Noname manuscript No.
(will be inserted by the editor)

Leveraging Granularity: Hierarchical Reinforcement Learning for Pedagogical Policy Induction

Guojing Zhou · Hamoon Azizsoltani · Markel Sanz Ausin · Tiffany Barnes · Min Chi

Received: date / Accepted: date

Abstract In interactive e-learning environments such as Intelligent Tutoring Systems, pedagogical decisions can be made at different levels of granularity. In this work, we focus on making decisions at *two levels*: whole problems vs. single steps and explore three types of granularity: *problem-level only (Prob-Only)*, *step-level only (Step-Only)* and *both problem and step levels (Both)*. More specifically, for Prob-Only, our pedagogical agency decides whether the next *problem* should be a worked example (WE) or a problem-solving (PS). In WEs, students observe how the tutor solves a problem while in PSs students solve the problem themselves. For Step-Only, the agent decides whether to elicit the student's next solution *step* or to tell the step directly. Here the student and the tutor *co-construct* the solution and we refer to this type of task as collaborative problem-solving (CPS). For Both, the agency first decides whether the next problem should be a WE, a PS, or a CPS and based on the problem-level decision, the agent then makes step-level decisions on whether to elicit or tell each step. In a series of classroom studies, we compare the three types of granularity under *random yet reasonable* pedagogical decisions. Results showed that while Prob-Only may be less effective for High students, Step-Only may be less effective for Low ones, Both can be effective for both High and Low students. Motivated by these findings, we propose and apply an offline, off-policy Gaussian Processes based *Hierarchical Reinforcement Learning (HRL)* framework to induce a *hierarchical pedagogical policy* that makes adaptive, effective decisions at both the problem and step levels. In an empirical classroom study, our results showed that the HRL policy is significantly more effective than a Deep Q-Network (DQN) induced step-level policy and a random *yet reasonable* step-level baseline policy.

Guojing Zhou · Hamoon Azizsoltani · Markel Sanz Ausin · Tiffany Barnes · Min Chi
Department of Computer Science, North Carolina State University
Raleigh, NC, 27695, USA
E-mail: {gzhou3,hazizso,msanzau,tmbarnes,mchi@ncsu.edu}

Keywords Hierarchical Reinforcement Learning · Decision Granularity · Pedagogical Policy

1 Introduction

Worked examples (WE) and problem-solving (PS) have a long history of being used for instructional purposes [58,27]. In PS, students are given tasks to complete either independently or with assistance; while in WEs, students are given detailed solutions. WEs have been used as an instructional approach for a long time [58]: early WEs were found in historical records such as Egyptian papyri, Babylonian tablets, and later in copies of lost Chinese manuscripts dating back to thousands of years ago [27]. At the end of the worked solutions, it is often stated that "this way you may solve similar problems," or "by the same method solve all similar problems,". Similarly, evidence of PS was also found in early Egyptian, Babylonian, and Chinese artifacts [57]. This includes pure collections of practice problems, reflecting the principle held by early educators that learning is an active process in which problem-solving should be involved. Accordingly, there has been a growing interest in utilizing WEs and PS in e-learning environments such as intelligent tutoring systems (ITSs) [56,31,30,29,59,38,47,33,42,70,69,68,71,67].

In STEM domains like math, probability, and logic, solving a *problem* often requires producing an argument, proof, or derivation consisting of one or more inference *"steps"*, each of which is the result of applying a domain principle or rule. VanLehn described tutoring in such domains as a two-loop procedure [61]: the outer loop governs problem-level pedagogical decisions such as selecting the next problem or task for the student to work on while the inner loop controls step-level pedagogical decisions such as whether to give feedback or to prompt the student with an example. *Pedagogical policies* are used to decide what action to take next in the face of alternatives. In this work, three types of granularity are explored: problem-level only (Prob-Only), step-level only (Step-Only), and both problem-level and step-level (Both).

Prob-Only: When a tutor determines the next problem or task, it often decides whether to show students how to solve the next problem directly using a *WE* or asks students to solve the problem on their own via *PS*. Presenting a WE is more *passive* since it often does not require students to act much; while PS is more *active* since it is expected that students would produce the solution with or without the tutor's assistance. Prior studies have shown that combining WEs with PSs can be more effective than using PSs alone [31,30, 29,56]. Note that we only focus only on WE and PS in this work and there are other similar instructional interventions such as incomplete examples or faded worked examples (FWEs) where students are presented with an example where certain steps are left out for them to complete [38,47].

Step-Only: Alternatively, the tutor can make decisions inside a problem to decide whether to *elicit* the next solution step from the student or to show or *tell* it directly. We refer to such decisions as *elicit/tell.* While a lecture can

be viewed as a monologue of an unbroken series of tells, human one-on-one tutoring often interleaves elicits and tells through which students *co-construct* the solution with the tutor. In the following, we refer to the interleaving of elicits and tells as *collaborative problem solving (CPS)*. Prior research showed that CPS can be more effective than elicits only [42] and students sometimes use bottom-out hints as tells to help them learn [50].
Both: Finally, a tutor can make decisions at both the problem and step levels. In this case, it first makes a problem-level decision to decide whether the next problem should be WE, PS, or CPS. If CPS is selected, it will make step-level decisions on whether to elicit or tell. While WE can be seen as an extreme case of CPS where tell is selected in every step, we argue that in reasonable decision-making (not random), a WE is fundamentally different from an all-tell CPS in that WE is *determined* at the problem level before the student starts working on it; while tells in CPS are decided at the step level partly based on the student's behavior in the current problem. The same argument applies to PS vs. all-elicit CPS.

Prior research has widely focused on either problem-only [31,30,33] or step-only [38,47,42] decisions. However, as far as we know, no prior work has directly compared all three types of granularity. Additionally, there is no well-established theory or consensus on how to make decisions at the two levels effectively. In this work, we first investigate the impact of decision granularity on student learning, and then, apply hierarchical reinforcement learning (HRL) to induce pedagogical policies that make decisions at both the problem and step levels.

1.1 Our Studies

In a series of four classroom studies conducted in the Fall 2014-2017 semesters, we investigate the impact of granularity on student learning. In Fall 2014, we directly compare the three types of granularity: 1) Prob-Only, 2) Step-Only, and 3) Both with two baseline conditions: WE-only (WE) and PS-only (PS). The study was conducted in the domain of probability, and we employed an ITS to strictly control the content to be equivalent across the five conditions. More specifically, all students went through the same general procedure and studied the same materials. During ITS training, all students received the same problems in the same order, and WE, PS, and CPS covered the same content. The only difference between the five conditions was the decision granularity. Results showed that decision granularity can have an impact on students' time on task in that: WE $<$ Prob-Only $<$ Step-Only, Both $<$ PS. However, there was no significant difference among the five conditions in learning performance.

The three follow-up studies were conducted under the same settings, but each of them included only one or two types of granularity. To further examine the impact of decision granularity, we combined the data collected in the four studies and conducted a post-hoc analysis focusing on the three granularity types. Overall, there was still no significant difference between the three conditions in learning performance. Considering the fact that our ITS mainly focuses on teaching a strategy for solving multiple-principle problems, we split students into different incoming competence groups based on their pretest performance on single- and multiple-principle problems and conducted a two-factor post-hoc analysis on granularity and incoming competence. Results showed that Prob-only may be less effective for High incoming competence students, Step-only may be less effective for Low ones, and more importantly, Both can be effective for both High and Low students. For time on task, the Prob-Only and Both conditions spent significantly less time than the Step-Only condition. The results suggest that granularity indeed can have an impact on student learning. This supports our hypothesis that WE, PS and CPS are different types of learning activities. Additionally, one implication of the results is that the impact of granularity depends on students' competence level, which suggests that effective interventions should adapt WE, PS and CPS to students' current learning state. To investigate whether adapting WE, PS and CPS can result in improved student learning performance, we apply hierarchical reinforcement learning (HRL) to induce pedagogical policies that make decisions at both the problem and step levels.

In an ITS, the tutor's decisions can be viewed as a temporal sequence, each of which affects the student's successive actions and performance. Its impact on student learning often cannot be observed immediately, and the effectiveness of one decision may also depend on subsequent decisions. Ideally, the tutor should adapt its pedagogical decisions to meet students' specific learning needs [2,35]. In recent years, there has been growing interest in applying data-driven approaches, such as reinforcement learning (RL), to directly induce pedagogical policies from student-system interaction logs. RL algorithms are designed to induce decision-making policies that specify the best action to take in any given situation so as to maximize a cumulative reward. A number of researchers have studied applying existing RL algorithms to improve the effectiveness of ITSs (e.g., [11,49,28,40,39,36,12,54,22,23,71,72]).

In an unpublished study, we compared the effectiveness of RL-induced problem-level and step-level policies. The same RL algorithm was applied to random problem- and step-level exploration data to induce these two policies. In a classroom study, we compared the two RL policies with a random problem-level policy and a random step-level policy. Note that since the problem level WE/PS and the step level elicit/tell are always considered to be reasonable interventions in our learning context, our random policies are *random yet reasonable*. Results showed that there was no significant difference between the two RL-induced policies in effectiveness, and none of them significantly outperformed the two random baseline policies. The results suggest that RL induced pedagogical policies that make decisions at a single granularity level may not always be effective.

Given the findings that RL induced problem- and step-level policies may not always be effective and motivated by the implication of the Granularity studies that WE, PS, and CPS are different types of instructional interventions, we then apply HRL to induce a policy that explicitly adapt WE/PS/CPS at the problem level and elicit/tell at the step level. When there are decisions to make, the HRL policy first decides whether the next problem should be WE, PS, or CPS. If WE is selected, an all-tell step policy will be carried out; if PS is selected, an all-elicit policy will be executed; finally, if CPS is selected, the tutor will decide whether to elicit or tell the next step based on the corresponding step-level policy. Again, the training data were collected by training students with our ITS using random yet reasonable policies.

In a classroom study, we compare the HRL policy with a Deep Q-network (DQN) induced step-level policy and a *random yet reasonable* step-level baseline policy. The DQN policy is induced using the same data and inferred immediate rewards that are used for inducing the HRL policy. Results showed that the HRL policy was significantly more effective than the DQN and the random baseline policy. For time on task, there was no significant difference between the HRL condition and the baseline condition, but the former (HRL) spent significantly more time than DQN. Finally, the induced HRL policy was more likely to select PS and CPS than WE. The results suggest that HRL can make effective pedagogical decisions to improve student learning, and HRL can be more effective than flat RL in pedagogical policy induction.

1.2 Contributions

This paper represents an extended version of our previously-published work in AIED 2019 [67]. It has been extended by: including the Granularity studies and analyzing and discussing the HRL study results from the decision granularity perspective. The contributions of this work are:

– For learning science, we investigated the impact of granularity on student learning. Results suggest that the granularity can have an impact on students' time on task, and that its impact on learning performance depends on students' current competence level.
– From a machine learning perspective, we proposed an offline off-policy HRL framework that can induce hierarchical pedagogical policies from pre-collected student-system interaction data.
– The HRL study provided empirical evidence showing that HRL can induce effective pedagogical policies, and HRL can be more effective than flat RL in pedagogical policy induction.
– Our analysis of the HRL policy's decision-making sheds some light on how to leverage the impact of decision granularity to improve student learning.

2 The Granularity Studies

The Granularity studies investigate the impact of three types of granularity: Prob-Only, Step-Only, and Both on student learning. This section describes the background, experimental design, results, and discussion of the Granularity studies.

2.1 Background – WE vs. PS vs. FWE vs. CPS

A lot of prior research has investigated the impact of worked example (WE), problem solving (PS), faded worked example (FWE) and collaborative problem solving (CPS) on student learning [56,31,30,29,59,38,47,33,42,70,69,68, 67]. Some of the works have focused on selecting WE vs. PS at the problem level. For example, McLaren et al. compared WE-PS pairs with PS-only [31]. Students in the WE-PS condition were given 5 WE-PS pairs, while those in the PS-only condition were required to solve 10 problems. Experimental results showed no significant difference between the two conditions in learning performance. However, the WE-PS condition spent significantly less time than the PS-only condition. In a follow-up study, McLaren et al. compared three conditions: WE-only, PS-only, and WE-PS pairs [30]. As before, there was no significant difference among the three conditions in terms of learning performance, but the WE condition spent significantly less time than the other two conditions; and no significant time on task difference was found between the PS and WE-PS conditions. In short, the two studies showed that problem-level WEs can be as effective as PSs, and the former can take significantly less time than the latter.

Some other work designed pedagogical policies based on the expertise reversal effect, which states that the relative effectiveness of instructional methods reverses as levels of learner knowledge in a domain change [24]. A widely used design is to give WE at the beginning of training, then gradually fade out the tell steps in an example, and finally require the student to solve the whole problem. Renkl et al. compared WE-FWE-PS with WE-PS pairs [38]. Students in the WE-FWE-PS condition studied and completed WEs, FWEs, and PSs in a fixed order, while those in the WE-PS condition received the same problems in WE-PS pairs. Results showed that the former significantly outperformed the latter in learning performance; while no significant difference was found between them in time on task. Similarly, Schwonke et al. compared WE-FWE-PS with PS-only [47]. Overall, there was no significant difference between the two conditions in terms of learning outcomes, but the WE-FWE-PS condition spent significantly less time than the PS-only condition. In sum, the studies showed that alternating among WE, PS and FWE can be more beneficial than using WE and PS only.

Alternatively, pedagogical decisions can be adaptively made at the problem level or the step level. Najar et al. compared adaptive WE/FWE/PS with WE-PS pairs [33]. The former used expert rules to make decisions based on students' learning states. Results showed that adaptive WE/FWE/PS significantly outperformed WE-PS pairs in terms of learning outcomes, and the former also spent significantly less time on task. In another study, Salden et al. investigated the impact of making adaptive step-level decisions by comparing three conditions: WE-FWE-PS, CPS, and PS-only [42]. The WE-FWE-PS condition gives WEs, FWEs, and PSs in a fixed order, while the CPS condition used an adaptive pedagogical policy – expert rules combined with data-driven student models – to decide whether to elicit or tell the next step.

Their results showed that CPS outperformed WE-FWE-PS, which in turn outperformed PS-only, and no significant time on task difference was found among the three conditions. Note that in this study, only the CPS condition involved adaptive decisions. Therefore, it is not clear whether it was the adaptation or the granularity that made the CPS condition more effective than the other two conditions. In short, previous studies have shown that making adaptive problem-level or step-level decisions can be more effective than using pre-determined ordering.

Overall, results from previous studies suggest that effective policies can lead to better learning performance or behavior. In addition, adaptive policies can be more effective than fixed ones. However, all previous works focused on either the problem-level or step-level decisions. To the best of our knowledge, no prior study has investigated Both levels. Additionally, in prior studies, the pedagogical decisions were made following certain fixed or adaptive policies. Therefore, it is not clear whether the impacts were caused by policy or granularity. In the Granularity studies, we directly compare the three types of granularity and use random policies to factor out the impact of pedagogical policy.

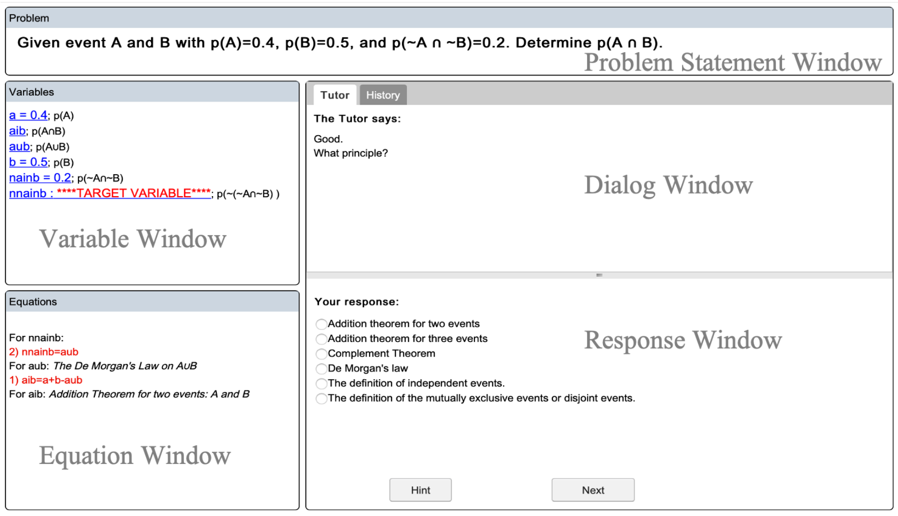
2.2 Experiment Setup

2.2.1 Conditions

In a series of four classroom studies conducted in the Fall semesters of 2014-2017, we investigate the impact of granularity on student learning by comparing the three types of decision granularity with two baseline conditions: WE-only (WE) and PS-only (PS). Based on the amount of work students need to do, the five conditions are:

– **WE:** where all problems are WEs;
– **Prob-Only:** where problem-level decisions are randomly made to decide whether the next problem should be WE or PS;
– **Step-Only:** where step-level decisions are randomly made on whether to elicit or tell the next step;
– **Both:** where the tutor first randomly decides whether the next problem should be WE, PS, or CPS, and if CPS is selected, it will randomly decide whether to elicit or tell each step;
– **PS:** where all problems are PSs.

The three granularity conditions use *random* decisions to factor out the impact of pedagogical policies. The Fall 2014 study includes all five conditions, while the following three involve one or two granularity types. In our analysis, we first focus on the empirical Fall 2014 study and then combine students across all four studies by their condition for post-hoc comparison. The followings describe the experiment setup and analysis results.


Problem

Given event A and B with p(A)=0.4, p(B)=0.5, and p(~A ∩ ~B)=0.2. Determine p(A ∩ B).

Variables

a = 0.4; p(A)
aib; p(A∩B)
aub; p(A∪B)
b = 0.5; p(B)
nainb = 0.2; p(~A∩~B)
nnainb : ****TARGET VARIABLE****; p(~(~A∩~B))

Tutor History

The Tutor says:

Good.
What principle?

Equations

For nnainb:
2) nnainb=aub
For aub: *The De Morgan's Law on A∪B*
1) aib=a+b-aub
For aib: *Addition Theorem for two events: A and B*

Your response:

- Addition theorem for two events
- Addition theorem for three events
- Complement Theorem
- De Morgan's law
- The definition of independent events.
- The definition of the mutually exclusive events or disjoint events.

Fig. 1 The interface of our probability tutor (Pyrenees)

2.2.2 Participants

The participants in all four studies were undergraduate students enrolled in the discrete math class offered by the computer science department at North Carolina State University. The study was given to them as one of the regular homework assignments. They had one week to complete the study and were graded based on their demonstrated effort, not performance.

2.2.3 Probability Tutor

The tutor we used in this study (named Pyrenees) is a web-based ITS that teaches 10 probability principles such as the Addition Theorem and Bayes' Theorem. The training is conducted by guiding students to complete training problems (See Appendix A for an example of the training problems). All instructions, such as explanations, feedback, and hints, are presented using natural language.

Figure 1 shows the interface of our ITS, which is divided into multiple windows. Through the dialogue window, the tutor provides messages to the students such as explaining an tell step, or giving hints. Students can enter their inputs, such as writing an equation or selecting an answer, through the response window. Any variables or equations that are defined in the training process are displayed on left side of the screen for reference.

Target Variable Strategy: Each training problem requires applying multiple principles to solve, and the tutor teaches students to use the Target Variable Strategy (TVS) to solve problems. In the domain of probability, TVS solves a problem by building a inference path backward from the goal to the given events [62, 9, 10]. The problem solving procedure involves three main phases: 1) translating the problem statement, 2) applying principles and generating equations, and 3) solving equations. In the first phase, each given probability event is defined as a known variable and the goal is defined as a sought (unknown) variable. In the second phase, probability principles are applied to build an inference path. Each principle application involves three steps: 1) *selecting a target variable* from the ones that are marked as sought, 2) *selecting a principle* to apply to the target variable, and 3) *writing an equation* for the selected principle. If the equation involves any undefined variables, the tutor or the student must define them before writing the equation. Once the equation is generated for the target variable, the tutor will remove its sought mark, but the new defined variables will be marked as sought. The principle application process repeats until there is no sought variable left. In the third phase, the tutor or the student solves the equations generated in the second phase to determine the value of the goal events.

By using TVS, the tutor mainly teaches students the strategy for solving multiple-principle problems. Target variable selection teaches students to determine where to extend the current solution path; principle selection helps students learn when a principle should be used; and intermediate variable definition and equation writing shows the specific process about how the solution path is extended.

Elicit vs. Tell: For each step in a problem, the tutor can either elicit it from the student or tell it to the student directly. In an elicit step, the tutor gives a question for the student to answer; while in a tell step, the tutor directly shows how to complete the step. The tutor provides feedback and hint in elicit steps. When the student gives a wrong answer, the tutor provides a feedback on it, which consists of a short message indicating that the answer is wrong and a hint about how to solve the step. Hints can also be requested proactively by clicking the [Hint] button. Our ITS uses the same set of messages for incorrect-answer hints and on-demand hints. The hint messages were organized in an increasingly specific order. The last message in the sequence, the bottom-out hint, shows the student exactly what to do.

2.2.4 Procedure and Grading

Procedure: All students went through four phases: 1) textbook, 2) pre-test, 3) training on the ITS, and 4) post-test. During **textbook**, all students read a general description of each principle, reviewed several examples, and solved several training problems. The students then took a **pre-test**, which contained a total of 14 single- and multiple-principle problems (see Table 1 below for example test problems). For each problem, students need to give a detailed step-by-step solution that specifies the name of the principles applied and the corresponding equations. No feedback was given on their answers, and they were not allowed to go back to earlier questions (this was also true for the post-test). During **training on the ITS**, all students received the same 12

Table 1 Single-principle Problem vs. Multiple-principle Problem

Single-principle Problem
Question: If $p(A \cap B) = 0.2$ and $p(B) = 0.5$, find $P(A\|B)$.
Answer: 1) Apply the Definition of Conditional Probability:
$p(A\|B) = p(A \cap B)/p(B) = 0.2/0.5 = 0.4$.
Multiple-principle Problem
Question: If $p(B) = 0.06$, $p(\sim A \cap \sim B) = 0.87$ and $p(A \cap B) = 0.03$, find $p(A)$.
Answer: 1) Apply the De Morgan's Law: $p(\sim (A \cup B)) = p(\sim A \cap \sim B) = 0.87$
2) Apply the Complement Theorem:
$p(A \cup B) = 1 - p(\sim (A \cup B)) = 1 - 0.87 = 0.13$
3) Apply the Addition Theorem for two events:
$p(A \cup B) = p(A) + p(B) - p(A \cap B)$, $p(A) = 0.13 + 0.03 - 0.06 = 0.1$.

problems in the same order. Each domain principle was applied at least twice in the 12 problems, and each of the problems required 20-50 steps to solve (See Appendix A for an example). Finally, all students took the 20-problem **post-test**: 14 of them were isomorphic to the pre-test, and the remainder were non-isomorphic multiple-principle problems. The experimental conditions differed only in decision granularity.

Grading Criteria: The pre- and post-test problems required students to derive an answer by writing and solving one or more equations. We used three scoring rubrics for grading: binary, partial credit, and one-point-per-principle. Under the binary rubric, a solution was worth 1 point if it was completely correct or 0 if not. Under the partial credit rubric, each problem score was defined by the proportion of correct principle applications evident in the solution. A student who correctly applied 4 of 5 possible principles would get a score of 0.8. The one-point-per-principle rubric in turn gave a point for each correct principle application. All of the tests were graded in a double-blind manner by a single experienced grader. The results presented below were based upon the partial-credit rubric, but the same results hold for the other two. For comparison purposes, all test scores were normalized to the range of $[0, 100]$.

2.2.5 Single- and Multiple-Principle Problems Based Competence Split

Motivated by the aptitude treatment interaction (ATI) theory that some instructional interventions can be more or less effective for particular students depending upon their specific abilities or knowledge [13, 53], we split students into multiple competence groups based on their pre-test performance. We first conducted a median split based on students' pre-test scores. A two-factor analysis on granularity and incoming competence showed that there was a marginally significant interaction effect (p = 0.077) that the impact of granularity differed for the High and Low groups (see Appendix B for the results).

Specifically, problem-level granularity may be less effective for High incoming competence students and step-level may be less effective for Low ones. This suggests that the impact of granularity may depend on students' competence level. Given the fact that our ITS mainly focuses on training students to solve multiple-principle problems, we think the impact of granularity may be more obvious if the split is done based on their ability to solve single- and multiple-principle problems. Thus, we then conducted a split based on students' performance on single- and multiple-principle problems and conducted a two-factor analysis. Results showed the same general trend as that found in the median split analysis but with stronger p-values (see Section 2.4.2).

Learning in STEM domains often involves acquiring two types of knowledge: declarative knowledge, which includes facts that we know, such as domain principles and procedural knowledge, which specifies how to retrieve and use declarative knowledge to solve problems [1]. Procedural knowledge often requires the interplay of many cognitive factors including but not limited to the following five ones in order of occurrence: 1) acquisition of declarative knowledge, 2) identification and retrieval of the proper declarative knowledge, 3) application of declarative knowledge, 4) organization and production of solution plans, and 5) execution of solution plans and evaluation of answers. Here, we argue that solving a single-principle probability problem mainly involves factors 1-3, while solving a multiple-principle probability problem involves all five.

Table 1 presents a comparison of a single- and a multiple-principle problem. As the example shows, a single-principle problem can be solved by applying a single domain principle once. The main challenge in this process is to properly retrieve and apply the principle. A multiple-principle problem, on the other hand, requires students to logically apply a series of domain principles to construct a solution path. The major challenge here is to build the multi-step inference. Generally speaking, multiple-principle problems involve a more complicated problem solving procedure, and thus are harder than single-principle problems to solve. Prior works have also viewed single- and multiple-principle problems as different types of tasks [9,10]. In the following, we refer to students who have clearly shown the ability to solve all the single-principle problems and at least part of a multiple-principle problem as Multiple-principle capable or High competence students (pre $> 10/14$) and those who have not clearly shown such ability as Single-principle only or Low competence students (pre $\leq 10/14$). Due to limited group size, the two-factor analysis (granularity $\times$ incoming competence) is only performed in the post-hoc analysis.

2.3 Fall 2014 Empirical Study Results

In the Fall 2014 study, 266 students were randomly assigned to five conditions: 31 for WE[1], 58 for Prob-Only, 59 for Step-Only, 59 for Both, and 59 for PS.

[1] Fewer students were assigned to the WE condition, because another purpose of this study was to collect training data for inducing the HRL policy.

Table 2 Fall 2014 Empirical Study Results

Condition	Pre	Iso Post	Post	Adj Post	Time (hours)
WE(21)	70.5(17.1)	79.0(18.4)	68.7(19.7)	66.8(13.5)	.78(.59)
Prob(36)	67.8(18.2)	77.6(13.6)	66.6(17.1)	66.3(13.6)	1.55(.42)
Step(36)	65.1(14.5)	74.6(16.1)	62.3(15.9)	63.6(11.1)	1.87(.62)
Both(33)	69.8(19.1)	81.9(13.7)	70.5(16.2)	69.0(14.5)	1.86(.51)
PS(28)	63.9(17.5)	80.2(12.2)	68.4(14.5)	70.4(12.4)	2.46(.65)

Due to preparations for exams and the length of the experiment, 162 students completed the study. Eight students who performed perfectly in the pre-test or completed the study in groups were excluded from our subsequent statistical analysis. The remaining 154 students were distributed as follows: 21 for WE, 36 for Prob-Only, 36 for Step-Only, 33 for Both and 28 for PS. A Chi-square test showed that the participants' completion rate did not significantly differ by condition: $\chi^2(4) = 4.61, p = 0.329$.

Incoming Competence: A one-way ANOVA analysis on the pre-test showed that there was no significant difference among the five conditions: $F(4, 149) = 0.78$, $p = 0.539$, $\eta = 0.021$. This suggests that the five conditions were balanced in incoming competence. Table 2 shows the mean and standard deviation (SD) of students' learning performance and total training time results. From left to right, it shows the condition with the number of students in parentheses, pre-test (Pre), isomorphic post-test (Iso Post), full post-test (Post), adjusted post-test (Adj Post) (post test scores adjusted by pre-test scores based on a linear model generated by ANCOVA analysis), and the total training time on the ITS in hours (Time).

Isomorphic Post-test: To measure the improvement students obtained from ITS training, we compared their isomorphic post-test and pre-test scores. A repeated measures analysis using test type (pre-test vs. isomorphic post-test) as a factor and test score as the dependent measure showed a main effect for test type: $F(1, 149) = 87.28$, $p < 0.0001$, $\eta = 0.362$ in that students scored significantly higher in the isomorphic post-test than in the pre-test. More specifically, all five conditions scored significantly higher in the isomorphic post-test: $F(1, 20) = 8.09$, $p = 0.010$, $\eta = 0.288$ for WE; $F(1, 35) = 16.37$, $p = 0.0003$, $\eta = 0.319$ for Prob-Only; $F(1, 35) = 18.90$, $p = 0.0001$, $\eta = 0.351$ for Step-Only; $F(1, 32) = 15.18$, $p = 0.0005$, $\eta = 0.322$ for Both and $F(1, 27) = 34.55$, $p < 0.0001$, $\eta = 0.561$ for PS. This suggests that the basic practice and problems, domain exposure, and interactivity of our ITS effectively help students acquire knowledge, even when the decisions are made randomly yet reasonably.

Learning Performance: To comprehensively evaluate students' final performance, we analyzed their full post-test performance, which has has six additional multiple-principle problems. An ANCOVA analysis on the post-test

Table 3 Fall 2015-2017 Empirical Studies Results

Year	Condition	Pre	Iso Post	Post	Adj Post	Time (hours)
2015	Prob(38)	79.4(15.4)	86.7(17.8)	78.5(19.4)	79.0(17.0)	1.99(.75)
	Step(34)	81.3(13.0)	92.4(8.6)	86.4(12.1)	85.9(11.5)	2.26(.48)
2016	Prob(31)	73.6(12.9)	87.0(12.4)	76.9(15.3)	75.6(14.8)	1.73(.66)
	Step(35)	69.9(17.1)	81.2(19.5)	69.5(19.3)	70.7(13.6)	1.88(.44)
2017	Both(55)	74.5(16.8)	87.8(14.3)	78.9(17.8)	78.9(13.5)	1.81(.53)

using the pre-test score as a covariate showed no significant difference among the five conditions: $F(4, 148) = 1.32$, $p = 0.264$, $\eta = 0.021$.
Time on Task: A one-way ANOVA analysis showed a significant difference among the five conditions: $F(4, 149) = 29.17$, $p < 0.0001$, $\eta = 0.439$. Subsequent contrast analysis revealed that WE spent significantly less time than Prob-Only ($p < 0.0001$), Step-Only ($p < 0.0001$), Both ($p < 0.0001$), and PS ($p < 0.0001$); Prob-Only spent significantly less time than Step-Only ($p = 0.014$), Both ($p = 0.023$), and PS ($p < 0.0001$); Step-Only spent significantly less time than PS ($p < 0.0001$); and Both spent significantly less time than PS ($p < 0.0001$). To summarize, for time on task, we have WE < Prob-Only < Step-Only, Both < PS. This suggests that granularity can have an impact on students' time on task.

2.4 Post-hoc Analysis Results

2.4.1 Analysis on the Impact of Granularity

The post-hoc analysis included the data collected in the Fall 2014-2017 studies, and students were combined by their condition. In the Fall 2015 study, 94 students were randomly assigned to the Prob-Only ($N = 47$) and Step-Only ($N = 47$) conditions. 73 students completed the study and one student who performed perfectly in the pre-test was excluded for subsequent analysis. For the remaining students, 38 were in the Prob-Only condition and 34 were in the Step-Only condition. In Fall 2016, 81 students were randomly assigned to the Prob-Only ($N = 40$) and the Step-Only ($N = 41$) conditions, and 67 of them completed the study. One student who got a perfect pre-test score was excluded for statistical analysis. The remaining students were distributed as follows: $N = 31$ for Prob-Only and $N = 35$ for Step-Only. Finally, the Fall 2017 study had 70 students assigned to the Both condition. 57 students completed the study and two that performed perfectly in the pre-test or completed the study in groups were excluded. Since the 2015-2017 studies did not include the baseline WE-only and PS-only conditions, the three granularity conditions were much larger in group size than the two baseline conditions. Thus, the post-hoc analysis mainly focused on comparing the three types of granularity.

The final post-hoc analysis data set includes 298 students: $N = 105$ for Prob-Only, $N = 105$ for Step-Only, and $N = 88$ for Both. A Chi-square test showed that the participants' completion rate did not differ significantly by condition overall: $\chi^2(2) = 0.535, p = 0.765$. Table 3 shows the test scores and training time results for the 2015-2017 Fall studies.

Incoming Competence: A one-way ANOVA analysis on the pre-test score showed no significant difference among the three conditions: $F(2, 295) = 0.29$, $p = 0.749$, $\eta = 0.002$. This suggests that the three conditions were balanced in incoming competence across the four years.

Isomorphic Post-test: A repeated measures analysis using test type (pre-test vs. isomorphic post-test) as a factor and test score as the dependent measure showed that there was a main effect for test type: $F(1, 295) = 168.76$, $p < 0.0001$, $\eta = 0.362$ in that students scored significantly higher in the isomorphic post-test than in the pre-test. Similarly, for each of the three conditions, students scored significantly higher in the isomorphic post-test than in the pre-test: $F(1, 104) = 46.72$, $p < 0.0001$, $\eta = 0.310$ for Prob-Only; $F(1, 104) = 63.64$, $p < 0.0001$, $\eta = 0.380$ for Step-Only and $F(1, 87) = 60.17$, $p < 0.0001$, $\eta = 0.409$ for Both. The results confirmed that our tutor has been effective over the years.

Learning Performance: A One-way ANCOVA analysis on the post-test score using the pre-test score as a covariate showed no significant difference among the three conditions: $F(2, 294) = 1.02$, $p = 0.362$, $\eta = 0.004$.

Time on Task: A one-way ANOVA analysis showed a significant difference among the three conditions: $F(2, 295) = 4.78$, $p = 0.009$, $\eta = 0.031$. Subsequent contrast analysis revealed that Prob-Only and Both spent significantly less time than Step-Only: $t(295) = -3.00$, $p = 0.003$, $d = 0.40$ and $t(295) = -2.08$, $p = 0.039$, $d = 0.32$ respectively.

2.4.2 Analysis on Granularity and Incoming Competence

To investigate whether the impact of granularity differs for students with different incoming competence, we split students into High (Multiple-principle capable) and Low competence (Single-principle only) groups and performed a two-factor analysis (granularity $\times$ competence) on their learning performance and time on task.

As expected, the High group significantly outperformed the Low group in the pre-test: $F(1, 296) = 725.98$, $p < 0.0001$, $\eta = 0.710$. The first column in Table 4 shows the groups with the number of students in the parentheses. As we can see, there were relatively more students in the High groups than in the Low groups, but a Chi-square test showed that the size of the two groups did not differ significantly by conditions: $\chi^2(2) = 1.66, p = 0.436$. Fortunately, the principle-based split preserved the balance of students' incoming competence among the three conditions in that there was no significant difference on pre-test among the three High or the three Low groups. Based on this split, We then conducted a 3×2 analysis on the factors of granularity and incoming

Table 4 Post-hoc Analysis Results

Group	Pre	Iso Post	Post	Adj Post	Time (hours)
Prob$_H$(63)	84.5(7.6)	87.8(10.9)	78.4(14.7)	70.5(13.5)	1.83(.69)
Step$_H$(55)	85.3(7.2)	91.7(9.0)	83.7(11.7)	75.2(10.7)	2.15(.49)
Both$_H$(53)	84.9(6.6)	91.7(9.2)	84.1(13.4)	75.9(12.4)	1.79(.48)
Prob$_L$(42)	57.4(11.7)	77.5(19.0)	67.3(20.8)	77.7(17.0)	1.65(.57)
Step$_L$(50)	57.2(9.3)	72.6(18.2)	60.2(17.7)	70.8(15.5)	1.84(.56)
Both$_L$(35)	54.2(12.4)	76.3(15.7)	63.1(15.6)	75.6(16.8)	1.88(.57)

competence to investigate their impacts on student learning performance and time on task.

Learning Performance: Table 4 shows the test score and training time results. A two-way ANCOVA analysis for the Full post-test on the factors of granularity and incoming competence using the pre-test score as a co-variate showed a significant interaction effect: $F(2, 291) = 4.41$, $p = 0.013$, $\eta = 0.018$. But there was no significant main effect of granularity or incoming competence. Subsequent contrast analysis revealed that for High students, the Both$_H$ group scored significantly higher than the Prob$_H$ group: $t(291) = 2.07$, $p = 0.040$, $d = 0.42$; and there was a trend that Step$_H$ scored higher than Prob$_H$: $t(291) = 1.80$, $p = 0.073$, $d = 0.39$. The Step$_H$ and Both$_H$ groups performed similarly with no significant difference: $t(291) = 0.28$, $p = 0.780$, $d = 0.06$. For Low students, the Prob$_L$ group significantly outperformed the Step$_L$ group: $t(291) = 2.34$, $p = 0.020$, $d = 0.43$. Both$_L$ also scored 4.8 points higher than Step$_L$ in the adjusted post-test, but the difference was not significant: $t(291) = 1.60$, $p = 0.110$, $d = 0.30$. The Prob$_L$ and Both$_L$ groups scored similarly with no significant difference: $t(291) = -0.59$, $p = 0.558$, $d = 0.12$. The results generally suggest that problem-level granularity may be less effective for High students; step-level granularity may be less effective for Low students; and both-level can be effective for both High and Low students.

Time on Task: A two-way ANOVA analysis on granularity and incoming competence showed a marginally significant interaction effect: $F(2, 292) = 2.90$, $p = 0.056$, $\eta = 0.019$ and a significant main effect of granularity: $F(2, 292) = 4.91$, $p = 0.008$, $\eta = 0.031$ in that Prob-Only and Both spent less time than Step-Only: $t(295) = -3.00$, $p = 0.003$, $d = 0.40$ and $t(295) = -2.08$, $p = 0.039$, $d = 0.32$ respectively. Subsequent contrast analysis revealed that the difference mainly came from High students. Specifically, for High students: Step$_H$ spent significantly more time than Prob$_H$ and Both$_H$: $t(292) = 3.01$, $p = 0.003$, $d = 0.52$ and $t(292) = 3.25$, $p = 0.001$, $d = 0.74$ respectively; but there was no significant difference between Prob$_H$ and Both$_H$: $t(292) = 0.37$, $p = 0.710$, $d = 0.07$. For Low students, none of the contrasts was significant: $t(292) = -1.53$, $p = 0.126$, $d = 0.32$ for Prob$_L$ vs. Step$_L$; $t(292) = -1.73$, $p = 0.086$, $d = 0.39$ for Prob$_L$ vs. Both$_L$; and $t(292) = -0.33$, $p = 0.738$, $d = 0.07$ for Step$_L$ vs. Both$_L$.

2.5 Conclusions and Discussion for the Granularity Studies

In four studies, we explored the impact of decision granularity on student learning by comparing three types of granularity: 1) problem-level only (Prob-Only), 2) step-level only (Step-Only), and 3) both problem- and step-levels (Both) with two baseline conditions: WE-only (WE) and PS-only (PS). In the Fall 14 study, all five conditions were included, results showed that the granularity can have an impact on students' time on task in that: WE $<$ Prob-Only $<$ Step-Only, Both $<$ PS. However, there was no significant difference between the five conditions in terms of learning performance. In three follow-up studies, one or two granularity types were involved. We then combined students in all four studies by their conditions and conducted a post-hoc analysis focusing on the three granularity types. Results showed that there was still no significant difference among the three conditions in learning performance. Though, once we split students into different incoming competence groups, results generally suggest that Prob-Only may be less effective for High incoming competence students (Multiple-principle capable), Step-Only may be less effective for Low ones (Single-principle only), and Both can be effective for both High and Low students. For time on task, Prob-Only and Both spent significantly less time than Step. The results suggest that granularity indeed can have an impact on student learning, and its effects depend on students' current competence level.

The learning performance and time on task results suggest that Step-Only might be more difficult than Prob-Only. Our results showed that Prob-Only may be less effective for High competence students and Step-Only may be less effective for Low ones. Applying the Zone of Proximal Development (ZPD) theory [8], we infer that Step-Only is harder than Prob-Only. More specifically, the theory states that students learn best when the difficulty of the task lies in their ZPD – not too easy and not too hard. Based on this theory, we infer that Prob-Only lies in Low students' ZPD, and Step-Only lies in High students' ZPD. For time on task, results showed that the Prob-Only condition spent significantly less time than the Step-Only condition (in both the Fall 2014 study and post-hoc analysis). This suggests that more effort is needed for learning with Step-Only than with Prob-Only.

One possible cause that Step-Only is harder than Prob-Only could be that the former requires students to integrate information from two sources while the latter does not. To elaborate further upon this, when learning with Prob-Only, students pay attention to either the tutor's solution in WEs or their own solution in PS. However, for Step-Only, they need to pay attention to both the tutor's solution and their own solution to integrate them together, which may require extra organization work and even modification of their solution plan. However, this is only our hypothesis; more research is needed to investigate the exact mental work required by different granularity types.

Overall, the Granularity studies showed that decision granularity indeed can have an impact on student learning and the impact depends on students' competence level, which supports our hypothesis that WE, PS, and CPS are different types of learning activities. On the other hand, all the explorations of the impact of granularity were done with random pedagogical interventions that do not consider the dynamic of students' learning process. This raises an interesting question whether adaptively provide WE, PS and CPS would improve student learning performance. To investigate this question, we apply HRL to induced pedagogical policies that make pedagogical decisions at both the problem and step levels.

3 The HRL Study

To investigate the question of whether making two levels of adaptive decisions can result in effective instructional interventions, we propose and apply an HRL framework to induce a policy that makes decisions at both the problem and step levels. This section describes the background of the HRL study, our proposed method, and an empirical evaluation study where the HRL policy was compared with a DQN-induced step-level policy and a *random yet reasonable* step-level baseline policy.

3.1 Background – Applying RL to Improve the Effectiveness of ITSs

Generally speaking, RL approaches can be categorized into online and offline. Online approaches learn a policy in real time by interacting with the environment while offline approaches learn a policy from pre-collected training data. Based on the relationship between the behavior policy and the estimation policy, RL approaches can also be grouped into on-policy and off-policy. The behavior policy controls how the agent explores the environment, while the estimation policy is the one being learned. In on-policy approaches, these two policies are the same, while in off-policy approaches, they are different, even unrelated. Both online and offline approaches have been used for pedagogical policy induction in previous research, but most of them adopted an off-policy method. Thus, we will describe prior RL work from the online vs. offline perspective.

Prior research in online RL pedagogical policy induction has mainly relied on simulations or simulated students (computational learner models that imitate the learning process of students). One reason for that is online approaches often need large amounts of exploration to learn an effective policy, which is often too expensive to carry out with real students. Simulations, in contrast, provide relatively low-cost exploration opportunities as well as fast and economy policy evaluations [15]. On the other hand, the success of these approaches is heavily dependent on the accuracy of the simulations. Beck et al. [7] applied an online approach, temporal difference, with off-policy ϵ-greedy exploration to induce pedagogical policies that would minimize students' time on task. Simulated students were used for policy induction, while real students were involved in the empirical evaluation study. Results showed that the RL condition spent significantly less time on the training task than the control condition, and there was no significant difference between them on learning performance. In another study, Iglesias et al. applied another popular online off-policy approach, Q-learning, to induce policies for efficient learning [23]. Again, the policy was induced using simulations and evaluated with real students in a classroom study. Results showed that there was no significant difference between the RL and the control condition on learning performance, but the RL condition spent significantly less time on task. More recently, Rafferty et al. applied an online POMDP approach with off-policy tree search to induce policies for faster learning[36]. They first trained a simulation model based on pre-collected real student data and then induced RL policies based on the model. Empirical study results showed that the policies can accelerate learning as compared to baseline policies.

Offline RL approaches, on the other hand, "take advantage of previously collected samples and generally provide robust convergence guarantees" [46]. These approaches avoid the possible errors and bias generated by simulations, but the success of these approaches is often heavily dependent on the quality of the training data. One common convention for offline policy induction is to collect data by training students on an ITS that makes random yet *reasonable* decisions and then apply RL to induce the policy. Shen et al. applied an offline approach, value iteration, on a pre-collected training corpus to induce pedagogical policies aimed at improving students' learning performance [49]. Empirical classroom study results showed that the RL induced policy was more effective than the baseline policies for certain students. In another study, Shen et al. applied an offline POMDP approach to induce two types of policies, one aimed at improving students' learning performance and the other targeted at reducing the time students spend on learning [48]. Classroom study results showed that the learning-enhancing policy can improve the learning performance for certain students, and the time-reducing policy can reduce the training time for certain students. Similarly, Chi et al. applied policy iteration to induce a pedagogical policy aimed at improving students' learning gains [11]. The policy was evaluated in classroom studies, and results showed that the RL policy can significantly improve students' learning performance as compared to baseline policies. Finally, Mandel et al. applied an offline POMDP approach to induce a policy that aims to improve student performance in an educational game [28]. In an empirical study, the policy was compared with random and an expert policy. Results showed that the POMDP policy significantly outperformed the other two policies.

Overall, previous research suggests that RL is a promising approach for improving student learning and/or behavior in ITSs. However, using RL does not guarantee effective interventions [16], and the necessity for accurate simulations (online) or large training corpora (offline) has limited the wide use of RL for pedagogical policy induction. In addition, prior research on applying RL (both online and offline) for pedagogical policy induction has treated all system decisions *equally or independently* and has not taken decision granularity into account. In the remainder of this paper, we will refer to these approaches as flat RL to differentiate them from our new hierarchical RL approach.

A considerable amount of prior research has shown that hierarchical reinforcement learning (HRL) can be more effective and data-efficient than flat RL approaches [14,41,34,64,25]. HRL generally breaks down a large decision-making problem into a hierarchy of small sub-problems and induces a policy for each of them. Since the sub-problems are small, they usually require less data to find optimal policies. For example, Cuayáhuitl et al. applied HRL to induce navigation policies that make decisions at three levels of granularity: buildings, floors, and corridors. Experimental results showed that HRL converged to an optimal policy in much fewer iterations than flat RL. Similarly, Peng et al. applied HRL to induce locomotion control policies that make decisions at two levels of temporal granularity for path-following and soccer-dribbling [34]. Results showed that HRL policies can complete tasks that flat RL policies could not complete. While promising, the use of hierarchy requires additional information, such as the transitions and rewards at different levels of granularity, to induce a policy. A simple and effective way to collect this information is to explore the environment during learning. Thus, most existing HRL approaches have been online. In our work, given the great challenges for building accurate simulation models, we induce pedagogical policies in an offline way from pre-collected data, which prevents us from using online approaches. Therefore, we propose and apply an offline, off-policy HRL framework to induce the policy. This line of research represents the first attempts to apply HRL for pedagogical policy induction.

3.2 Methods – Policy Induction

3.2.1 Policy Induction Data and Challenges

Prior research applying RL to induce pedagogical policies often formalized student-system interactions as a Markov Decision Processes (MDP). The central idea behind RL approaches is to transform the problem of inducing effective policies into a computational problem of finding an optimal policy for choosing actions in an MDP. An MDP describes a stochastic control process using a 4-tuple $< S, A, T, R >$. In pedagogical policy induction, states S are often represented by vectors composed of relevant learning environment features, such as the percentage of correct attempts a student has made so far and so on. Actions A are the tutor's possible actions, such as elicit or tell. The reward function R is usually calculated from the system's success measures, such as students' learning performance. Once the $< S, A, R >$ has been defined, the transition probability function T is estimated from the training corpus.

In this work, the learning environment states are represented using 142 features extracted from system logs (See Appendix C for a list of the features). The features can be grouped into five categories:
Autonomy (10 features): the amount of work done by a student, such as the number of elicit steps completed *nElicit* or the number of elicits since the

last tell *nElicitSinceTell*. This category describes the amount of work the student has done recently or in a certain period of time.
Temporal (29): time related information about the student's behavior, such as the average time per step *avgStepTime*, or the total time on training so far *timeOnTutoring*. This category reflects the student's working speed or the amount of effort he/she has put on learning.
Problem Solving (35): information about the current problem solving context, such as problem difficulty *problemDifficulty*, or the number of principles needed to solve the problem *nPrincipleInProblem*. This category provides information about the current task the student is working on.
Performance (57): information about the student's performance so far, such as the percentage of correct principle applications *pctCorrectPrin*. This category reflects the student's current competence level.
Hints (11): information about the student's hint usage, such as the total number of hints requested *nHint*. This category describes the student's hint usage behavior.
The actions are WE, PS and CPS at the problem level and elicit and tell at the step level. Since our primary interest is to improve students' final learning, we used Normalized Learning Gain (NLG) as the reward because it measures students' gain *irrespective of their incoming competence*. $NLG = \frac{posttest - pretest}{\sqrt{1 - pretest}}$ [2] where *pretest* and *posttest* refer to the students' test scores before and after the ITS training respectively, and 1 is the maximum score.

Based on the definition of $< S, A, R >$, we transferred system logs into a dataset $\mathcal{D}$ which consists of trajectories d in the form of $s_1 \xrightarrow{a_1, r_1} s_2 \xrightarrow{a_2, r_2} \cdots s_n \xrightarrow{a_n, r_n}$. Here $s_i \xrightarrow{a_i, r_i} s_{i+1}$ indicates that at the i_{th} turn in d, the learning environment was in state s_i, the agent executed action a_i and received reward r_i, and then the learning environment transferred into state s_{i+1}. Given that a student's NLG will not be available until the entire training is completed, only terminal states have non-zero rewards. Thus for a trajectory d, $r_1 \cdots$, r_{n-1} are all equal to 0, and only the final reward r_n is equal to the student's $NLG \times 100$, which is in the range of (-∞, 100]. Applying this dataset for offline HRL policy induction, we face at least two challenges: one is that the rewards in ITSs are often much delayed, and the other is the uncertainty in our training data. We elaborate on these two challenges below.

In RL policy induction, immediate rewards are generally more effective than delayed rewards. This is because it is easier to assign appropriate credit or blame when the feedback is tied to a single decision. The more we delay the rewards or punishments, the harder it becomes to assign credit or blame properly. The availability of immediate rewards is especially important when the training data is limited, because it allows the agent to use the transition information more efficiently. Immediate rewards are available when the impact of each individual action can be observed and evaluated immediately. For ex-

[2] A square root was used in this definition to reduce the variance and the difference between different incoming competence groups, see Appendix D for a comparison of two NLG definitions.

ample, in an automatic call center system, the agent can receive an immediate reward for every question it asks because the impact of each question can be assessed instantaneously [66].

On the other hand, the most appropriate reward to use in ITS (student NLG) is typically unavailable until the entire training process is complete. This is due to the complex nature of the learning process, which makes it difficult to assess students' learning moment by moment, and more importantly, many instructional interventions that boost short-term performance may not be effective over the long-term. Besides policy effectiveness, another issue raised by delayed rewards lies in the induction of hierarchical policies. Since each training problem in our ITS covers different knowledge components, we induce an independent step-level sub-policy for each of them, and this requires us to assign a reward to each problem (or each step). To tackle this challenge, we apply a Gaussian Processes based (GP-based) approach [4] to infer "immediate rewards" from the delayed rewards.

For the second challenge, there are two sources of uncertainty in our data: non-determinism in the control process and imperfect observations of students' knowledge levels. Uncertainty arises in the control process because neither system tutorial actions nor students' knowledge levels deterministically influence learning outcomes. In addition, students' underlying knowledge levels are only indirectly and partially observed through system logs. For example, when the student correctly applied a domain principle, the logs do not say whether the student made it by guess or solid inference. Thus, our system logs can be seen as *imperfect* observations of students' learning states.

In recent years, a lot of approaches have been proposed for principled handling of uncertainty for modeling in environments that are dynamic, noisy/uncertain, observation-costly, and time-sensitive. Among them, GP has been shown to be a robust, stable, computationally tractable and principled approach that naturally accommodates these real-world challenges [37]. GP handles uncertainty by using a probabilistic model (a mean and a kernel function) to represent the target function, which limits the impact of dirty data points. In function approximation, it starts with a prior mean and kernel function, which specifies the similarity between data points. Then, it incorporates new information into the prior model using Bayesian inference to generate a posterior estimation. This process allows us to achieve a good posterior estimation with just relatively few data points. When applying GP to RL, it recursively estimates the Q-value function following the Bellman equation until convergence.

Figure 2 shows an overview of the policy induction procedure. From the raw logs, we first extract student-system interaction trajectories with delayed rewards; then immediate rewards are inferred using our Gaussian Processes (GP) based approach; finally, the trajectories with inferred immediate rewards are used for both HRL and DQN to induce pedagogical policies.

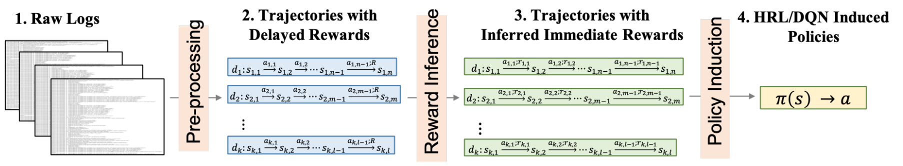

1. Raw Logs
Pre-processing
2. Trajectories with Delayed Rewards
$d_1: s_{1,1} \xrightarrow{a_{1,1}} s_{1,2} \xrightarrow{a_{1,2}} \cdots s_{1,n-1} \xrightarrow{a_{1,n-1};R} s_{1,n}$
$d_2: s_{2,1} \xrightarrow{a_{2,1}} s_{2,2} \xrightarrow{a_{2,2}} \cdots s_{2,m-1} \xrightarrow{a_{2,m-1};R} s_{2,m}$
$d_k: s_{k,1} \xrightarrow{a_{k,1}} s_{k,2} \xrightarrow{a_{k,2}} \cdots s_{k,l-1} \xrightarrow{a_{k,l-1};R} s_{k,l}$
Reward Inference
3. Trajectories with Inferred Immediate Rewards
$d_1: s_{1,1} \xrightarrow{a_{1,1};r_{1,1}} s_{1,2} \xrightarrow{a_{1,2};r_{1,2}} \cdots s_{1,n-1} \xrightarrow{a_{1,n-1};r_{1,n-1}} s_{1,n}$
$d_2: s_{2,1} \xrightarrow{a_{2,1};r_{2,1}} s_{2,2} \xrightarrow{a_{2,2};r_{2,2}} \cdots s_{2,m-1} \xrightarrow{a_{2,m-1};r_{2,m-1}} s_{2,m}$
$d_k: s_{k,1} \xrightarrow{a_{k,1};r_{k,1}} s_{k,2} \xrightarrow{a_{k,2};r_{k,2}} \cdots s_{k,l-1} \xrightarrow{a_{k,l-1};r_{k,l-1}} s_{k,l}$
Policy Induction
4. HRL/DQN Induced Policies
$\pi(s) \rightarrow a$

Fig. 2 Flow Chart of the HRL policy induction procedure

3.2.2 GP-Based Approach for Immediate Rewards Inference

The GP-Based immediate rewards inference approach takes state-action trajectories $s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \cdots s_n \xrightarrow{a_n}$ and the corresponding delayed rewards R as the input and outputs a inferred immediate reward r_i for each state-action pair (s_i, a_i). To infer the immediate rewards, we apply GP [37,5] to learn a reward function that distributes inferred immediate rewards inside each trajectory by assuming that they follow Gaussian distributions and that these rewards add up to the delayed reward for each trajectory [4]. The learning objective is to minimize the additive error of its output rather than the direct output error [20].

More specifically, in the context of GP, a function can be specified by a mean and a covariance function. In our case, we first assume a prior reward function based on our knowledge: $\mathbf{r} \sim \mathcal{N}(\mu_\mathbf{r}, \mathbf{C}_\mathbf{r})$ where $\mu_\mathbf{r}$ is the prior mean and $\mathbf{C}_\mathbf{r}$ is the prior covariance. Then, we take the delayed rewards as observations to estimate a posterior reward function: $(\mathbf{r}|\mathbf{R}) \sim \mathcal{N}(\mathbb{E}[\mathbf{r}|\mathbf{R}], \mathbb{C}[\mathbf{r}|\mathbf{R}])$. The conditional mean and covariance of the posterior function are calculated using the theorem of conditional probability density functions for multivariate Gaussian [17], as shown in the following two equations [4]:

$$\mathbb{E}[\mathbf{r}|\mathbf{R}] = \mathbb{E}[\mathbf{r}] + \mathbb{C}_{\mathbf{rR}}\mathbb{C}_{\mathbf{RR}}^{-1}(\mathbf{R} - \mathbb{E}[\mathbf{R}]) \tag{1}$$

$$\mathbb{C}[\mathbf{r}|\mathbf{R}] = \mathbb{C}_{\mathbf{rr}} - \mathbb{C}_{\mathbf{rR}}\mathbb{C}_{\mathbf{RR}}^{-1}\mathbb{C}_{\mathbf{Rr}} \tag{2}$$

In these equations, $\mathbb{E}[\mathbf{r}]$ is the mean of the prior function; $\mathbb{C}_{\mathbf{rR}}$ and $\mathbb{C}_{\mathbf{Rr}}$ are the cross-covariance between the delayed rewards and the immediate rewards; $\mathbb{C}_{\mathbf{RR}}$ is the covariance matrix of the delayed rewards; $\mathbf{R}$ is the observed delayed rewards; $\mathbb{E}[\mathbf{R}]$ is the mean of the observed delayed rewards and $\mathbb{C}_{\mathbf{rr}}$ is the covariance matrix of the immediate rewards. Here, $\mathbb{E}[\mathbf{r}] = \mu_\mathbf{r}$ and $\mathbb{C}_{\mathbf{rr}} = \mathbf{C}_\mathbf{r}$ are determined by the prior reward function, and $\mathbb{C}_{\mathbf{RR}}$ and $\mathbb{C}_{\mathbf{rR}}$ are estimated using our data based on the definition of covariance and cross-variance respectively [18]. Next, we will describe how $\mathbb{C}_{\mathbf{RR}}$ and $\mathbb{C}_{\mathbf{rR}}$ are calculated with the assumption that the sum of the immediate rewards equals to the delayed reward.

For each individual trajectory, we have $R = \sum_{i=0}^{n-1} \gamma^i r_i + \varepsilon$ where R is the delayed reward for the trajectory, r_i is the immediate reward in the ith term,

n is the length of the trajectory, $\gamma \in [0,1]$ is a discount factor, and ε is a white noise. Here, we assume that the white noise follows an Independent, Identically Distributed Gaussian distribution with zero mean and variance $\varepsilon \sim \mathcal{N}\left(0, \sigma_{\mathbf{R}}^2\right)$, where $\sigma_{\mathbf{R}}^2$ is the variance of the delayed rewards. Placing all the trajectories together, we can represent the summation relationship between immediate and delayed rewards using matrices: $\mathbf{R} = \mathbf{D}\mathbf{r} + \boldsymbol{\varepsilon}$ where $\mathbf{R} \in \mathbb{R}^m$ with m being the number of trajectories in our dataset; $\mathbf{r} \in \mathbb{R}^l$ with $l = \sum_{i=1}^m n_i$, where n_i is the length of trajectory i; $\boldsymbol{\varepsilon} \in \mathbb{R}^m$ is the white noise and $\mathbf{D} \in \mathbb{R}^{m \times l}$ is the reward transformation matrix in the following form:

$$\mathbf{D} = \begin{bmatrix} \overbrace{1\ \gamma\ \gamma^2\ \dots}^{n_1} & \overbrace{0\ \ \dots}^{n_2} & & 0 \\ 0\ \ \dots\ 0 & 1\ \gamma\ \dots & 0\ \dots & 0 \\ 0 & \dots & \dots & \ddots \end{bmatrix} \tag{3}$$

Following the relationship between immediate and delayed reards, we have $\mathbb{E}[\mathbf{R}] = \mathbf{D}\mathbb{E}[\mathbf{r}] + \mathbb{E}[\boldsymbol{\varepsilon}] = \mathbf{D}\mu_{\mathbf{r}}$. Based on the definition of covariance for $\mathbb{C}_{\mathbf{RR}}$ and the definition of cross-variance for $\mathbb{C}_{\mathbf{rR}}$, assuming the independence between $\mathbf{r}$ and $\boldsymbol{\varepsilon}$, and because $\mathbb{E}[\boldsymbol{r\varepsilon}] = 0$ and $\mathbb{E}[\mu_{\mathbf{r}}\boldsymbol{\varepsilon}] = 0$, we have the following two equations:

$$\begin{aligned} \mathbb{C}_{\mathbf{RR}} &= \mathbb{E}[(\mathbf{R} - \mathbb{E}[\mathbf{R}])(\mathbf{R} - \mathbb{E}[\mathbf{R}])^{\mathrm{T}}] \\ &= \mathbb{E}\left[(\mathbf{Dr} + \boldsymbol{\varepsilon} - \mathbf{D}\mu_{\mathbf{r}})(\mathbf{Dr} + \boldsymbol{\varepsilon} - \mathbf{D}\mu_{\mathbf{r}})^{\mathrm{T}}\right] \\ &= \mathbf{D}\mathbf{C}_{\mathbf{r}}\mathbf{D}^{\mathrm{T}} + \sigma_{\mathbf{R}}^2\mathbf{I}. \end{aligned} \tag{4}$$

$$\begin{aligned} \mathbb{C}_{\mathbf{rR}} &= \mathbb{E}[(\mathbf{r} - \mathbb{E}[\mathbf{r}])(\mathbf{R} - \mathbb{E}[\mathbf{R}])^{\mathrm{T}}] \\ &= \mathbb{E}\left[(\mathbf{r} - \mu_{\mathbf{r}})(\mathbf{Dr} + \boldsymbol{\varepsilon} - \mathbf{D}\mu_{\mathbf{r}})^{\mathrm{T}}\right] = \mathbf{C}_{\mathbf{r}}\mathbf{D}^{\mathrm{T}}. \end{aligned} \tag{5}$$

Plugging equations 4 and 5 into equations 1 and 2, we can calculate the posterior mean and covariance for the reward function.

3.2.3 An Offline, Off-policy GP-based HRL for Policy Induction

Most HRL research is based upon an extension of MDPs called discrete Semi-Markov Decision Processes (SMDPs), which add a set of complex activities [6] or options [55] to the primitive action set. The complex activities can invoke other activities recursively, thus allowing the hierarchical policy to function. The *complex* activities are distinct from the primitive actions in that a complex activity may contain multiple *primitive* actions. A complex activity consists of three elements: a policy π that maps *states to each available option*, a termination condition, and an initiation set. A solution to an SMDP is an optimal policy (π^*), a mapping from *state to complex activities or primitive actions*, that maximizes the expected discounted cumulative rewards for each state. In our case, WE, PS, and CPS are complex activities; while elicit and tell are primitive actions.

Since the complex activities in SMDPs can take a variable number of low-level activity (or actions) to execute across multiple time steps, it is necessary to extend the state-transition function to take into account the activity length. If an activity a in state s takes t' time steps to be executed, then the state transition probability function given s and a is defined by the joint distribution of the result state s' and the number of time steps t' the activity a takes: $P(s', t'|s, a)$. Accordingly, the expected reward function needs to be extended to accumulate over the waiting time t' in s given activity a: $R(s, a, t', s')$. Similar to RL, HRL learns a policy through estimating the Q-value function $Q(s, a)$, which denotes the expected cumulative rewards the agent will receive if it takes action a in state s and follows the policy to the end. The optimal Q-value function Q^* denotes the expected cumulative rewards the agent can receive if it follows the optimal policy and Q^* satisfies the Bellman equation[55]. In SMDPs, the Bellman equation can be rewritten as:

$$Q^*(s,a) = R(s,a) + \sum_{s',t'} \gamma^{t'} P(s',t'|s,a) \max_{a' \in A} Q(s',a'), \tag{6}$$

where $0 \leq \gamma \leq 1$ is a discount factor. Once Q^* is calculated, the optimal policy can be easily determined by simply taking the action a with the highest Q value in state s. For HRL, learning occurs at multiple levels. Global learning generates a policy for the top-level decisions and local learning generates a policy for each complex activity. This process retains the fundamental assumptions of RL: that goals are defined by their association with rewards, and thus the objective is to discover actions that maximize the long-term cumulative reward. Local learning focuses not on learning the best policy for the overall task but the best policy for the task defined by the complex activity using local rewards.

In our offline off-policy HRL framework, both problem- and step-level policies were learned by recursively using the standard Gaussian Processes to estimate the Q-value function in equation 6. The algorithm is shown in Algorithm 1 (See Appendix E for computational details of using GP for Q-function approximation). To induce the hierarchical policy, we define a problem-level semi-MDP for determining whether the next problem should be WE, PS, or CPS and for each of the training problems, we define a step-level semi-MDP for inducing a step-level policy to determine elicit vs. tell if a complex activity CPS is selected for that training problem. Inferred immediate rewards are used for all semi-MDPs. All 142 features are used to represent the state. To equalize the impact of each feature, all features are normalized to the range of [0,1].

Our training corpus was collected from training students on an ITS using random (yet reasonable) pedagogical decisions at the problem and step levels. Part of the data were collected from the Granularity studies and the rest were collected through other data collection efforts. All data were collected in the same class using the same material and following the same general

Algorithm 1 Learning Algorithm for GP based HRL

```
1: # Input: a dataset $\mathcal{D}$ consisting of trajectories in the form of $s_1 \xrightarrow{a_1,r_1} s_2 \xrightarrow{a_2,r_2} \cdots s_n \xrightarrow{a_n,r_n}$
2: # Denote the collection of state-action pairs $(s_i, a_i)$ appeared in $\mathcal{D}$ as $(\widetilde{S}, \widetilde{A})$
3: Give each $(s_i, a_i)$ in $(\widetilde{S}, \widetilde{A})$ an initial Q value $\dot{Q}(s_i, a_i)$; denote the resulting $(s_i, a_i; \dot{Q}(s_i, a_i))$ collection as $(\widetilde{S}, \widetilde{A}; \widetilde{Q})$
4: Converged ← False
5: Policy decision queue (with capacity m) $QUEUE \leftarrow \emptyset$
6: while not Converged do
7:     Fit a Q value estimation model $GP(s, a) = q$ based on $(\widetilde{S}, \widetilde{A}; \widetilde{Q})$
8:     Generate a policy $\pi$ based on $GP(s, a)$ where $\pi(s) = \text{argmax}_a Q(s, a)$
9:     if QUEUE.length == m then
10:        QUEUE.popFirst()
11:    Add $\pi$'s decisions on $\widetilde{S}$ to $QUEUE$
12:    if $QUEUE$ is full and all elements are identical then
13:        Converged ← True
14:    Update $(\widetilde{S}, \widetilde{A}; \widetilde{Q})$ following the bellman equation and using the current GP model, setting $\dot{Q}(s_i, a_i) = r(s_i, a_i) + \gamma \max_{a' \in A} GP(s_{i+1}, a')$
15: return $\pi$
```

procedure as the Granularity studies. Overall, the training corpus contains 1,118 students' interaction logs. Each student spent around 2 hours on the ITS and completed around 400 steps (since students received the same problems, they all completed around 400 steps).

3.2.4 DQN for Policy Induction

The RL methods we have used for pedagogical policy induction can be roughly divided into classic RL vs. Deep RL approaches. In classic RL approaches, the Q-value function is represented using tables or non-neural-network models (including GP) while in Deep RL approaches, it is represented using (deep) neural network. In one of our previous RL studies (mentioned in Section 1.1), we applied a classic RL approach to induce pedagogical policies, but the RL induced policies did not significantly outperform random baselines. On the other hand, in recent years, Deep RL has achieve superhuman performance in many complex tasks, including Atari games [32], Go [51], Chess/Shogi [52], Starcraft II [63], and robotic control [3]. Deep RL combines deep learning (neural networks) and novel reinforcement learning algorithms to handle decision-making problems. The use of neural networks allows it to handle large and complex environments, such as the screen of Atari games and the board of Go. Given that our learning environment is also large and complex (represented by 142 features), we apply Deep RL to induce the step-level policy. Note that we expect Deep RL to be more effective than classic RL, and thus we chose a weak classic RL approach to test the effectiveness of HRL. If our classic HRL can be more effective than DQN, we expect that a Deep HRL approach would also be.

Prior research has proposed many Deep RL algorithms, including value-based methods such as DQN [32], Double DQN [60] or Dueling DQN [65];

policy-based methods such as Trust Region Policy Optimization [44] or Proximal Policy Optimization [45]; and Actor-Critic methods such as Deep Deterministic Policy Gradients [26] or Soft Actor-Critic [21]. Each of these methods has its own advantages and drawbacks. In this work, we use the offline Double-DQN algorithm [60] with prioritized experience replay [43]. More specifically, it uses a multi-layer perceptron neural network to approximate the Q-function. The inputs to the neural network are the last 3 step observations of a student and the outputs are the Q values for each possible step-level action (in our case, elicit and tell). The network consists of two 64-unit layers with the rectified linear unit (ReLU) activation function (except that the output layer has no activation function). As a convention for this algorithm, an experience replay buffer and a target network are used to stabilize the training. The data and immediate rewards used for DQN policy induction are identical to those used for HRL.

3.3 Empirical Experiment

Similar to the Granularity studies, participants in the study were undergraduate students from the discrete math class where the study was given as one of the regular homework assignments. 180 Students were randomly assigned into the HRL, DQN, and Baseline conditions (60 for each). Due to preparations for exams and the length of the experiment, 140 students completed the study. 3 students who scored perfectly in the pre-test were excluded from our subsequent analysis. In addition, 9 students who completed the study in groups were excluded. The remaining 128 students were distributed as follows: $N = 44$ for HRL, $N = 45$ for DQN, and $N = 39$ for Baseline. A Chi-square test showed that the participants' completion rate did not differ by condition: $\chi^2(2) = 1.03, p = 0.598$. The study followed exactly the same 4-phase procedure as the Granularity studies, covered the same content and used the same ITS.

Incoming Competence: Despite random assignment, a one-way ANOVA analysis on the pre-test score showed a marginally significant difference among the three conditions: $F(2, 125) = 2.805$, $p = 0.064$, $\eta = 0.043$. Subsequent contrast analysis showed that DQN scored significantly higher than HRL on the pre-test: $t(125) = 2.06$, $p = 0.042$, $d = 0.46$ and Baseline: $t(125) = 2.01$, $p = 0.046$, $d = 0.46$; but there was no significant difference between HRL and Baseline: $t(125) = 0.02$, $p = 0.986$, $d = 0.00$. The results suggest that while our random assignment indeed balanced students' incoming competence between the HRL and Baseline conditions, it did not do so for the DQN condition. Therefore, we mainly focus on comparing learning performances that consider the pre-test differences, that is, the adjusted post-test, and NLG especially the latter because it is the reward we used for policy induction.

Table 5 shows the mean and standard deviation (SD) of students' learning performance and total training time results across three conditions. From left to right, it shows the conditions with the number of students in the

Table 5 HRL Study Results

	Improvement			Learning Performance			
Condition	Pre	Iso Post	Iso NLG	Full Post	Adj Post	NLG	Time (h)
HRL(44)	66.4(18.8)	85.8(14.6)	33.3(15.4)	75.3(16.9)	77.1(10.3)	14.3(19.2)	2.19(.64)
DQN(45)	73.9(13.6)	85.2(13.1)	21.0(24.8)	74.2(14.6)	70.8(12.0)	-2.2(29.4)	1.81(.58)
Random(39)	66.3(18.9)	80.5(19.5)	23.6(23.1)	69.0(19.6)	70.8(13.8)	-0.1(35.0)	1.97(.52)

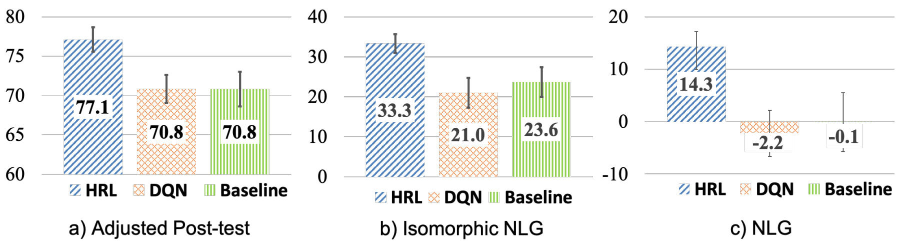


Fig. 3 Comparisons of learning performance among the HRL, DQN and Baseline conditions

parentheses, pre-test (Pre), isomorphic post-test (Iso Post), isomorphic NLG (Iso NLG), full post-test (Full Post), adjusted post-test (Adj Post), full NLG (NLG), and Total training time in hours (Time).

Improvement from training: *Isomorphic Post-test:* To measure the improvement students made through training, we compared their isomorphic post-test scores with their pre-test scores. A repeated measures analysis using test type (pre-test vs. isomorphic post-test) as a factor and test score as the dependent measure showed a main effect for test type: $F(1, 127) = 158.63$, $p < 0.0001$, $\eta = 0.555$ in that students scored significantly higher in the isomorphic post-test than in the pre-test. More specifically, all three conditions scored significantly higher in the isomorphic post-test than in the pre-test: $F(1, 43) = 110.74$, $p < 0.0001$, $\eta = 0.720$ for HRL, $F(1, 44) = 34.73$, $p < 0.0001$, $\eta = 0.441$ for DQN, and $F(1, 38) = 38.47$, $p < 0.0001$, $\eta = 0.503$ for Baseline. This suggests one more time that the ITS training is effective.

Isomorphic NLG: Based on the pre- and isomorphic post-test, we calculated the isomorphic NLG. Results showed that all three conditions had a great learning gain with the lowest one scoring 21.0, as shown in Figure 3.b. This suggests that our ITS indeed helps student learn. A one-way ANOVA analysis on the isomorphic NLG showed that there was a significant difference among the three conditions: $F(2, 125) = 4.04$, $p = 0.020$, $\eta = 0.061$. Subsequent contrast analysis showed that the HRL condition scored significantly higher than the DQN condition: $t(125) = 2.72$, $p = 0.008$, $d = 0.60$ and the Baseline condition: $t(125) = 2.06$, $p = 0.042$, $d = 0.50$. The difference between DQN and Baseline was not significant. The results suggest that the HRL policy is significantly more effective than the DQN policy and the Baseline policy.

Learning Performance: *Adjusted Post-test:* To comprehensively evaluate students' learning performance, we conducted analyses on their adjusted post-test score and NLG, which are calculated based on the pre- and full post-test. An ANCOVA analysis on the full post-test using the pre-test score as a covariate showed a significant difference among the three conditions: $F(2, 124) = 3.86$, $p = 0.024$, $\eta = 0.030$. Figure 3.a shows a comparison of the adjusted post-test scores among the three conditions, which are calculated by adjusting the full-post test scores using the pre-test scores based on a linear model generated by ANCOVA analysis. Contrast analysis on the adjusted post-test scores showed that the HRL condition scored significantly higher than the DQN condition: $t(125) = 2.47$, $p = 0.015$, $d = 0.56$ and the Baseline condition: $t(125) = 2.37$, $p = 0.020$, $d = 0.52$. No significant difference was found between DQN and Baseline.

NLG: Similarly, a one-way ANOVA analysis showed that there was a significant difference among the three conditions: $F(2, 125) = 4.39$, $p = 0.014$, $\eta = 0.066$, as shown in Figure 3.c. Contrast analysis revealed that HRL scored significantly higher than DQN: $t(125) = 2.75$, $p = 0.007$, $d = 0.66$ and Baseline: $t(125) = 2.30$, $p = 0.023$, $d = 0.52$. Again, no significant difference was found between DQN and Baseline. Overall, the results suggest again that the HRL policy significantly outperformed the DQN and the Baseline policy.

Time on Task: A one-way ANOVA analysis showed a significant difference among the three conditions: $F(2, 125) = 4.74$, $p = 0.010$, $\eta = 0.071$. More specifically, the HRL condition ($M = 2.19$, $SD = .64$ in hours) spent significantly more time than the DQN condition ($M = 1.81$, $SD = .58$): $t(125) = 3.07$, $p = 0.003$, $d = 0.62$ and marginally significantly more time than the Baseline condition ($M = 1.97$, $SD = .52$): $t(125) = -1.75$, $p = 0.082$, $d = 0.39$.

Tutor Decisions: Note that during ITS training, two of the 12 problems were fixed to be PS and the policies made decisions on the remaining 10 problems, which are the focus of our analysis. Figure 4 shows the problem-level decisions each student in the HRL condition received. During training, students received 0 to 5 WEs ($M = 0.95$, $SD = 1.16$), 0 to 10 PSs ($M = 5.07$, $SD = 2.58$) and 0 to 9 CPSs ($M = 3.98$, $SD = 2.49$). The results suggest that at the problem level, the HRL policy is more likely to choose PS and CPS than WE. Among the 44 students in the HRL conditions, 40 distinct patterns of WE, PS, and CPS on the 10 problems were found. This suggests that the HRL policy can make personalized decisions.

Table 6 Step-Level Tutor Decisions

Condition	Elicit	Tell	Pct Tell
HRL	309.0(60.4)	88.7(66.1)	22.0(15.9)
DQN	205.8(51.6)	188.9(53.0)	47.8(13.0)
Baseline	200.5(15.9)	203.5(17.4)	50.4(2.48)

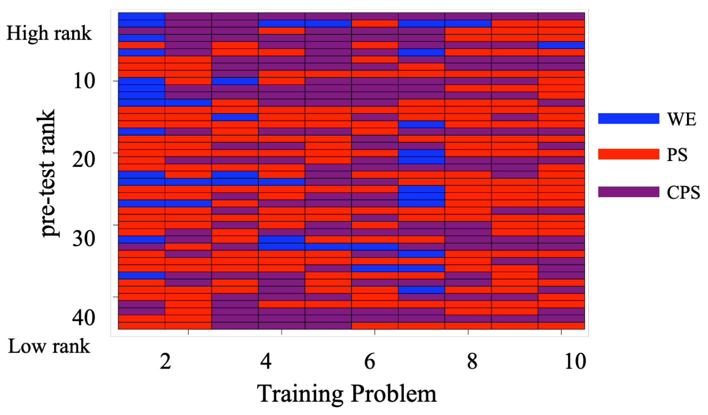


Fig. 4 Problem level decisions for the HRL policy, blue for WE, red for PS and purple for CPS. The X-axis shows the training problems in the order they appeared, and the Y-axis shows the students ordered by their pre-test scores, with the highest on the top.

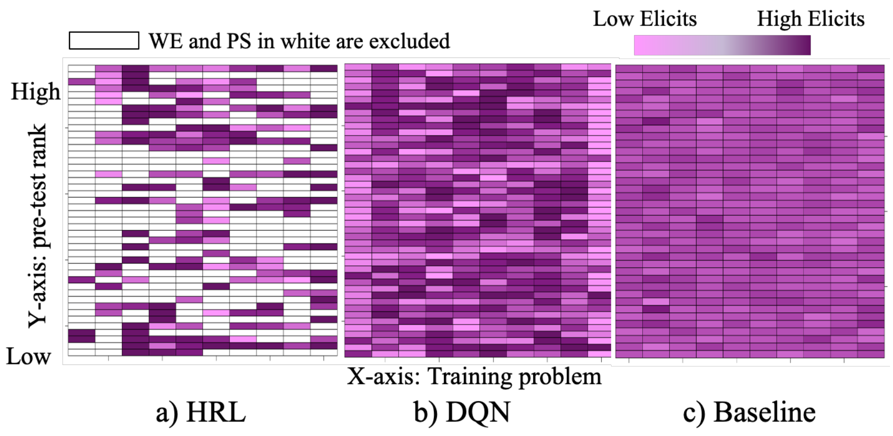


Fig. 5 Percentage of Elicit for the a. HRL, b. DQN and c. Random Baseline policies' step-level decisions on each problem. The higher the percentage, the darker a cell is. Problem-level decisions in HRL were set to white. The X-axis shows the training problems in the order they appeared, and the Y-axis shows the students ordered by their pre-test scores, with the highest on the top.

The step-level analysis focuses on the elicits and tells students received. To conduct an analysis across the three conditions, we transferred problem level WE and PS into step level tells and elicits. Table 6 shows the number of step-level decisions students received across the three conditions. Form left to right, it shows the condition, the number of elicits (Elicit) and tells (Tell), and finally the percentage of tells (Pct Tell). A one-way ANOVA analysis on the percentage of tells showed a significant difference among the three conditions: $F(2, 125) = 71.47$, $p < 0.0001$, $\eta = 0.533$. Subsequent contrast analysis showed that the HRL condition received significantly less tells than the DQN condition: $t(125) = -10.00$, $p < 0.0001$, $d = 1.78$ and the Baseline condition: $t(125) = -10.60$, $p < 0.0001$, $d = 2.42$. In addition, HRL and DQN had a much higher SD in tell percentage than Baseline. This suggests that the HRL and DQN policies made more personalized decisions than the Baseline policy.

To examine how the HRL and DQN policies tailored the step-level decisions for individual students, we calculated the percentage of elicit on each training problem each student received. Figure 5 shows the percentage of elicit pattern for the HRL, DQN, and Baseline conditions respectively. From the figures we can see that the HRL and DQN policies indeed made personalized decisions. The HRL policy gave more elicits than tells while DQN seems to have balanced them. Finally, as expected, the Baseline policy seems to have balanced the amount of elicits and tells in every problem. In some cases, a CPS can turn into a WE or PS. Among the 175 CPSs selected by the HRL policy, 45 (25.7%) turned into PSs and 3 (1.7%) turned into WEs. For the DQN policy, 31 (6.9%) out of 450 CPSs turned into PSs and 19 (4.2%) turned into WEs. However, the Baseline (random) policy did not turn any CPS into WE or PS.

4 Conclusions and General Discussion

In this work, we first investigated the impact of granularity on student learning and then explored taking decision granularity into account in data-driven pedagogical policy induction. In a series of classroom studies, we compared three types of granularity: problem-level only (Prob-Only), step-level only (Step-Only) and both the problem and step levels (Both). Overall, results showed that there was no significant difference among the three conditions on learning performance. However, a two-factor analysis on granularity and incoming competence showed that Prob-Only may be less effective for High students, Step-Only may be less effective for Low ones, and Both can be effective for both High and Low students. The results suggest that WE, PS, and CPS are different types of learning activities, and their effects depend on students' competence level. An implication of the results is that effective systems should adapt WE, PS, and CPS to students' learning state. To create such two-level adaptive interventions, we proposed and applied an offline, off-policy GP-based HRL framework to induce a hierarchical pedagogical policy that makes decisions first at the problem level and then at the step level. When there are decisions to make, it first decides whether the next problem should be a WE, PS, or CPS. If CPS is selected, a corresponding HRL induced step-level policy will be activated to decide whether to elicit or tell on each step. In an empirical classroom study, we compared the HRL policy with a DQN-induced step-level policy and a random step-level Baseline policy. Results showed that the HRL policy was significantly more effective than the DQN policy and the Baseline policy, and no significant difference was found between the latter two. For time on task, there was no significant difference between the HRL and Baseline conditions, but the former (HRL) spent significantly more time than the

DQN condition. Finally, log analysis suggests that both the HRL and DQN policies made personalized decisions and that the HRL policy was more likely to choose PS and CPS than WE.

Both the Granularity studies and the HRL study suggest that making Both-level decisions can be more effective then using a single level. This supports our intuition that decision granularity can have an impact on student learning. One potential explanation is that WE, PS, and CPS involve different learning mechanisms. Here, we argue that in WEs, students learn from observing; in PSs, students learn from doing; while in CPSs, they learn from co-constructing the solution with the tutor. For the Granularity studies, Both-level decisions allow students to experience all of them, which reduces the chance that the intervention is ineffective during the entire training. For the HRL study, explicitly selecting WE vs. PS vs. CPS at the problem level increases the likelihood that the most appropriate activity is chosen. Although step-level models can also construct WE and PS with tells and elicits, without explicit problem-level decisions, it is less likely to do so. However, this is only our hypothesis, and more research is needed to understand the exact learning mechanism behind WE, PS, and CPS.

The HRL study results are in line with the prior findings that adaptive decision-making can improve student learning [33,42]. Generally speaking, adaptive pedagogical policies can be more effective than fixed ones because the former can better consider students' learning dynamics. Fixed policies are usually designed based on assumptions of students' knowledge change during training. For example, the expertise reversal effect suggests providing learning activities in the WE-FWE-PS order [38,47]. That's based on the assumption that students' knowledge levels grow as the training proceeds. This is generally true, but the actual circumstance can be more complicated. Students may start the training with different incoming competence and learn at different rates. Thus, a pre-determined policy can hardly meet all students' learning needs perfectly and is generally not sensitive to unexpected situations. Adaptive pedagogical policies, on the other hand, make decisions based on real-time observations of students' learning states and thus are more likely to be effective. Prior research has also shown that adaptive policies can be more effective than fixed ones [33,42].

Although the decisions made according to the HRL policy are effective, the decisions cannot be well-explained by existing theories. For example, the cognitive load theory suggests that Low competence students should be given more WEs or tells, but we did not find such a trend in our analysis. The expertise reversal effect suggests that decisions should be made in the WE-FWE-PS order, but we found no trend showing this order. Results from the Granularity studies generally suggest that step-level decisions can be relatively more effective for High competence students, and problem-level decisions can be relatively more effective for Low ones. However, we found no implication that the HRL policy made decisions following this effect either. In fact, the HRL policy did not give more CPS to High incoming competence students nor select more CPS in the late phase of training. One possible reason for that is student learning is a dynamic process. Students who had relatively high competence at the beginning of training may not retain that level of competence after several problems and similarly, those who had relatively low competence in the early stage of training may not still have a lower competence in later stages. Additionally, the training problems were organized in increasing difficulty order. Students good at solving easy problems may not always be good at solving difficult ones. Given that our HRL policy may consider these dynamics in decision-making, it is not surprising that a complicated pattern was generated. However, more investigation is needed to understand why the HRL policy made decisions in that way.

Analysis of the HRL policy's decision-making suggests that it indeed made personalized decisions. Its selection of WE vs. PS vs. CPS varied a lot among individual students. Some students received all 10 training problems as PS, while some others received a maximum of 9 CPSs. For WE, while students in the HRL condition received less than one on average, some received as many as 5. This suggests that from the HRL policy's perspective, each of the activities can be effective for certain students but ineffective for some others, and thus individualized decision-making is desired in tutoring. In addition, the results also suggest that WE, PS, and CPS all can be the best action to take for the HRL policy, and thus support our hypothesis that they involve different learning mechanisms.

Our log analysis showed that the HRL policy sometimes elicits or tells all the steps in a problem. They may appear to be the same for students, but the system's underlying decision-making processes are different. When the tutor decides step-level adaptation is not needed for a problem, it selects WE or PS. Otherwise, it selects CPS. But in CPS, the step-level policy may decide to elicit or tell all the steps in the problem, turning it effectively (from the student perspective) into problem-level PS or WE. There are two possibilities for this situation. One is that students' learning state changed while completing the problem, and the step-level policy adapted the intervention accordingly for the change. The other one is that the problem- and step-level policies have inconsistent goals. Theoretically, the problem-level policy makes decisions to maximize students' learning over the whole training; while each step-level policy makes decisions to maximize the outcome for the current problem. It is possible that in some cases, CPS is the most effective intervention for long-term learning; while to elicit/tell all the steps is the best decision for the current problem. Thus, a CPS is turned into a WE or PS. In this sense, while the step-level policy can turn a CPS into a WE or PS, a problem-level policy is still needed to make sure that the intervention is effective over the long term. In other words, different levels of granularity contribute differently to student learning. However, this is only our hypothesis, more research is needed to understand why the HRL policy turned CPS into WE or PS.

From the machine learning perspective, our results are in line with prior findings that HRL can be more effective than flat RL given the same amount of data. In our work, the better performance of HRL may be explained by the use of the two-level learning structure. At the problem level, a problem is viewed as an atomic action. This abstraction aggregates the effects of all the steps in a problem, and thus may allow the HRL agent to better capture the effect and reward of taking a problem-level activity (WE vs. PS vs. CPS). Additionally, it converts a long step-level sequence into a short problem-level sequence, which may give the agent a better view of the long-term effects of each problem. At the step level, it still allows the HRL agent to fine tailor the instruction as flat RL approaches may do. In the policy induction procedure, the two-level learning structure may allow sequential information to propagate more efficiently over long trajectories and still retain the flexibility in tailoring the instruction. Theoretically, flat RL could learn the impact of a problem by aggregating step-level information, but there is no guarantee that it would.

As our data set contains only 1,118 students, insufficient data may be a reason that DQN did not generate an effective policy. Since each step-level trajectory includes hundreds of steps and there are noise and uncertainty in the student learning data, it may be difficult for the agent to estimate the long term impact of each action through directly propagating and accumulating step-level information. Additionally, the deep neural network used in DQN may require a large amount of data to reach a good convergence. In this sense, our work showed that HRL can be more effective than flat RL when a limited amount of data is available. It is still an open question whether the advantage of HRL still exists with large data sets. However, given the high cost for collecting student learning data, HRL can be a reasonable option for inducing adaptive pedagogical policies.

Due to limited group size, in the HRL study, we were not able to split students into different incoming competence groups for analysis. This leaves questions about the effectiveness of HRL across different students open. Would HRL policies benefit students at all competence levels? For which incoming competence group(s) are HRL policies more effective than flat RL policies? These are interesting questions for future research.

One limitation of our offline off-policy HRL framework is that it is computationally expensive. Theoretically, GP has $\mathcal{O}(n^3)$ time complexity and $\mathcal{O}(n^2)$ space complexity (in our case, n is the total number of problems/steps in the training trajectories). This makes it practically difficult to improve policy quality through using more data as well as apply our approach to tasks with large data sets. In the future, we will reduce the computational complexity of our HRL approach to make it suitable for large data sets. One possible solution towards this goal is to replace GP with methods that have lower computational complexity, such as neural networks. For example, we may build a neural network to infer the immediate rewards. The network takes state-action sequences as input, uses a hidden neuron to generate the inferred reward for each state-action unit in the sequence, and outputs the summation of the inferred immediate rewards. During training, we train the network to minimize the difference between the summation of the inferred rewards and the actual delayed rewards. Once the training is done, we can use the resulting model to infer rewards by taking the output value of the hidden neurons that generate immediate rewards. Similarly, we can use a regression neural network to replace the GP Q-value function approximator in the HRL algorithm.

Another limitation lies in the interpretability of the HRL and DQN policies. Both policies are represented by a complicated non-linear model with a lot of parameters, which performs a lot of computation to make a decision. The complex models make it very challenging to extract simple and human-understandable decision-making rules from those policies. As a result, it is hard to understand how the learning environment features are used by the models for effective decision-making. Moreover, since HRL and DQN are represented by different models with different structures, there is no existing way to directly compare their decision-making process.

The third limitation is we did not compare our data-driven HRL policy with a broader range of policies, such as the learning-theory-based policy designed by Najar et al. [33]. One challenge for this type of comparison is that theory-based policies often require additional information to function. For example, the policy designed by Najar et al. [33] requires measurements of students' cognitive state to make decisions. Since our training data is collected with a procedure that does not include such measurements, currently, we cannot directly compare an HRL policy with a learning-theory-based policy using the same procedure. Without such comparisons, questions about the benefits of making two-level decisions remain open. How effective HRL policies can be as compared to theory-based policies? In what circumstances is making two-level decisions more effective than one-level? We will investigate these questions in the future.

To summarize, this work investigated the impact of decision granularity on student learning and applied hierarchical reinforcement learning (HRL) to induce pedagogical policies. Results from the granularity studies showed that the impact of granularity depends on students' competence level in that problem level may be less effective for High incoming competence students, step level may be less effective for Low ones, and Both levels can be effective both High and Low students. The results suggest that adapting decision granularity to students' competence level may lead to better learning outcome than focusing on one level of granularity. To test this hypothesis, we proposed and applied an offline, off-policy Gaussian Processes based Hierarchical Reinforcement Learning (HRL) framework to induce a hierarchical pedagogical policy that makes decisions at both the problem and step levels. Results from an empirical classroom study showed that the HRL policy was significantly more effective than a DQN induced step-level policy and a random yet reasonable step-level policy. Our results confirm the intuition that HRL should outperform flat RL on pedagogical policy induction because it can simultaneously learn at two levels of granularity - the problem-level outer loop and the step-level inner loop.

5 Acknowledgements:

This research was supported by the NSF Grants: CAREER: Improving Adaptive Decision Making in Interactive Learning Environments(1651909), Integrated Data-driven Technologies for Individualized Instruction in STEM Learning Environments(1726550), Generalizing Data-Driven Technologies to Improve Individualized STEM Instruction by Intelligent Tutors (2013502), and Educational Data Mining for Individualized Instruction in STEM Learning Environments (1432156).

References

1. Anderson, J.R.: Problem solving and learning. American Psychologist **48**(1), 35 (1993)
2. Anderson, J.R., Corbett, A.T., Koedinger, K.R., Pelletier, R.: Cognitive tutors: Lessons learned. The journal of the learning sciences **4**(2), 167–207 (1995)
3. Andrychowicz, M., Baker, B., et al.: Learning dexterous in-hand manipulation. arXiv preprint arXiv:1808.00177 (2018)
4. Azizsoltani, H., Kim, Y.J., Ausin, M.S., Barnes, T., Chi, M.: Unobserved is not equal to non-existent: Using gaussian processes to infer immediate rewards across contexts. IJCAI pp. 1974–1980 (2019)
5. Azizsoltani, H., Sadeghi, E.: Adaptive sequential strategy for risk estimation of engineering systems using gaussian process regression active learning. Engineering Applications of Artificial Intelligence **74**(July), 146–165 (2018)
6. Barto, A.G., Mahadevan, S.: Recent advances in hierarchical reinforcement learning. Discrete event dynamic systems **13**(1-2), 41–77 (2003)
7. Beck, J., Woolf, B.P., Beal, C.R.: Advisor: A machine learning architecture for intelligent tutor construction. AAAI/IAAI **2000**(552-557), 1–2 (2000)
8. Chaiklin, S., et al.: The zone of proximal development in vygotsky's analysis of learning and instruction. Vygotsky's educational theory in cultural context **1**, 39–64 (2003)
9. Chi, M., Vanlehn, K.: Accelerated future learning via explicit instruction of a problem solving strategy. FRONTIERS IN ARTIFICIAL INTELLIGENCE AND APPLICATIONS **158**, 409 (2007)
10. Chi, M., VanLehn, K.: Meta-cognitive strategy instruction in intelligent tutoring systems: How, when, and why. Educational Technology & Society **13**(1), 25–39 (2010)
11. Chi, M., VanLehn, K., Litman, D., Jordan, P.: Empirically evaluating the application of reinforcement learning to the induction of effective and adaptive pedagogical strategies. User Modeling and User-Adapted Interaction **21**(1-2), 137–180 (2011)
12. Clement, B., Oudeyer, P.Y., Lopes, M.: A comparison of automatic teaching strategies for heterogeneous student populations. In: EDM 16-9th International Conference on Educational Data Mining (2016)
13. Cronbach, L.J., Snow, R.E.: Aptitudes and instructional methods: A handbook for research on interactions. Irvington (1977)
14. Cuayáhuitl, H., Dethlefs, N., Frommberger, L., Richter, K.F., Bateman, J.: Generating adaptive route instructions using hierarchical reinforcement learning. In: International Conference on Spatial Cognition, pp. 319–334. Springer (2010)
15. Doroudi, S., Aleven, V., Brunskill, E.: Robust evaluation matrix: Towards a more principled offline exploration of instructional policies. In: Proceedings of the fourth (2017) ACM conference on learning@ scale, pp. 3–12 (2017)
16. Doroudi, S., Aleven, V., Brunskill, E.: Where's the reward? International Journal of Artificial Intelligence in Education **29**(4), 568–620 (2019). DOI 10.1007/s40593-019-00187-x. URL https://doi.org/10.1007/s40593-019-00187-x
17. Eaton, M.L.: Multivariate statistics: a vector space approach. JOHN WILEY & SONS, INC., 605 THIRD AVE., NEW YORK, NY 10158, USA, 1983, 512 pp. 116–117 (1983)
18. Feller, W.: An introduction to probability theory and its applications, vol. 2. John Wiley & Sons (2008)

19. Goldberg, P.W., Williams, C.K., et al.: Regression with input-dependent noise: A gaussian process treatment. In: NIPS, pp. 493–499 (1998)
20. Guo, D., Shamai, S., Verdú, S.: Mutual information and minimum mean-square error in gaussian channels. IEEE Transactions on Information Theory **51**(4), 1261–1282 (2005)
21. Haarnoja, T., Zhou, A., et al.: Soft actor-critic algorithms and applications. arXiv:1812.05905 (2018)
22. Iglesias, A., Martínez, P., Aler, R., Fernández, F.: Learning teaching strategies in an adaptive and intelligent educational system through reinforcement learning. Applied Intelligence **31**(1), 89–106 (2009)
23. Iglesias, A., Martínez, P., Aler, R., Fernández, F.: Reinforcement learning of pedagogical policies in adaptive and intelligent educational systems. Knowledge-Based Systems **22**(4), 266–270 (2009)
24. Kalyuga, S., Renkl, A.: Expertise reversal effect and its instructional implications: Introduction to the special issue. Instructional Science **38**(3), 209–215 (2010)
25. Kulkarni, T.D., Narasimhan, K., Saeedi, A., Tenenbaum, J.: Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In: Advances in neural information processing systems, pp. 3675–3683 (2016)
26. Lillicrap, T.P., Hunt, J.J., et al.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)
27. Liz, B., Dreyfus, T., Mason, J., Tsamir, P., Watson, A., Zaslavsky, O.: Exemplification in mathematics education. In: Proceedings of the 30th Conference of the International Group for the Psychology of Mathematics Education, vol. 1, pp. 126–154. ERIC (2006)
28. Mandel, T., Liu, Y.E., Levine, S., Brunskill, E., Popovic, Z.: Offline policy evaluation across representations with applications to educational games. In: Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems, pp. 1077–1084. International Foundation for Autonomous Agents and Multiagent Systems (2014)
29. McLaren, B.M., van Gog, T., Ganoe, C., Yaron, D., Karabinos, M.: Exploring the assistance dilemma: Comparing instructional support in examples and problems. In: Intelligent Tutoring Systems, pp. 354–361. Springer (2014)
30. McLaren, B.M., Isotani, S.: When is it best to learn with all worked examples? In: International Conference on Artificial Intelligence in Education, pp. 222–229. Springer (2011)
31. McLaren, B.M., Lim, S.J., Koedinger, K.R.: When and how often should worked examples be given to students? new results and a summary of the current state of research. In: CogSci, pp. 2176–2181 (2008)
32. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529 (2015)
33. Najar, A.S., Mitrovic, A., McLaren, B.M.: Adaptive support versus alternating worked examples and tutored problems: Which leads to better learning? In: UMAP, pp. 171–182. Springer (2014)
34. Peng, X.B., Berseth, G., Yin, K., Van De Panne, M.: Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. ACM Transactions on Graphics (TOG) **36**(4), 41 (2017)
35. Phobun, P., Vicheanpanya, J.: Adaptive intelligent tutoring systems for e-learning systems. Procedia-Social and Behavioral Sciences **2**(2), 4064–4069 (2010)
36. Rafferty, A.N., Brunskill, E., Griffiths, T.L., Shafto, P.: Faster teaching via pomdp planning. Cognitive science **40**(6), 1290–1332 (2016)
37. Rasmussen, C.E.: Gaussian processes in machine learning. In: Advanced lectures on machine learning, pp. 63–71. Springer (2004)
38. Renkl, A., Atkinson, R.K., Maier, U.H., Staley, R.: From example study to problem solving: Smooth transitions help learning. The Journal of Experimental Education **70**(4), 293–315 (2002)
39. Rowe, J., Mott, B., Lester, J.: Optimizing player experience in interactive narrative planning: a modular reinforcement learning approach. In: Tenth Artificial Intelligence and Interactive Digital Entertainment Conference (2014)
40. Rowe, J.P., Lester, J.C.: Improving student problem solving in narrative-centered learning environments: A modular reinforcement learning framework. In: International Conference on Artificial Intelligence in Education, pp. 419–428. Springer (2015)

41. Ryan, M., Reid, M.: Learning to fly: An application of hierarchical reinforcement learning. In: In Proceedings of the 17th International Conference on Machine Learning. Citeseer (2000)
42. Salden, R.J., Aleven, V., Schwonke, R., Renkl, A.: The expertise reversal effect and worked examples in tutored problem solving. Instructional Science **38**(3), 289–307 (2010)
43. Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized experience replay. arXiv preprint arXiv:1511.05952 (2015)
44. Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P.: Trust region policy optimization. In: International conference on machine learning, pp. 1889–1897 (2015)
45. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
46. Schwab, D., Ray, S.: Offline reinforcement learning with task hierarchies. Machine Learning **106**(9-10), 1569–1598 (2017)
47. Schwonke, R., Renkl, A., Krieg, C., Wittwer, J., Aleven, V., Salden, R.: The worked-example effect: Not an artefact of lousy control conditions. Computers in Human Behavior **25**(2), 258–266 (2009)
48. Shen, S., Ausin, M.S., Mostafavi, B., Chi, M.: Improving learning & reducing time: A constrained action-based reinforcement learning approach. In: Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization, pp. 43–51. ACM (2018)
49. Shen, S., Chi, M.: Reinforcement learning: the sooner the better, or the later the better? In: Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization, pp. 37–44. ACM (2016)
50. Shih, B., Koedinger, K.R., Scheines, R.: A response time model for bottom-out hints as worked examples. Handbook of educational data mining pp. 201–212 (2011)
51. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al.: Mastering the game of go with deep neural networks and tree search. nature **529**(7587), 484 (2016)
52. Silver, D., Hubert, T., Schrittwieser, J., et al.: A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science **362**(6419), 1140–1144 (2018)
53. Snow, R.E.: Aptitude-treatment interaction as a framework for research on individual differences in psychotherapy. Journal of consulting and clinical psychology **59**(2), 205 (1991)
54. Stamper, J.C., Eagle, M., Barnes, T., Croy, M.: Experimental evaluation of automatic hint generation for a logic tutor. In: International Conference on Artificial Intelligence in Education, pp. 345–352. Springer (2011)
55. Sutton, R.S., Precup, D., Singh, S.: Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. Artificial intelligence **112**(1-2), 181–211 (1999)
56. Sweller, J., Cooper, G.A.: The use of worked examples as a substitute for problem solving in learning algebra. Cognition and Instruction **2**(1), 59–89 (1985)
57. Swetz, F.: To know and to teach: Mathematical pedagogy from a historical context. Educational Studies in Mathematics **29**(1), 73–88 (1995)
58. Swetz, F.J.: Capitalism and arithmetic: the new math of the 15th century, including the full text of the Treviso arithmetic of 1478, translated by David Eugene Smith. Open Court Publishing (1987)
59. Van Gog, T., Kester, L., Paas, F.: Effects of worked examples, example-problem, and problem-example pairs on novices' learning. Contemporary Educational Psychology **36**(3), 212–218 (2011)
60. Van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double q-learning. In: AAAI, vol. 2, p. 5. Phoenix, AZ (2016)
61. Vanlehn, K.: The behavior of tutoring systems. IJAIED **16**(3), 227–265 (2006)
62. VanLehn, K., Bhembe, D., Chi, M., Lynch, C., Schulze, K., Shelby, R., Taylor, L., Treacy, D., Weinstein, A., Wintersgill, M.: Implicit versus explicit learning of strategies in a non-procedural cognitive skill. In: International Conference on Intelligent Tutoring Systems, pp. 521–530. Springer (2004)

63. Vinyals, O., Babuschkin, I., Czarnecki, W., et al.: Grandmaster level in starcraft ii using multi-agent reinforcement learning. Nature **575**, 350 (2019)
64. Wang, X., Chen, W., Wu, J., Wang, Y.F., Yang Wang, W.: Video captioning via hierarchical reinforcement learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4213–4222 (2018)
65. Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., De Freitas, N.: Dueling network architectures for deep reinforcement learning. arXiv preprint arXiv:1511.06581 (2015)
66. Williams, J.D.: The best of both worlds: unifying conventional dialog systems and pomdps. In: INTERSPEECH, pp. 1173–1176 (2008)
67. Zhou, G., Azizsoltani, H., Ausin, M.S., Barnes, T., Chi, M.: Hierarchical reinforcement learning for pedagogical policy induction. In: International Conference on Artificial Intelligence in Education (2019)
68. Zhou, G., Chi, M.: The impact of decision agency & granularity on aptitude treatment interaction in tutoring. In: Proceedings of the 39th annual conference of the cognitive science society, pp. 3652–3657 (2017)
69. Zhou, G., Lynch, C., Price, T.W., Barnes, T., Chi, M.: The impact of granularity on the effectiveness of students' pedagogical decision. In: Proceedings of the 38th annual conference of the cognitive science society, pp. 2801–2806 (2016)
70. Zhou, G., Price, T.W., Lynch, C., Barnes, T., Chi, M.: The impact of granularity on worked examples and problem solving. In: Proceedings of the 37th annual conference of the cognitive science society, pp. 2817–2822 (2015)
71. Zhou, G., Wang, J., Lynch, C., Chi, M.: Towards closing the loop: Bridging machine-induced pedagogical policies to learning theories. In: EDM (2017)
72. Zhou, G., Yang, X., Azizsoltani, H., Barnes, T., Chi, M.: Improving student-tutor interaction through data-driven explanation of hierarchical reinforcement induced pedagogical policies. In: In proceedings of the 28th Conference on User Modeling, Adaptation and Personalization. ACM (2020)

Appendices

A An Example Training Problem

Table 7 shows an example of the training problem. Steps in the problem are packed as a series of main steps. In training, students need to first select the main step to work on and then carry it out. To reduce students' typing and calculation load, the tutor completes the specific "specify given" and "solve equation" procedure for them, as shown in the "Tutor" column.

Table 7 An Example Training Problem

Given event A, B and C with $p(A) = 0.8$, $p(B) = 0.6$, $p(C) = 0.9$ and A, B and C are independent events, find $p(\sim A \cap \sim B \cap \sim C)$.

Step	Student	Tutor
1	Select main step: specify given	A = 0.8
2	Select main step: specify sought	set $p(\ A \cap\ B \cap\ C)$ as sought
3	Select main step: specify given	B = 0.6
4	Select main step: specify given	C = 0.9
5	Select main step: apply principle	-
		set $\ A \cap\ B \cap\ C$ as target
6	Select principle: Independent Theorem	-
7	Apply principle (enter equation1):	
	$p(\sim A \cap \sim B \cap \sim C) = p(\sim A) * p(\sim B) * p(\sim C)$	-
8	Select main step: apply principle	-
9	Select target variable $\sim A$	-
10	Select principle: Complement Theorem	-
11	Apply principle: p$(\sim A) + p(A) = 1$ (eq2)	-
12	Select main step: apply principle	-
13	Select target variable $\sim B$	-
14	Select principle: Complement Theorem	-
15	Apply principle: p$(\sim B) + p(B) = 1$ (eq3)	-
16	Select main step: apply principle	-
		set $\sim C$ as target
17	Select principle: Complement Theorem	-
18	Apply principle: p$(\sim C) + p(C) = 1$ (eq4)	-
19	Select main step: solve equation	solve eq4
20	Select main step: solve equation	solve eq3
21	Select main step: solve equation	solve eq2
22	Select main step: solve equation	solve eq1
23	Select main step: quit	move to next problem

B Analysis on Granularity and Incoming Competence with Median Split

Table 8 Learning Performance and Time on Task Results with Median Split

Group	Pre	Iso Post	Post	Adj Post	Time (hours)
$\text{Prob}_H(53)$	86.4(6.9)	88.9(11.0)	80.3(14.6)	71.1(14.0)	1.86(.73)
$\text{Step}_H(47)$	87.3(5.8)	92.8(7.6)	85.2(10.8)	75.4(10.3)	2.14(.48)
$\text{Both}_H(47)$	86.3(5.6)	92.5(8.5)	85.4(13.0)	76.3(12.2)	1.77(.49)
$\text{Prob}_L(52)$	60.8(12.6)	78.3(17.5)	67.6(19.2)	75.7(16.4)	1.66(.54)
$\text{Step}_L(58)$	59.5(10.4)	74.3(18.1)	62.2(17.7)	71.2(15.1)	1.88(.57)
$\text{Both}_L(41)$	57.1(13.5)	77.6(15.5)	64.7(15.7)	75.3(16.3)	1.90(.54)

Table 8 shows the test score and training time results for the High and Low incoming competence groups (split by the median of pre-test). A two-way ANCOVA analysis for the Full post-test on the factors of granularity and incoming competence using the pre-test score as a covariate showed a marginally significant interaction effect: $F(2, 291) = 2.59$, $p = 0.077$, $\eta = 0.011$. But there was no significant main effect of granularity or incoming competence. Subsequent contrast analysis showed that for High students, there was a trend that the Both_H group scored higher than the Prob_H group: $t(291) = 1.81$, $p = 0.071$, $d = 0.39$. The Step_H group also scored 4.3 points higher than the Prob_H group, but the difference was not significant: $t(291) = -1.53$, $p = 0.127$, $d = 0.35$. The Step_H and Both_H groups scored similarly with no significant difference: $t(291) = 0.28$, $p = 0.782$, $d = 0.07$. For Low students, there was a trend that the Prob_L group scored higher than the Step_L group: $t(291) = 1.66$, $p = 0.098$, $d = 0.29$. Both_L also seems to score higher than Step_L, but the difference was not significant: $t(291) = 1.38$, $p = 0.168$, $d = 0.26$. Prob_L and Both_L scored similarly with no significant difference: $t(291) = -0.16$, $p = 0.870$, $d = 0.02$.

For time on task, a two-way ANOVA analysis on granularity and incoming competence showed a significant interaction effect: $F(2, 292) = 3.15$, $p = 0.044$, $\eta = 0.020$ and a significant main effect of granularity: $F(2, 292) = 4.89$, $p = 0.008$, $\eta = 0.031$ in that Prob-Only and Both spent less time than Step-Only: $t(295) = -3.00$, $p = 0.003$, $d = 0.40$ and $t(295) = -2.08$, $p = 0.039$, $d = 0.32$ respectively. Subsequent contrast analysis showed that for High students, Step_H spent significantly more time than Prob_H and Both_H: $t(292) = 2.47$, $p = 0.014$, $d = 0.45$ and $t(292) = 3.20$, $p = 0.002$, $d = 0.77$ respectively; but there was no significant difference between Prob_H and Both_H: $t(292) = -0.83$, $p = 0.406$, $d = 0.15$. For Low students, Step_L and Both_L spent significantly more time than Prob_L: $t(292) = 2.07$, $p = 0.039$, $d = 0.40$ and $t(292) = 2.00$, $p = 0.047$, $d = 0.44$ respectively; but there was no significant difference between Step_L and Both_L: $t(292) = 0.11$, $p = 0.915$, $d = 0.02$.

C Features Used for State Representation

C.1 Autonomy

Autonomy features describe the amount of work the student or the tutor has done, either recently or over a long period. The following 4 features describe the amount of work the student or the tutor has done recently.

- ntellsSinceElicit: The number of tells the student has received since the last elicit.
- ntellsSinceElicitKC: ntellsSinceElicit for the current KC.

- nElicitSinceTell: The number of elicits the student has received since the last tell.
- nElicitSinceTellKC: nElicitSinceTell for the current KC.

The following 6 features describe the amount of work the student or the tutor has done over a long period.

- pctElicit: The total number of elicit steps divided by the total number of steps the students have received so far.
- pctElicitKC: pctElicit for the current KC.
- pctElicitSession: pctElicit for the current session.
- pctElicitKCSession: pctElicit for the current KC and the current session.
- nTellSession: the total number of tells the student has has received so far in the current session.
- nTellKCSession: nTellSession for the current KC.

C.2 Temporal

Temporal features describe time-related information, such as the amount of time the student has spent on the current session or on a specific KC. The following five features are calculated based on the difference between the two timestamps, such as the difference between the current timestamp and the beginning of the current session.

- durationKCBetweenDecision: The time since the last tutorial decision was made on the current KC.
- timeInSession: The time that has elapsed since the start of the current session.
- timeBetweenSession: The time elapsed between the end of the previous session and the beginning of the current one.
- timeOnCurrentProblem: The time elapsed since the start of the current problem.
- timeOnLastStepKCElicit: the time the student spent on the last elicit step with the same KC as the current step.

In the following, the total time is defined as the summation of the time student has spent on certain steps that were the focus of the training. All other intervals, such as between problem intervals or time spent on irrelevant steps, were excluded. The following 12 features describe the total amount of time the student has spent on certain materials.

- timeOnTutoring: The total time the student has spent on the tutoring.
- timeOnTutoringTell: The total time the student has spent on tells.
- timeOnTutoringElicit: The total time the student has spent on Elicits.
- timeOnTutoringKC: The total time the student has spent on the current KC.
- timeOnTutoringKCTell: The total time the student has spent on the current KC with tell.
- timeOnTutoringKCElicit: The total time the student has spent on the current KC with elicit.
- timeOnTutoringSession: The total time the student has spent on the current session.
- timeOnTutoringSessionTell: timeOnTutoringSession with tells.
- timeOnTutoringSessionElicit: timeOnTutoringSession with elicits.
- timeOnTutoringProblem: The total time the student has spent on the current problem.
- timeOnTutoringProblemTell: timeOnTutoringProblem with tells.
- timeOnTutoringProblemElicit: timeOnTutoringProblem with elicits.

The following 12 features describe the student's working speed.

- avgTimeOnStep: The average time the student spent on each step.
- avgTimeOnStepTell: The average time the student spent on each tell step.
- avgTimeOnStepElicit: The average time the student spent on each elicit step.
- avgTimeOnStepKC: avgTimeOnStep for the current KC.
- avgTimeOnStepKCTell: avgTimeOnStepTell for the current KC.
- avgTimeOnStepKCElicit: avgTimeOnStepElicit for the current KC.
- avgTimeOnStepSession: avgTimeOnStep for the current session.
- avgTimeOnStepSessionTell: avgTimeOnStepTell for the current session.
- avgTimeOnStepSessionElicit: avgTimeOnStepElicit for the current session.
- avgTimeOnStepProblem: avgTimeOnStep for the current problem.
- avgTimeOnStepProblemTell: avgTimeOnStepTell for the current problem.
- avgTimeOnStepProblemElicit: avgTimeOnStepElicit for the current problem.

C.3 Problem Solving

Problem solving features describe the context of the learning environment, such as the difficulty of the current problem and the students' progress. The following seven features describe the student's progress and the amount of practice they have done.

- stepOrdering: The total number of steps the student has received so far.
- stepOrderingSession: stepOrdering for the current session.
- stepOrderingPb: stepOrdering for the current problem.
- nKCs: The number of steps the student has completed for the current KC.
- nKCsAsElicit: The number of elicit steps the student has completed for the current KC.
- nKCsSession: nKCs for the current session.
- nKCsSessionElicit: nKCsAsElicit for the current session.

The following nine features describe the category and difficulty level of the current problem or step.

- earlyTraining: For the first two problems and the first conditional probability problem, the value is 1 and for the rest, the value is 0.
- simpleProblem: For the first two problems and the first two conditional probability problems, the value is 1 and for the rest, the value is 0.
- newLevelDifficulty: If the current problem is more complicated than the prior problem, the value is 1; otherwise, the value is 0. In our case, the value is one for the first, third, fifth, eighth, tenth, and twelfth problem.
- performanceDifficulty: Students' average performance on the current KC (calculated based on our historical data). More specifically $\frac{correct\ elicits}{total\ elicits}$ across all students.
- principleDifficulty: The difficulty of the principle needed for the current step, which depends on the equation of the principle. If the step does not require a probability principle, the value is 1 (easiest).
- principleCategory: If the current step requires a probability theorem principle, the value is 1; if it requires a conditional probability principle, the value is 2; and if it does not require a probability principle the value is 0.
- problemDifficulty: The difficulty of the current problem, which is calculated based on the principles needed to solve the problem.
- problemComplexity: The value of this feature is determined by the number of principle applications needed to solve the current problem, 2 for easy problems (first, eighth, eleventh), 3 for medium problems (second, third, ninth and tenth) and 4 for hard problems (fourth, fifth, sixth, seventh, and twelfth).
- problemCategory: If the problem does not require any conditional probability principle to solve, the value is 0, otherwise the value is 1.

The following three features describe the number of principles that appeared in the current problem or session.

- nPrincipleInProblem: The number of principles needed to solve the current problem (some principles may be applied more than once).
- nDistinctPrincipleInSession: The total number of distinct principles that have appeared in the current session.
- nPrincipleInSession: The total number of principles appeared in the current session.

The following nine features describe the tutor's use of words and probability concepts.

- nTutorConceptsSession: The number of probability concepts the tutor has mentioned so far in the current session.
- tutAverageConcepts: The average number of probability concepts the tutor has mentioned in each step.
- tutAverageConceptsSession: tutAverageConcepts for the current session.
- tutConceptsToWords: The number of probability concepts the tutor has mentioned divided by the total number of words the tutor has used so far.
- tutConceptsToWordsSession: tutConceptsToWords for the current session.
- tutAverageWords: the average number of words the tutor used in each step.
- tutAverageWordsSession: tutAverageWords for the current session.

- tutAverageWordsElicit: the average number of words the tutor used in each elicit step.
- tutAverageWordsSessionElicit: tutAverageWordsElicit for the current session.

The following feature is about quantitative and qualitative steps.

- quantitativeDegree: The number of quantitative steps (select principle and apply principle) the student has received divided by the total number of steps the student has completed.

The following six features describe the number of each probability principles needed to solve the current problem. Conditional probability principles are not included because they are not heavily needed for problem solving (in terms of occurrence), and the conditional probability problems appear late in the training process.

- nAdd2Prob: The number of times the Addition Theorem for Two Events is needed to solve the current problem.
- nAdd3Prob: The number of times the Addition Theorem for Three Events is needed to solve the current problem.
- nDeMorProb: The number of times the De Morgan's Theorem is needed to solve the current problem.
- nIndeProb: The number of times the Independent Theorem is needed to solve the current problem.
- nCompProb: The number of times the Complement Theorem is needed to solve the current problem.
- nMutualProb: The number of times the Mutually Exclusive Theorem is needed to solve the current problem.

C.4 Performance

Performance features describe the students' competence level. The following twelve features describe the performance measures calculated based on the number of correct/incorrect steps or the percentage of correct steps.

- pctCorrect: The number of elicit steps the student has correctly solved (on the first attempt) divided by the total number of elicit steps the student has received so far.
- pctOverallCorrect: Denote the number of tell steps the student has received so far as *tells*, the number of elicit steps the student has correctly solved as *correct elicits*, and the total number of steps the student has received so far as *steps*. The feature value is calculated following the equation $\frac{tells+correct\ elicits}{steps}$.
- nCorrectKC: The total number of elicit steps the student has correctly solved for the current KC so far.
- nIncorrectKC: The total number of elicit steps the student failed to solve on the first attempt for the current KC so far.
- pctCorrectKC: pctCorrect for the current KC.
- pctOverallCorrectKC: pctOverallCorrect for the current KC.
- nCorrectKCSession: nCorrectKC for the current session.
- nIncorrectKCSession: nIncorrectKC for the current session.
- pctCorrectSession: pctCorrect for the current session.
- pctCorrectKCSession: pctCorrectKC for the current session.
- pctOverallCorrectSession: pctOverallCorrect for the current session.
- pctOverallCorrectKCSession: pctOverallCorrectKC for the current session.

The following twelve features describe certain types of steps the student has received since the last wrong elicit step.

- nStepSinceLastWrong: The number of steps (both elicit and tell) the student has completed since the last wrong elicit step (where the student failed the first attempt).
- nStepSinceLastWrongKC: nStepSinceLastWrong for the current KC.
- nTellsSinceLastWrong: The number of tell steps the student has received since the last wrong elicit step.
- nTellsSinceLastWrongKC: nTellsSinceLastWrong for the current KC.
- nStepSinceLastWrongSession: nStepSinceLastWrong for the current session.

- nStepSinceLastWrongKCSession: nStepSinceLastWrong for the current KC in the current session.
- nTellsSinceLastWrongSession: nTellsSinceLastWrong for the current session.
- nTellsSinceLastWrongKCSession: nTellsSinceLastWrong for the current KC in the current session.
- timeSinceLastWrongStepKC: The time that has elapsed since the last wrong elicit step for the current KC.
- nCorrectElicitStepSinceLastWrong: The number of elicit steps the student has successfully solved since the last wrong elicit step.
- nCorrectElicitStepSinceLastWrongKC: nCorrectElicitStepSinceLastWrong for the current KC.
- nCorrectElicitStepSinceLastWrongKCSession: nCorrectElicitStepSinceLastWrong for the current KC in the current session.

The following eight features describe students' performance on the steps that require a probability principle (the select- or apply-principle steps).

- pctCorrectPrin: pctCorrect for the steps that require a probability principle.
- pctCorrectPrinSession: pctCorrectPrin for the current session.
- nStepSinceLastWrongPrin: nStepSinceLastWrong for the steps that require a probability principle.
- nTellsSinceLastWrongPrin: nTellsSinceLastWrong for the steps that require a probability principle.
- nStepSinceLastWrongPrinSession: nStepSinceLastWrongPrin for the current session.
- nTellsSinceLastWrongPrinSession: nTellsSinceLastWrongPrin for the current session.
- nCorrectElicitStepSinceLastWrongPrin: nCorrectElicitStepSinceLastWrong for the steps that require a probability principle.
- nCorrectElicitStepSinceLastWrongPrinSession: nCorrectElicitStepSinceLastWrongPrin for the current session.

The following four features describe students' performance on the first occurred select- and apply-principle steps in each problem, which are more complicated than the rest of principle-realted steps.

- pctCorrectFirst: pctCorrect for the first occurred select- and apply-principle steps in each problem.
- nStepsSinceLastWrongFirst: nStepSinceLastWrong for the first occurred select- and apply-principle steps in each problem.
- nTellsSinceLastWrongFirst: nTellsSinceLastWrong for the first occurred select- and apply-principle steps in each problem.
- nCorrectElicitStepSinceLastWrongFirst: nCorrectElicitStepSinceLastWrong for the first occurred select- and apply-principle steps in each problem.

The following two features describe students performance on the last problem.

- pctCorrectLastProb: pctCorrect for all the steps in the last problem.
- pctCorrectLastProbPrin: pctCorrect for all the steps that require a probability principle in the last problem.

The following 18 features describe students' current competence on the six probability principles.

- pctCorrectAdd2Select: pctCorrect for the select-principle steps that require selecting the Addition Theorem for Two Events.
- pctCorrectAdd3Select: pctCorrect for the select-principle steps that require selecting the Addition Theorem for Three Events.
- pctCorrectCompSelect: pctCorrect for the select-principle steps that require selecting the Complement Theorem.
- pctCorrectDeMorSelect: pctCorrect for the select-principle steps that require selecting the De Morgan's Law.
- pctCorrectIndeSelect: pctCorrect for the select-principle steps that require selecting the Independent Theorem.

- pctCorrectMutualSelect: pctCorrect for the select-principle steps that require selecting the Mutually Exclusive Theorem.
- pctCorrectAdd2Apply: pctCorrect for the apply-principle steps that require entering the equation of the Addition Theorem for Two Events.
- pctCorrectAdd3Apply: pctCorrect for the apply-principle steps that require entering the equation of the Addition Theorem for Three Events.
- pctCorrectCompApply: pctCorrect for the apply-principle steps that require entering the equation of the Complement Theorem.
- pctCorrectDeMorApply: pctCorrect for the apply-principle steps that require entering the equation of the De Morgan's Law.
- pctCorrectIndeApply: pctCorrect for the apply-principle steps that require entering the equation of the Independent Theorem.
- pctCorrectMutualApply: pctCorrect for the apply-principle steps that require entering the equation of the Mutually Exclusive Theorem.
- pctCorrectAdd2All: pctCorrect for the select- or apply-principle steps that require the Addition Theorem for Two Events.
- pctCorrectAdd3All: pctCorrect for the select- or apply-principle steps that require the Addition Theorem for Three Events.
- pctCorrectCompAll: pctCorrect for the select- or apply-principle steps that require the Complement Theorem.
- pctCorrectDeMorAll: pctCorrect for the select- or apply-principle steps that require the De Morgan's Law.
- pctCorrectIndeAll: pctCorrect for the select- or apply-principle steps that require the Independent Theorem.
- pctCorrectMutualAll: pctCorrect for the select- or apply-principle steps that require the Mutually Exclusive Theorem.

The following feature describes students' competence in selecting main steps.

- pctCorrectSelectMain: pctCorrect for the steps that require the student to select the next main step.

C.5 Hints

The following five features describe the number of hints the student requested in a certain period.

- nTotalHint: The total number of hints the student has requested so far.
- nTotalHintSession: nTotalHint for the current session.
- nHintKC: nTotalHint for the current KC.
- nHintSessionKC: nTotalHint for current KC in the current session.
- nTotalHintProblem: nTotalHint for the current problem.

The following six features describe the student's hint request behavior or working behavior in hint-requested steps.

- AvgTimeOnStepWithHint: The average time the students spent on each hint-requested step.
- durationSinceLastHint: The time that has elapsed since the last hint was requested.
- stepsSinceLastHint: The number of steps the student has completed since the last hint-requested step.
- stepsSinceLastHintKC: stepsSinceLastHint for the current KC.
- totalTimeStepsHint: The total time the student has spent on hint-requested steps.
- totalStepsHint: The total number of steps where hints were requested.

D Two NLG Definitions

In order to choose a reliable reward measure, we compared two NLG definitions $\frac{posttest-pretest}{1-pretest}$ and $\frac{posttest-pretest}{\sqrt{1-pretest}}$, using the data collected in the Granularity studies. Table 9 shows a comparison of the two NLG scores for the High and Low groups respectively (to be consistent with the score range in the main paper, all numbers are timed 100). As expected, the square root can reduce the variance, especially for the High group (from 120.24 to 35.18). In addition, the square root rose the average of High from -35.79 to -10.04 and reduced the average of Low from 15.23 to 10.15, which reduced the difference between the High and Low groups from 51.02 to 20.19.

Table 9 A Comparison of the Two NLG Definitions

Definition	High	Low
$\frac{posttest-pretest}{1-pretest}$	-35.79(120.24)	15.23(38.73)
$\frac{posttest-pretest}{\sqrt{1-pretest}}$	-10.04(35.18)	10.15(24.98)

E Gaussian Processes for Q-function Approximation

The standard GP Regression is used to approximate the Q function. Remind that in the context of GP, a function can be specified by the mean and a covariance function. In Q-function approximation, it takes state-action-Q observations $(S, A) \to Q$ and a prior covariance function (kernel) as input and specifies the Q-function's posterior mean and covariance: $(\hat{\mathbf{Q}}^{\pi})' \sim \mathcal{N}\left(\overline{(\hat{\mathbf{Q}}^{\pi})'},\ \mathrm{COV}((\hat{\mathbf{Q}}^{\pi})')\right)$. To model possible uncertainty, we add an Independent and Identically-Distributed noise to the prior covariance function: $\mathcal{E} \sim \mathcal{N}\left(0, \boldsymbol{\sigma_n^2}\right)$. According to the theorem of conditional probability density functions for multivariate Gaussians [37], the mean, $\overline{(\hat{\mathbf{Q}}^{\pi})'}$, and covariance $\mathrm{COV}((\hat{\mathbf{Q}}^{\pi})')$ of the posterior distribution can be calculated using the following two equations [19,37]:

$$\overline{(\hat{\mathbf{Q}}^{\pi})'} = K\left(\boldsymbol{X}', \boldsymbol{X}\right)\left[K\left(\boldsymbol{X}, \boldsymbol{X}\right) + \sigma_n^2 \mathbf{I}\right]^{-1} \overline{(\hat{\mathbf{Q}}^{\pi})} \tag{7}$$

$$\begin{aligned}\mathrm{COV}((\hat{\mathbf{Q}}^{\pi})') = K\left(\boldsymbol{X}', \boldsymbol{X}'\right) + \mathbf{C}_{\hat{\mathbf{Q}}^{\pi}} - K\left(\boldsymbol{X}', \boldsymbol{X}\right) \\ \left[K\left(\boldsymbol{X}, \boldsymbol{X}\right) + \sigma_n^2 \mathbf{I}\right]^{-1} K\left(\boldsymbol{X}, \boldsymbol{X}'\right).\end{aligned} \tag{8}$$

where $\boldsymbol{X}$ is the observation points (the state-action pairs (S, A) in our training data), $\overline{(\hat{\mathbf{Q}}^{\pi})}$ and $\mathbf{C}_{\hat{\mathbf{Q}}^{\pi}}$ are the mean and covariance matrix of the corresponding observed Q values for the (S, A) pairs, $\mathbf{X}'$ is the approximation points (the state-action pairs whose Q-value GP estimates), $\boldsymbol{\sigma_n^2}$ is a parameter, $K\left(\boldsymbol{X}, \boldsymbol{X}\right)$ is a covariance matrix evaluated on the observation points, $K\left(\boldsymbol{X}', \boldsymbol{X}'\right)$ is a covariance matrix evaluated on the approximation points, and $K\left(\boldsymbol{X}, \boldsymbol{X}'\right)$ is a covariance matrix evaluated on the observation and approximation points [37].