problem selection thrashing with conjunctive knowledge tracing. In *Educational data mining 2011*.

[18] Kenneth R Koedinger, Albert T Corbett, and Charles Perfetti. 2012. The Knowledge-Learning-Instruction framework: Bridging the science-practice chasm to enhance robust student learning. *Cognitive science* 36, 5 (2012), 757–798.

[19] Chinmay Kulkarni, Koh Pang Wei, Huy Le, Daniel Chia, Kathryn Papadopoulos, Justin Cheng, Daphne Koller, and Scott R. Klemmer. 2013. Peer and Self Assessment in Massive Online Classes. *ACM Trans. Comput.-Hum. Interact.* 20, 6, Article 33 (Dec. 2013), 31 pages. DOI: http://dx.doi.org/10.1145/2505057

[20] Chinmay E Kulkarni, Michael S Bernstein, and Scott R Klemmer. 2015. PeerStudio: rapid peer feedback emphasizes revision and improves performance. In *Proceedings of the second (2015) ACM conference on learning@ scale*. ACM, 75–84.

[21] Robert V Lindsey, Michael C Mozer, William J Huggins, and Harold Pashler. 2013. Optimizing Instructional Policies. In *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 2778–2786. http://papers.nips.cc/paper/4887-optimizing-instructional-policies.pdf

[22] Yun-En Liu, Travis Mandel, Emma Brunskill, and Zoran Popović. 2014a. Towards Automatic Experimentation of Educational Knowledge. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 3349–3358. DOI: http://dx.doi.org/10.1145/2556288.2557392

[23] Yun-En Liu, Travis Mandel, Emma Brunskill, and Zoran Popovic. 2014b. Trading Off Scientific Knowledge and User Learning with Multi-Armed Bandits.. In *EDM*. 161–168.

[24] 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*. International Foundation for Autonomous Agents and Multiagent Systems, 1077–1084.

[25] Piotr Mitros. 2015. Learnersourcing of complex assessments. In *Proceedings of the Second (2015) ACM Conference on Learning@ Scale*. ACM, 317–320.

[26] Allen Newell and Paul S Rosenbloom. 1981. Mechanisms of skill acquisition and the law of practice. *Cognitive skills and their acquisition* 1, 1981 (1981), 1–55.

[27] Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J Guibas, and Jascha Sohl-Dickstein. 2015. Deep knowledge tracing. In *Advances in neural information processing systems*. 505–513.

[28] Martha C Polson and J Jeffrey Richardson. 2013. *Foundations of intelligent tutoring systems*. Psychology Press.

[29] J. vanMarrienboer K. Yates R. Clark, D. Feldon and S. Early. 2008. *Handbook of research on educational communications and technology* (3rd ed.). Chapter Cognitive task analysis for training, 577–593.

[30] Georg Rasch. 1960. Studies in mathematical psychology: I. Probabilistic models for some intelligence and attainment tests. (1960).

[31] Doug Rohrer. 2009. The effects of spacing and mixing practice problems. *Journal for Research in Mathematics Education* (2009), 4–17.

[32] Michael Schaarschmidt, Sven Mika, Kai Fricke, and Eiko Yoneki. 2019. RLgraph: Modular Computation Graphs for Deep Reinforcement Learning. In *Proceedings of the 2nd Conference on Systems and Machine Learning (SysML)*.

[33] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[34] Avi Segal, Yossi Ben David, Joseph Jay Williams, Kobi Gal, and Yaar Shalom. 2018. Combining Difficulty Ranking with Multi-Armed Bandits to Sequence Educational Content. In *Artificial Intelligence in Education*, Carolyn Penstein Rosé, Roberto Martínez-Maldonado, H. Ulrich Hoppe, Rose Luckin, Manolis Mavrikis, Kaska Porayska-Pomsta, Bruce McLaren, and Benedict du Boulay (Eds.). Springer International Publishing, Cham, 317–321.

[35] Kikumi K Tatsuoka. 1995. Architecture of knowledge structures and cognitive diagnosis: A statistical pattern recognition and classification approach. *Cognitively diagnostic assessment* (1995), 327–359.

[36] J. Sewall V. Aleven, B. McLaren and K. Koedinger. 2006. The cognitive tutor authoring tools (CTAT): preliminary evaluation of efficiency gains. In *Intelligent Tutoring Systems (ITS '06)*. 61–70.

[37] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. 2016. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224* (2016).

[38] Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning* 8, 3-4 (1992), 279–292.

[39] Jacob Whitehill, Joseph Williams, Glenn Lopez, Cody Coleman, and Justin Reich. 2015. Beyond prediction: First steps toward automatic intervention in MOOC student stopout. *Available at SSRN 2611750* (2015).

[40] Joseph Jay Williams, Juho Kim, Anna Rafferty, Samuel Maldonado, Krzysztof Z. Gajos, Walter S. Lasecki, and Neil Heffernan. 2016. AXIS: Generating Explanations at Scale with Learnersourcing and Machine Learning. In *Proceedings of the Third (2016) ACM Conference on Learning @ Scale (L@S '16)*. ACM, New York, NY, USA, 379–388. DOI: http://dx.doi.org/10.1145/2876034.2876042

[41] Joseph Jay Williams, Anna N. Rafferty, Dustin Tingley, Andrew Ang, Walter S. Lasecki, and Juho Kim. 2018. Enhancing Online Problems Through Instructor-Centered Tools for Randomized Experiments. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, Article 207, 12 pages. DOI: http://dx.doi.org/10.1145/3173574.3173781