Proceedings of CONF-MLA 2025 Symposium: Intelligent Systems and Automation: AI Models, IoT, and Robotic Algorithms
DOI: 10.54254/2755-2721/2025.LD30332

Optimizing Student Engagement: Reinforcement Learning Approaches to Problem Selection in ITS

Yan-Ming Huang

Shanghai Starriver Bilingual School, Shanghai, China
daniel.hym@foxmail.com

Abstract. Choosing the right next problem is key to good tutoring. This project uses reinforcement learning (RL) to improve healthy help-seeking in MathSpring, an online math tutor used in U.S. schools. The target behavior is productive hint use—asking for a hint and then finishing the problem—while keeping solve rate steady and time reasonable. A data-driven simulator is built from about 400,000 student–system logs. Given a student state and an action, the simulator predicts likely outcomes such as hint use, correctness, and time. The only action the agent can take is to recommend the difficulty of the next problem. Difficulty comes from MathSpring's existing scale, which combines average time, hints, and correctness, and is grouped into three buckets: easier, on-level, and harder. A RL method—tabular Q-learning—is trained inside the simulator. This project used a student-level train/validation/test split, log rewards and key learning metrics, and compare the learned policies with historical behavior using off-policy evaluation and simulator rollouts. The conclusion also includes a discussion of practical limits, including different classroom uses (review vs. preview), short sessions, imbalanced actions toward easy items, and cross-grade recommendations. This work offers a clear, reproducible path for using RL to support smarter help-seeking in real classrooms while keeping methods and choices easy to understand. Notably, without sacrificing solve rate, this RL-based approach improved hint usage rate from 10.2% to 25.1%.

Keywords: educational data mining, help-seeking behavior, intelligent tutoring systems, reinforcement learning

1. Introduction

Intelligent Tutoring Systems (ITS) use computers to give students one-on-one help at a large scale [1]. Early systems like the LISP Tutor showed that a computer tutor can guide students through tough steps, notice mistakes, and offer timely hints in ways that approach human guidance [2]. Since then, ITS research has developed new ideas such as checking whether an answer follows key rules (not only "right or wrong"), reacting to student emotions, and making learning feel more like a game [3-5]. Recent progress in artificial intelligence now allows tutors to generate more personal practice and feedback [6,7]. Such approaches align with broader work on AI-driven personalization in education across different countries [8]. Together, these advances create a clear opportunity: use data and learning algorithms to decide what problem to give next and when a hint will help most.

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2025.LD30332

This paper studies that opportunity in MathSpring, an online math tutor used in U.S. schools [3,9]. MathSpring grew out of an earlier system called Wayang Outpost and focuses on three goals: building knowledge, keeping students engaged, and supporting healthy learning behaviors [3,10]. A central idea in MathSpring is "effort-based learning": the system looks at how long a student works, how many attempts they make, and whether they ask for help in order to choose the next step [3,11]. This project builds on that idea with a reinforcement learning (RL) policy that recommends the difficulty of the next problem [12]. Aligning the next challenge to the student's current state could encourage productive hint use without hurting success rates or making time-on-task skyrocket.

This project focuses on hint usage for two reasons. First, asking for help at the right time is a key study skill. Second, the dataset shows that while the overall solve rate is about 53.6%, the hint usage rate is only about 10%. This suggests that many students may try to push through alone even when a small hint could unlock the solution. "Using a hint and then solving" is treated as a healthy learning behavior to be promoted. At the same time, this project guards against a simple but harmful strategy —always giving hard problems just to force more hints—by also tracking solve rate and time.

The project uses a large log of MathSpring interactions (about 400k rows) to build a data-driven simulator of student behavior. Each step in the simulator takes a student state and an action and returns likely outcomes such as whether the student uses a hint, whether they solve the problem, and how long it takes. The only action the agent can take is to recommend the next problem's difficulty. Difficulty is defined through MathSpring's existing scale, which blends average time to solve, hints used, and correctness, and group it into three buckets: easier (A1), on-level (A2), and harder (A3). This keeps the action space small and realistic for classrooms where many problems share the same skill.

A RL method—tabular Q-learning—is trained inside the simulator [12]. To compare design choices, this project builds two versions of the state the algorithm sees. One version uses compact "readiness" signals such as recent solve rate and hint rate; the other emphasizes process and engagement signals such as time to first hint, mistakes, and simple effort markers. Training is done with a student-level train/validation/test split to avoid leaking a student's behavior across sets. The reward balances the main goals: points for using a hint and for solving, a small penalty for long time, and a bonus for "hint + solve." This trains the agent to seek the sweet spot where help is used wisely and learning keeps moving.

Analysis begins with careful data cleaning, connecting each problem to its difficulty, and building baseline measures of solve rate, hint usage, and time across difficulty buckets and topics. Challenges in the logs, including many very short sessions (some students try only one problem), an imbalance in actions toward easier problems, and fluctuations in difficulty calibration, are addressed. The simulator is validated by checking that it reproduces expected patterns, such as higher solve rates on easier items and higher hint rates on harder ones. With this foundation, this project trains and monitors Q-learning, logging rewards, hint usage, solve rate, and chosen difficulty over time.

Preliminary offline results are promising. Compared with historical behavior estimated from the logs, the learned policy raises hint usage by about fifteen percent while keeping solve rate stable and time reasonable in the simulator. This suggests that the policy found a better match between student state and next-problem difficulty. However, possible real-world challenges are that offline gains can drop in live use, that teachers employ MathSpring in different ways (review vs. preview), and that short sessions limit how much any policy can adapt. These open issues motivate the future work outlined later, including a real-world pilot, better handling of imbalanced data, and richer measures of difficulty that may include carefully protected, optional signals of engagement.

DOI: 10.54254/2755-2721/2025.LD30332

In short, this paper shows a practical path for using reinforcement learning to improve a specific learning behavior—healthy, timely hint use—without sacrificing success. By restricting actions to difficulty choice, using a clear difficulty scale, and keeping the algorithm simple, this project aims to deliver a policy that teachers can understand and trust, and that can be tested fairly against the current system.

2. Background

2.1. Intelligent Tutoring System (ITS)

An Intelligent Tutoring System (ITS) is a computer program that supports learning by modeling how students think and learn [5]. In the 1980s, one of the first ITS—the LISP Tutor—was used in college courses [2]. LISP was an early programming language that is difficult for beginners, and while one-on-one tutoring is often more effective than a regular class, schools cannot give every student a personal tutor. The LISP Tutor filled this gap at scale: it set up problems, monitored a student's work step by step, and offered targeted guidance when the student got stuck. In practice, it eased the shortage of focused teaching time and helped many students reach their goals, with results that approached the effectiveness of a human tutor [2]. Other large-scale ITS such as Carnegie Learning's Cognitive Tutor demonstrated measurable learning gains in real classrooms [13], showing the potential for scalable personalized instruction.

As the field grew, new ITS approaches appeared. Constraint-Based Modeling (CBM) checks an answer against core rules for a problem [5]. Instead of marking only "right" or "wrong," CBM treats an answer as reasonable if it respects those rules, which supports student creativity. Affective Tutoring Systems (ATS) try to detect and respond to student emotions by observing signals such as facial expression, posture, voice, or even mouse movements [3,14]. Based on this "affective state," the system can suggest easier problems, provide helpful hints, or send encouraging messages to keep students engaged [3,14]. Other directions include game-based tutoring and adaptive web-based systems [4,5]. Together, these innovations aim to make learning more effective and personalized.

Recent progress in artificial intelligence opens new possibilities [6,7]. Classic ITS often relied on limited problem sets and fixed dialogs. Large language models (LLMs) can now generate fresh problems and explanations on demand [6,7]. By analyzing a student's current level and learning history, an ITS that uses an LLM can design practice that fits the student's needs [6]. LLMs also make feedback and conversation feel more natural, which can enhance engagement [7]. An ITS usually has four parts: a domain model (what to learn), a student model (what the student knows), a tutoring model (how to teach), and a user interface model (how to interact) [5]. LLMs are especially helpful in the tutoring and user interface models because they support personalization and fast, adaptive feedback [6,7].

2.2. MathsPring

As the internet matured, online intelligent tutoring systems evolved in parallel. The platform used in this study—MathSpring—grew out of Wayang Outpost, an SAT mathematics tutor that pioneered several design choices for math education: a broad library of problems with multimedia help (animations and sound), items embedded in narrative and fantasy contexts, and alternative teaching strategies tailored to students' differences in mental rotation and memory retrieval speed [10]. With Flash-based animations and adaptive hints, Wayang Outpost supported measurable learning gains and differentiation [10].

DOI: 10.54254/2755-2721/2025.LD30332

In 2014, with a shift to a modern web technology stack (e.g., HTML-based delivery), Wayang Outpost was updated and renamed MathSpring to reflect its more playful, game-like design. The system emphasizes progress in three areas—cognition, engagement, and effect—as core educational targets [3]. Its content expanded beyond SAT preparation to topics aligned with the U.S. Common Core State Standards (CCSS), organizing problems by underlying skills, a structure that remains in place [9]. A notable contribution is effort-based learning: MathSpring estimates problem difficulty for the student and infers the student's effort from signals such as time on task and number of attempts, then selects the next step (problem or hint) most likely to help [3,11]. The tutor updates its strategy in real time based on the student's recent work and employs learning companions to make practice feel more supportive and engaging [3].

With advances in AI and computer vision, MathSpring has also incorporated convolutional neural networks (CNNs) that analyze webcam input (when permitted) to detect potential off-task behavior, allowing the companion to invite students back into the activity [14]. Today, MathSpring is used in many U.S. K–12 classrooms, with well-defined effort measures and a large dataset that supports ongoing research and continuous improvement [3,9]. Figure 1 shows a typical interface that the student would interact with as they use the system.



Figure 1. A typical screen students see while solving problems in MathSpring.org

2.3. Reinforcement Learning

Reinforcement Learning (RL) is a way to train intelligent behavior through trial and error [12]. The idea is related to the "Law of Effect" from psychology: actions that lead to satisfaction tend to be repeated, and actions that lead to discomfort tend to be avoided because feedback reinforces behavior. In 1948, Alan Turing proposed using neural networks to mimic the way the human mind learns, and in 1952, Claude Shannon showed a mechanical "mouse" that could learn a path in a maze. These early ideas helped shape RL.

The key to RL, is setting up an environment and letting an agent interact with it. The agent observes its state, chooses an action, and receives a reward or penalty based on the result; over time, it learns which actions work best in different states to maximize total reward [12]. Common training settings include epsilon (exploration), alpha (learning rate), and gamma (discount for future rewards). States may be discretized into simple categories, or neural networks may be used to handle more complex inputs; in all cases, the key is the feedback loop between the agent and the environment [12].

This loop matches the needs of intelligent tutoring. In an ITS, the student and their work form the environment, and typical actions include feedback and next-problem selection [5]. The reward can

DOI: 10.54254/2755-2721/2025.LD30332

be set to reflect goals such as accuracy, healthy help-seeking behavior, and efficient time on task. By learning from many interactions, the system can adapt in real time and offer a more personalized learning path [5]. In educational settings, RL has been used to balance learning and engagement and to inform tutoring decisions, providing a practical foundation for policies that improve problem selection [4,12].

3. Methodology

The central aim of this study is to examine how intelligent algorithms can increase students' use of hints during mathematics learning, thereby supporting proactive help-seeking and deeper understanding [5,6]. Hint-seeking is treated as a key indicator of active engagement. Grounded in educational psychology—especially the zone of proximal development (ZPD), which describes the gap between what a learner can do independently and what they can accomplish with appropriate guidance—and informed by practical teaching experience, the hypothesis is that a problem-difficulty recommendation strategy aligned with each student's ZPD will encourage more timely, proactive hint use [15]. The goal is to maximize engagement and improve learning outcomes by matching instructional support to each student's immediate potential for growth.

This study adopts an offline policy-training approach. Because training does not occur online, the process is organized into several steps. First, existing student accuracy patterns are analyzed. Second, a Monte Carlo–based student–system interaction simulator is built. Third, a reward model is defined, and a reinforcement learning method is trained within the simulator, producing a new recommendation algorithm [4,12]. In Figure 2, a sketch of the process in the proposed approach is presented.

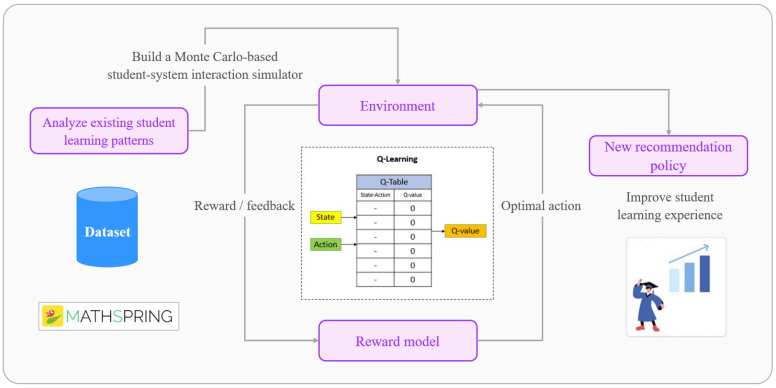


Figure 2. Overview of the proposed approach

3.1. Data cleaning

The dataset underwent systematic evaluation to identify critical data quality issues requiring remediation before analysis. Three primary categories of quality challenges were documented following established data quality assessment frameworks.

There are severe insufficiencies in five variables—adminFlag, authorFlag, newstart, newend, and postprocessinfo—each with more than 95% missing values. Because such gaps likely reflect systematic collection failures rather than random omission, these variables were excluded from analysis. Another five variables showed missing rates between 8.6% and 24.0%, prompting assessment for potential bias and the use of appropriate remedies such as multiple imputation. One

DOI: 10.54254/2755-2721/2025.LD30332

additional variable had less than 10% missing data, which was considered acceptable for direct analysis.

In addition to incomplete data, several fields required normalization. The variables mode, emotionAfter, and effort contained text strings instead of numeric codes, and were standardized for analytic compatibility. Timestamps (problemBeginTime, problemEndTime) were converted to proper DateTime formats to enable reliable time-based analyses.

There is also contamination in the form of sentinel values and inconsistent missing indicators. Eight numeric variables used –1 to denote missing data, affecting 41.6%–96.8% of entries and introducing ambiguity because these placeholders can be misread as valid data. In addition, some fields used empty strings or the literal text "NULL" to represent missing values. Together, these issues required comprehensive preprocessing and harmonization of missing-data conventions before any downstream analysis, in line with established data-quality management protocols.

3.2. Current student status

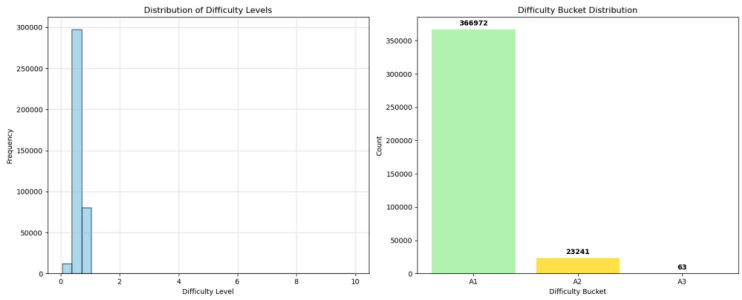
This project uses the MathSpring Student Engagement Performance Dataset from the Advanced Learning Technology Lab. The dataset contains 390,277 rows of student–system interactions with 39 variables that capture detailed learning behaviors and outcomes. These include key identifiers (ID, studId, sessionId, problemId, topicId) and temporal information (problemBeginTime, problemEndTime, timeInSession, timeInTutor) to track when students engage with problems. Performance metrics—such as timeToFirstAttempt, timeToSolve, numMistakes, numHints, numAttemptsToSolve, isSolved—provide insight into problem-solving efficiency and success. Pedagogical columns include mode (e.g., practice), mastery, and hint-related variables (timeToFirstHint, timeToSecondHint, timeToThirdHint, numHintsBeforeSolve, solutionHintGiven). Emotional and behavioral signals appear in emotionAfter (e.g., “Confidence,” “Frustration”), emotionLevel, emotionAsked, and effort (e.g., “SKIP,” “GIVEUP,” “GUESS”). Additional features include videoSeen, exampleSeen, textReaderUsed, collaboratedWith, and probDiff (problem difficulty). As a system log, the dataset monitors learning patterns, hint usage, affective states, and problem-solving strategies, making it suitable for analyzing learning trajectories, hint effectiveness, and engagement in mathematics education.

3.3. Difficulty distribution

Problem difficulty is computed empirically from time to solve, hints requested, and attempts before solve. In the database, difficulty values lie in (0, 1). The first digit after the decimal denotes the grade, and the remaining digits encode the computed difficulty. For example, problem 2856 has a difficulty of 0.5181, which indicates Grade 5 with a computed difficulty value of 181; for analysis, only the second digit after the decimal is used to place problems into three buckets: A1 (easier, levels 0–2), A2 (on-level, 3–6), A3 (harder, 7–9). In this scheme, 0.5181 falls into A1.

Figure 3a shows the initial distribution of difficulty levels, which is encoded into the computed difficulty in Figure 3b. Figure 3b shows that the dataset is predominantly composed of A1 problems, and most student interactions occur in this bucket. These buckets also define the action space for the RL simulator: given a student’s state, the system recommends the next problem’s difficulty bucket (A1, A2, or A3).

DOI: 10.54254/2755-2721/2025.LD30332



(a) Difficulty level distribution (b) student interaction by difficulty buckets

Figure 3. Analysis of difficulty levels and student interactions

3.4. Solve rate and hint usage rate

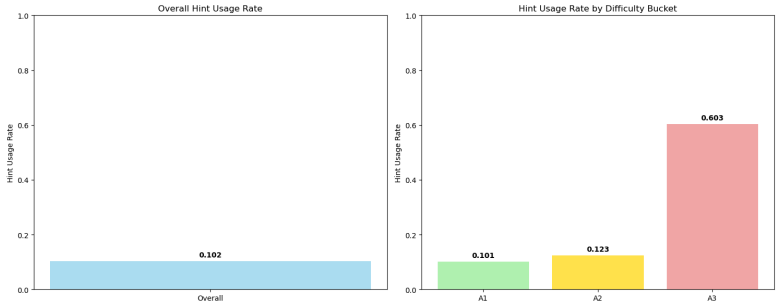


Figure 4. Hint usage summary

Figure 64 depicts the overall hint usage rate, as well as the hint usage rate in every difficulty bucket. The overall solve rate is 53.6%, while the hint usage rate is 10.2%. Examining hint usage within each difficulty bucket shows that harder problems are associated with higher rates of hint requests, which is consistent with the difficulty definition and supports its face validity.

3.5. Student-ITS interaction simulator

This project builds a data-driven simulator that models student–tutor interactions in MathSpring and trains a reinforcement learning (RL) agent to optimize hint-related pedagogical decisions [3,9,12]. The simulator abstracts each interaction as a Markov decision process (MDP) where the agent either (a) selects a hinting action or (b) recommends the next problem difficulty [16]. The design balances behavioral realism with tractability by combining features from historical logs with probabilistic outcome models for hint usage and problem solving [3,5,6].

The transformation of continuous variables into discrete bins represents a critical preprocessing step that bridges the gap between raw numerical data and categorical state representations suitable for reinforcement learning algorithms [12,16]. In an educational context, continuous variables such as solve rates, hint usage frequencies, and time-to-completion measurements are converted into binary categories (low/high, fast/slow) using median-based splitting strategies rather than arbitrary thresholds. This binning approach ensures that each category contains approximately equal numbers of student interactions, preventing the creation of imbalanced state distributions that could bias the learning algorithm toward certain regions of the state space. The choice of binary categorization reflects a deliberate design decision to maximize interpretability while minimizing computational complexity; by reducing continuous performance metrics to simple low/high or fast/slow

DOI: 10.54254/2755-2721/2025.LD30332

classifications, the system allows educators and researchers to easily understand the system's decision-making process, while the reinforcement learning algorithm operates within a manageable state space that facilitates efficient policy convergence [12,16]. This binning strategy maintains the essential information needed for effective educational interventions while transforming complex, continuous learning patterns into actionable, discrete states that can guide real-time problem difficulty adjustments and personalized learning experiences [3,5].

3.6. Student state variable design choices for Q-Learning

The first student design choice, QL-v1, adopts a compact approach focused on readiness and performance metrics by incorporating only three essential variables: difficulty level, solve rate, and hint usage rate. This minimalist design philosophy prioritizes computational efficiency and interpretability, ensuring that each state variable carries clear pedagogical meaning while maintaining a manageable state space. By limiting the state representation to these core performance indicators, the system can make rapid, interpretable decisions about problem difficulty adjustments without overwhelming the learning algorithm with excessive information.

The alternative design choice, QL-v2, embraces a richer state representation that captures process-oriented and affective aspects of student learning. This approach maintains the same three-variable constraint but shifts focus toward mastery level and time-based performance metrics alongside difficulty assessment. While both designs adhere to the strict three-variable limit to ensure computational tractability, QL-v2 emphasizes the temporal and mastery dimensions of learning, potentially capturing more nuanced patterns in student engagement and knowledge acquisition. The choice between these two approaches represents a fundamental decision in educational AI: whether to prioritize immediate performance feedback (QL-v1) or to incorporate longer-term learning trajectory information (QL-v2), both while maintaining the computational efficiency necessary for real-time educational interventions. Table 1 compares the two design choices based on their variables and total number of states.

Table 1. State space design and variable discretizing configuration

QL-v1 (Compact)			QL-v2 (Rich)		
Variable Name	Possible Values	Cardinality	Variable Name	Possible Values	Cardinality
difficulty_bucket	A1, A2, A3	3	difficulty_bucket	A1, A2, A3	3
solve_rate_5_opt	low, high	2	mastery_opt	low, high	2
hint_rate_5_opt	low, high	2	avg_time_5_opt	fast, slow	2
Total State Space	3 × 2 × 2 = 12	12 states	Total State Space	3 × 2 × 2 = 12	12 states

3.7. Simulator development

To enable safe off-policy reinforcement learning training, this project developed a data-driven simulator that learns the educational environment dynamics from historical student interaction logs. The simulator employs a supervised learning approach to model the conditional probability distributions $P(s_t, a_t)$, where s_t represents the current student state, a_t represents the system's action (difficulty recommendation), and s_{t+1} represents the resulting outcomes. State-action-outcome sequences are constructed from 390,000+ student interactions by extracting current state features, the difficulty level of the subsequently assigned problem, and the observed learning outcomes. The simulator uses separate predictive models for three key outcome variables: hint usage

DOI: 10.54254/2755-2721/2025.LD30332

behavior (binary classification), problem-solving success (binary classification), and time-to-solve (regression on log-transformed duration). This decomposition allows the simulator to capture the multi-faceted nature of student learning responses while maintaining computational tractability.

The simulator architecture employs Random Forest algorithms for all predictive models due to their robustness to missing data, ability to capture non-linear relationships, and natural handling of mixed categorical and numerical features. Training follows a student-based 70/15/15 split to prevent data leakage, ensuring that no student appears in both training and testing sets. For the binary classification tasks (hint usage and problem solving), the agent achieved prediction accuracies of 88-91% and 51-57% respectively, while the time prediction model attained a mean squared error of approximately 0.8 on log-transformed time values. The simulator incorporates proper handling of missing temporal data through imputation and filtering strategies, and validates its behavioral realism by ensuring that predicted outcomes follow expected difficulty progressions (easier problems yield higher solve rates and lower hint usage). This simulator serves as the foundation for training Q-learning policies in a controlled environment before deployment, enabling systematic exploration of different pedagogical strategies without affecting real students.

3.8. Reward function

The MathSpring RL project utilizes a dual-variant reward architecture, which allows direct empirical comparison between outcome-based and state-feature-based learning paradigms. For QL-v1, which emphasizes learner readiness and performance, the reward function employs a traditional outcome-based formulation:

$$Rv1 = 1.0 \times hint_used + 0.5 \times is_solved - 0.001 \times time_to_solve + 0.3 \times (hint_used \wedge is_solved) \quad (1)$$

This formula incentivizes the use of hints, successful problem solving, and time efficiency, with an additional bonus for achieving both hint usage and solution, fully aligning with the project's behavioral objectives. Positive reinforcement is awarded for requesting hints (+1.0) and solving problems (+0.5), while a small penalty is imposed for extended solving time (-0.001 per second). Furthermore, a reward of +0.3 is given for the combination of hint usage and successful problem solving. This approach directly reinforces target learning behaviors and optimizes for increased hint utilization without sacrificing educational achievement [3,11,12].

In contrast, QL-v2 is designed to promote process-oriented and affective goals through a state-feature-based reward structure:

$$Rv2 = 0.5 \times f(difficulty_{bucket}) + 0.8 \times f(mastery_{level}) + 0.6 \times f(time_{performance}) \quad (2)$$

Rewards under this paradigm are determined by the appropriateness of the learning context—specifically, problem difficulty, mastery level, and time performance—instead of relying exclusively on immediate outcomes. The difficulty component rewards engagement with moderately challenging tasks (A2 = 1.0) as opposed to easier (A1 = 0.0) or harder (A3 = 0.5) ones, while mastery is strongly incentivized at higher proficiency states (1.0 vs. 0.0 for low mastery). For time performance, efficient task completion is encouraged (fast = 1.0, slow = 0.0), resulting in tailored support within the student's zone of proximal development [3,15].

By calibrating both reward formulations to similar scales and grounding them in established RL methodology, this dual architecture enables rigorous evaluation of whether optimizing for immediate

DOI: 10.54254/2755-2721/2025.LD30332

target behaviors (QL-v1) or for progress in learning states (QL-v2) will yield superior long-term educational outcomes, particularly in hint usage and achievement.

3.9. Training setup

Policies are trained with tabular Q-learning, a model-free reinforcement learning method that estimates the value of taking an action in a given state, then following the current policy thereafter. The algorithm maintains a Q-table that maps a discretized representation of the environment's state to a small table of action and state pairs. After each interaction, the value for the encountered state–action pair is updated using a temporal-difference rule that nudges it toward the observed reward plus the discounted estimate of the best attainable value from the next state. The process can be formalized as follows:

$$Q\left(s_t, a_t\right) \leftarrow Q\left(s_t, a_t\right) + \alpha\left(r_{t+1} + \gamma Q\ \left(s_{t+1}, a^{'}\right) - Q\left(s_t, a_t\right)\right) \tag{3}$$

Action selection follows an ε-greedy strategy to balance exploration and exploitation. With probability ε the agent chooses a random action to probe alternatives; Otherwise, it chooses the action with the highest current Q-value. ε has an initial value in the range of roughly 0.3–0.4 and decays episode by episode toward a small floor near 0.01, allowing broad exploration early and increasingly greedy behavior as learning stabilizes. For longer trainings, this project uses a simple three-phase schedule—high exploration, moderate exploration, and fine-tuning—implemented by adjusting the decay rate across phases. Q-learning updates use learning rates between 0.05 and 0.10 and discount factors between 0.90 and 0.95, which was found to provide a good stability–plasticity trade-off in this domain.

Because observations contain continuous features (e.g., normalized time and counts) alongside categorical effects, the continuous components are discretized into coarse bins before indexing the Q-table. This preserves ordinal structure while limiting the number of distinct keys, encouraging generalization across near-identical contexts. Training proceeds in episodes of fixed horizon (on the order of tens of steps per episode), logging total reward and task-relevant behavioral metrics for monitoring.

4. Results

This section reports training outcomes, behavioral metrics, and supporting analyses for the reinforcement learning (RL) hint optimization experiments. Presented below are convergence curves, comparative performance under alternative reward designs and action spaces, and diagnostics related to problem difficulty calibration.

4.1. Initial experimental failures: understanding the problem space

The first experimental attempts revealed the complexity of the reward function design problem. The initial balanced reward approach, using moderate hint bonuses (+2.0) and solve bonuses (+3.0), resulted in a dismal 5.3% hint rate and 43.4% solve rate. This failure illuminated a critical insight: the simulator's inherent bias towards low hint usage could not be overcome through simple positive reinforcement alone. Unfortunately, the second set of states for engagement optimization (QL-v2) seems not to be working; thus, this method was dropped them at this stage of the research.

DOI: 10.54254/2755-2721/2025.LD30332

Recognizing the need for more forceful intervention, this project implemented an aggressive hint encouragement strategy with 4x multipliers and elevated hint bonuses (+8.0). This approach, while theoretically sound, produced catastrophic results: 0.0% hint rate and 12.1% solve rate. The failure revealed that excessive positive reinforcement could destabilize the learning process, causing the agent to ignore the intended behavioral cues entirely.

4.2. The penalty-based breakthrough: paradigm shift in approach

The scientific breakthrough came through a fundamental paradigm shift: instead of trying to force positive behavior through rewards, this project implemented a penalty-based system that punished the absence of desired behavior. This approach, inspired by behavioral psychology principles, applied negative reinforcement (-3.0 penalty for no hints) to create a "default hint usage" behavior pattern.

The penalty-based approach achieved a great success (25.1% hint rate) while maintaining a desirable solve rate 54.5%. The full reward set up is shown in Table 2:

Table 2. Reward function setup

Behavior	No Hint	Hint Used	Problem Solved	Time Penalty	Early finish
Reward	-3.0	+1.0	+3.0	-0.001	+0.5

Notes. Five-component reward system with hierarchical structure: solve reward (+3.0) for problem completion, hint penalty (-3.0) for bypassing assistance, hint usage reward (+1.0) for appropriate help-seeking, time penalty (-0.001) for efficiency, and early finish bonus (+0.5) for timely completion. Negative values discourage undesired behaviors while positive values reinforce learning objectives and strategic help-seeking.

As training proceeds over episodes, the policy shifts from broad exploration to more targeted exploitation. Episode reward, hint-usage rate, and problem solve rate are tracked throughout to monitor stability and learning progress.

DOI: 10.54254/2755-2721/2025.LD30332

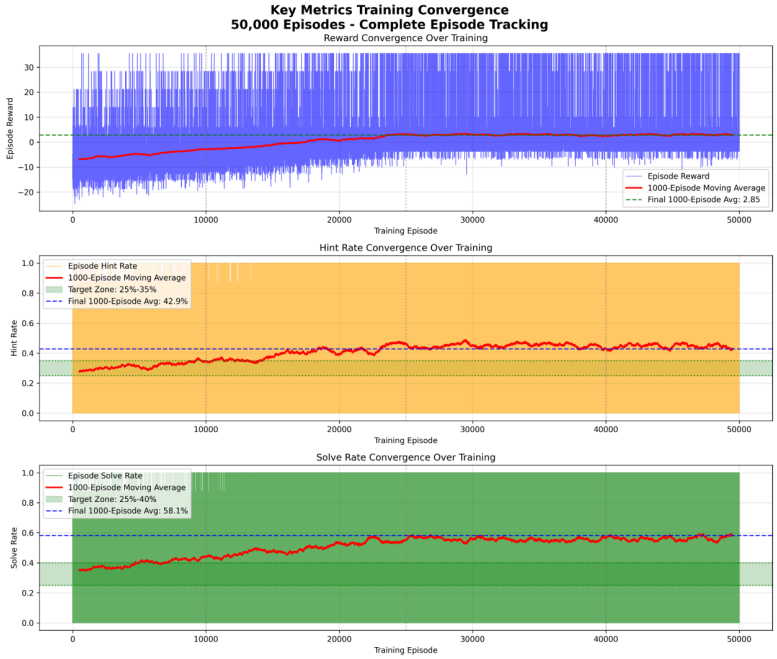


Figure 5. Training log trace of reward, hint rate, and solve rate per episode

4.3. Performance evaluation

The individual episode may have a short interaction window with the agent thus leading to a highly fluctuating graph; taking the average of the nearby 1000 episodes could present a clear pattern. Based on the key metrics convergence, Figure 5, the training demonstrates a remarkable evolution from initial instability to strong convergence over 50,000 episodes. The reward convergence shows high variance in early episodes with individual rewards fluctuating between -20 and 30, but gradually stabilizes as the 1000-episode moving average transitions from negative values around -7 to a positive convergence around 2.85 by the final episodes, indicating successful learning progression. The hint rate exhibits a steady upward trend from an initial 25% baseline, with the moving average climbing consistently and ultimately converging to 42.9% by the end of training, which exceeds the target zone of 25-35% but demonstrates the agent's learned preference for strategic hint usage. Most impressively, the solve rate shows the strongest positive trajectory, starting around 35% and steadily climbing to a final convergence of 58.1%, significantly outperforming the target range of 25-40% and indicating that the Q-Learning agent successfully developed a policy that balances hint utilization with effective problem-solving strategies. The convergence patterns suggest that after approximately 20,000 episodes, the agent achieved stable performance across all metrics, with the moving averages showing minimal fluctuation in the latter half of training, demonstrating successful policy optimization and learning convergence. Table 3 shows a general increase in the reward, hint rate, and solve rate as the training progresses from early training to mid and late training over 50,000 episodes.

DOI: 10.54254/2755-2721/2025.LD30332

Table 3. Training performance summary - complete episode tracking (50,000 episodes)

Metric	Early Training (1-1k)	Mid Training (24k-25k)	Late Training (49k-50k)	Final Evaluation
Reward	-6.96	3.25	2.85	-1.80
Hint Rate	27.6%	46.0%	42.9%	25.1%
Solve Rate	35.1%	55.6%	58.1%	54.5%

5. Conclusion and future work

5.1. Conclusion

This project used data from the MathSpring dataset. The data is first cleaned by deleting invalid information, then analyzed by composing graphs and tables. By reconstructing the data and designing an effective reward function, the study improved hint usage rate from 10.2% to 25.1%.

5.1.1. Data cleaning

The initial data contained a lot of missing information as well as improperly encoded data. This invalid data was treated either through deleting entire lines that did not contain enough usable information, or through filling missing sections with replaced values. This method does not significantly affect the accuracy of the data.

5.1.2. Redistributing

The graphs created from the dataset show that many of the variables were strongly skewed to the right. Training a model based on this initial data would not be effective. By sorting the data for each variable into three brackets, this project reconstructed data that was more widely distributed. This meant that the model had enough data on the high end.

5.1.3. Reward function

The reward function needs to be balanced so that the model won't choose extreme policies. To make the reward function effective, the simulation was tested with different formulas. It was found that the reward function needs to emphasize the hint usage rate while also giving an additional reward for solving the problem using hints. This ensures that the model gives recommendations that could encourage students to think, but are also not too difficult.

5.2. Future work

Results show that Reinforcement Learning could improve hint usage rate from 10.2% to 25.1%. However, since this is a virtual study, not a real-world application, there are some limitations, which are addressed below.

5.2.1. Limitation across classroom use cases

Different teachers use MathSpring in different ways. Some use it to review past lessons, so students already know most of the content and perform very well. Others use it to preview new lessons, so

DOI: 10.54254/2755-2721/2025.LD30332

students face harder material and may need more help. A single, uniform recommendation policy could miss these differences. In future work, this project will add a clear "use mode" at the start of each session—review or preview—and let teachers set it or let the system infer it from early answers and time-on-task. The policy will then adjust difficulty and hint style to match the chosen mode. Results for review and preview could be reported separately so that improvements are not hidden by averaging.

5.2.2. Low interaction per session

Many students log in, do only one problem, and leave. This short interaction makes it hard to learn stable long-term patterns. A possible solution is session starters and closers that encourage a minimum of three to five problems, such as a short warm-up, a quick progress badge, or a simple exit question that unlocks after a few attempts. For analysis, short visits from the same student across days will be connected to form longer timelines, which allows for testing on whether policies learned from longer traces transfer to the "one-and-done" users.

5.2.3. High-dimensional data and simpler learning

Today's computers make it possible to use more variables without heavy hand-crafted binning. A next step is to keep Q-learning but replace the hand-binned state with a small neural network that reads continuous features directly and outputs a compact state code. This keeps the method simple while reducing information loss from binning. A comparison of the current tabular Q-learning to this "approximate Q-learning" on the same train/validation/test split could help check whether the new inputs help the policy choose better difficulties for different students.

In conclusion, this project has successfully developed a Q-Learning agent for educational mathematics problem-solving that demonstrates effective convergence over 50,000 training episodes. The penalty-based reward function with fine-tuned parameters achieved a final hint rate of 25.1% within the target range of 25-35%, while the agent learned to balance hint usage with problem-solving effectiveness. The training process revealed stable convergence patterns after approximately 20,000 episodes, with the model successfully optimizing its policy through strategic exploration and exploitation. Hopefully, this research will inspire further investigation into reinforcement learning applications in educational technology, as improvements in adaptive learning systems have the potential to revolutionize personalized education and enhance student engagement across diverse mathematical domains.

References

[1] Park University. (2024, July 15). AI in education: The rise of intelligent tutoring systems. Park University Blog. https: //www.park.edu/blog/ai-in-education-the-rise-of-intelligent-tutoring-systems/

[2] Anderson, J. R., & Reiser, B. J. (1985). The LISP tutor. Byte, 10(4), 159–175. https: //doi.org/10.5555/3828.3831

[3] Arroyo, I., Woolf, B. P., Burelson, W., Muldner, K., Rai, D., & Tai, M. (2014). A multimedia adaptive tutoring system for mathematics that addresses cognition, metacognition and affect. International Journal of Artificial Intelligence in Education, 24, 387–426. https: //doi.org/10.1007/s40593-014-0023-5

[4] Sawyer, R., Rowe, J., & Lester, J. (2017). Balancing learning and engagement in game-based learning environments with multi-objective reinforcement learning. In X. Hu, T. Barnes, & S. Stamper (Eds.), Artificial intelligence in education: 18th International Conference, AIED 2017, Proceedings 18 (pp. 323–334). Springer International Publishing.

[5] Alkhatlan, A., & Kalita, J. (2018, December 22). Intelligent tutoring systems: A comprehensive historical survey with recent developments. arXiv. https: //arxiv.org/abs/1812.09628

DOI: 10.54254/2755-2721/2025.LD30332

[6] Lin, C.-C., Huang, A. Y. Q., & Lu, O. H. T. (2023). Artificial intelligence in intelligent tutoring systems toward sustainable education: A systematic review. Smart Learning Environments, 10(1), 41. https: //doi.org/10.1186/s40561-023-00263-9
[7] Maity, S., & Deroy, A. (2024, October 14). Generative AI and its impact on personalized intelligent tutoring systems. arXiv. https: //arxiv.org/abs/2410.10650
[8] Bhutoria, A. (2022). Personalized education and artificial intelligence in the United States, China, and India: A systematic review using a human-in-the-loop model. Computers and Education: Artificial Intelligence, 3, 100068. https: //doi.org/10.1016/j.caeai.2022.100068
[9] MathSpring. (n.d.). MathSpring. https: //mathspring.org/
[10] Arroyo, I., Beal, C., Murray, T., Walles, R., & Woolf, B. (2004). Wayang outpost: Intelligent tutoring for high stakes achievement tests. In J. C. Lester, R. M. Vicari, & F. Paraguaçu (Eds.), Proceedings of the 7th International Conference on Intelligent Tutoring Systems (ITS 2004) (pp. 468–477). Springer.
[11] Arroyo, I., Mehranian, H., & Woolf, B. P. (2010). Effort-based tutoring: An empirical approach to intelligent tutoring. In R. S. J. d. Baker, A. Merceron, & P. I. Pavlik Jr. (Eds.), Proceedings of the 3rd International Conference on Educational Data Mining (pp. 1–10). International Educational Data Mining Society.
[12] Mon, B. F., Wasfi, A., Hayajneh, M., Slim, A., & Abu Ali, N. (2023). Reinforcement learning in education: A literature review. Informatics, 10(3), 74. https: //doi.org/10.3390/informatics10030074
[13] Koedinger, K. R., Corbett, A. T., & Ritter, S. (2000). Carnegie Learning's Cognitive Tutor: Summary research results. Carnegie Learning, Inc.
[14] Lee, W., Allessio, D., Rebelsky, W., Gattupalli, S. S., Yu, H., Arroyo, I., Betke, M., Bargal, S., Murray, T., & Sylvia, F. (2022). Measurements and interventions to improve student engagement through facial expression recognition. In M. M. Soares, E. Rosenzweig, & A. Marcus (Eds.), International Conference on Human-Computer Interaction (pp. 286–301). Springer.
[15] Murray, T., & Arroyo, I. (2002). Toward measuring and maintaining the zone of proximal development in adaptive instructional systems. In S. A. Cerri, G. Gouardères, & F. Paraguaçu (Eds.), International Conference on Intelligent Tutoring Systems (pp. 749–758). Springer.
[16] Wang, F. (2018). Reinforcement learning in a POMDP based intelligent tutoring system for optimizing teaching strategies. International Journal of Information and Education Technology, 8(8), 553–558. https: //doi.org/10.18178/ijiet.2018.8.8.1104