

## ABSTRACT

ZHOU, GUOJING. Improving Student Learning Through Hierarchical Reinforcement Learning Induced Pedagogical Policies. (Under the direction of Min Chi.)

In interactive e-learning environments such as Intelligent Tutoring Systems, there are pedagogical decisions to make at two main levels of granularity: whole problems and single steps. Here, we focus on making the problem-level decisions of worked example (WE) vs. problem solving (PS) and the step-level decisions of elicit vs. tell. More specifically, we first investigate the impact of decision granularity on student learning and then explore taking granularity into account in data-driven pedagogical policy induction.

In a series of classroom studies, we explored the impact of three types of granularity: problem-level only (Prob-Only), step-level only (Step-Only), and both problem and step levels (Both) on student learning. Results showed that Prob-Only can be more effective for Low incoming competence students, Step-Only can be more effective for High ones, and Both can be effective for both Low and High students. This suggests that granularity indeed can have an impact on student learning. However, there was no significant difference among the three granularity conditions overall. One possible reason is that the pedagogical decisions were randomly made rather than adaptively.

Prior research has shown that effective pedagogical decision-making can significantly improve student learning. In recent years, there has been growing interest in applying data-driven approaches to induce pedagogical policies directly from student-system interaction logs. Though, most of the prior works treated all system decisions *equally, or independently* without considering the long-term impact of higher-level actions or the interaction of decisions made at different levels. Here, we apply reinforcement learning (RL) to induce pedagogical policies that make decisions at different granularity levels and evaluate their effectiveness in empirical classroom studies.

We first applied RL to induce a problem-level and a step-level policy and evaluated their effectiveness in a classroom study by comparing them with two random *yet reasonable* policies, one at the problem-level and one at the step-level. Results showed that there was no significant difference between the two RL conditions, and none of them was significantly more effective than the two random baseline conditions. The results suggest that RL induced policies that make decisions at a single granularity level may not always be effective.

On the other hand, results from the granularity studies showed that Both-level decisions can benefit more students than either the problem- or step-level, and thus, considering both levels of decisions in RL policy induction may lead to effective policies. Therefore, we then applied an offline, off-policy Gaussian Processes based Hierarchical Reinforcement Learning (HRL) approach to induce a hierarchical pedagogical policy that makes decisions at both the problem and step levels. In an empirical classroom study, the HRL policy was compared with a Deep Q-Network (DQN) induced step-level policy and a random *yet reasonable* step-level baseline policy. Results showed that the HRL policy was significantly more effective than the DQN induced policy and the random

baseline policy. The results suggest that by taking decision granularity into account, HRL indeed can induce effective policies that can significantly improve student learning.

© Copyright 2020 by Guojing Zhou

All Rights Reserved

Improving Student Learning Through Hierarchical Reinforcement Learning  
Induced Pedagogical Policies

by  
Guojing Zhou

A dissertation submitted to the Graduate Faculty of  
North Carolina State University  
in partial fulfillment of the  
requirements for the Degree of  
Doctor of Philosophy

Computer Science

Raleigh, North Carolina

2020

APPROVED BY:

---

James Lester

---

Roger Azevedo

---

Tiffany Barnes

---

Min Chi  
Chair of Advisory Committee

## BIOGRAPHY

The author was born in a small city in China called Guilin. Growing up with parents that are professors, he developed an intrinsic interest in thinking and making things easier. However, he never managed to make his own life easy, including his academic path. Before the end of 8th grade, his greatest accomplishment was to make his video game easy through hacking. In the 9th grade, he built a website for video game fans, but it has never been hosted by a real webserver because he did not know how to do so. These activities did not help his academic performance, but they ignited his enthusiasm for building computer applications that can be helpful for people.

Following his father's suggestion that math is the foundation of computer technologies, he joined Beihang University for college study in 2005, and earned his B.S. in Mathematics and Applied Mathematics in 2009. After that, he joined the University of California, Riverside, to pursue his M.S. in Computer Science and earned a degree in 2011. However, the switch was not easy. Math prepared him to understand the theoretical foundation of the algorithms, but it did not get him ready to implement those algorithms effectively and efficiently. Fortunately, after two years of being busy with the stresses of graduate school, he prepared himself to enter the industry. He then went back to China and started his first job as a researcher focusing on data science.

Although he was excited about this job at the beginning as he finally got the opportunity to develop real applications, he inevitably lost interest. He had the skills to build applications, but this job required him to do even more. It required him to design products as well as explain the advantages of these products and who will be interested in purchasing them. It was at this time that he realized he needed to broaden his vision and deepen his thinking by returning to school.

He joined the Department of Computer Science at North Carolina State University in 2014 for his Ph.D. studies. With the goal of making life more convenient, he chose to work with Dr. Min Chi. in the area of artificial intelligence (AI) because AI can automate complex tasks that otherwise could not be easily solved. Luckily, he quickly found his research interests in the intersection of cognitive science and machine learning with a focus on intelligent tutoring systems (ITSs). In cognitive science, he was the first person that explicitly explored the impact of decision granularity on learning; while in machine learning, he made the first attempt to apply hierarchical reinforcement learning (HRL) for pedagogical policy induction. He was happy with his work because it can be useful to researchers, educators, and students. The work was published in strictly peer-reviewed conferences, including Cognitive Science (CogSci), educational data mining (EDM), artificial intelligence in education (AIED), User Modeling, Adaptation and Personalization (UMAP), and International Joint Conference on Artificial Intelligence (IJCAI). His HRL work has also won the best paper award at AIED 2019.

## ACKNOWLEDGEMENTS

I have to thank my advisor, Dr. Min Chi, for all she has done for me during my Ph.D. studies. My Ph.D. path was a lot harder than I thought it would be at the beginning, and it was with Dr. Chi's great help that I have been able to complete my Ph.D. smoothly. In the six years, I have learned a lot of things beyond doing research, and these lessons will be a valuable asset in my life forever. One lesson that I have learned is that research needs to be done with faith. It is much more than just a word. We put a lot of time and effort into developing our approach, but sometimes, it may just not work. It was the unbreakable trust in science and enormous curiosity in nature that held me through, and I have to say that Dr. Chi guided me to find the faith. Another precious lesson I learned is to face challenges with strength, confidence, and optimism. If I can manage to do that, then I can overcome any obstacle. I am very fortunate to have received all this guidance from Dr. Chi, and I cannot imagine how my life would be without them.

I would also like to thank the rest of my committee, Dr. James Lester, Dr. Roger Azevedo, and Dr. Tiffany Barnes for devoting time to my work. Dr. Barnes gave me a lot of help during my entire Ph.D. studies in every aspect, such as study design, data analysis, and writing. She fixed countless grammar issues in my papers, and her editing improved the quality of my paper a lot. She is like my second advisor. Dr. Lester asked interesting questions in my defense, full of imagination and inspiration, which really broadened my vision. Dr. Azevedo asked deep and fundamental questions, which are very useful to my future research.

Additionally, I need to thank all my friends for making my Ph.D. life colorful. I will never forget the moments we created together, making food, eating hotpot, cooking barbecue, going shopping, playing sports, watching movies or shows, and so forth. It was these moments that supported me to tackle challenges. It was also these moments that motivated me to create a bright future for everyone I love and care about.

Finally, and most importantly, I have to thank my family. My parents created a golden childhood for me that was very innocent and happy. My childhood was the best thing that I have ever experienced in the world and I look back on it fondly. It gave me the unlimited motivation to move on, especially when I came upon hard times. I wish to someday create a golden childhood for all of my children (though I don't have any right now) so that they too may experience what I was fortunate enough to have as a child. With the completion of my Ph.D. studies, I believe that I am getting one step closer to this goal.

## TABLE OF CONTENTS

<b>LIST OF TABLES</b> .....	<b>vi</b>
<b>LIST OF FIGURES</b> .....	<b>vii</b>
<b>Chapter 1 Introduction</b> .....	<b>1</b>
1.1 Decision Granularity .....	1
1.2 Experimental Studies .....	2
1.2.1 Granularity Studies .....	2
1.2.2 Reinforcement Learning (RL) Study .....	3
1.2.3 Hierarchical Reinforcement Learning (HRL) Study .....	4
1.3 Contributions .....	5
1.4 Organization .....	5
<b>Chapter 2 Our ITS and the General Experiment Procedure</b> .....	<b>7</b>
2.1 Our ITS - Pyrenees .....	7
2.1.1 Probability Principles .....	9
2.1.2 Problem Solving Stages and Steps .....	9
2.1.3 Elicit vs. Tell .....	12
2.1.4 Feedback and Hints .....	13
2.2 Experimental Procedure .....	13
2.2.1 Textbook .....	13
2.2.2 Pre-test .....	15
2.2.3 Training on Pyrenees .....	15
2.2.4 Post-test .....	16
2.2.5 Grading .....	17
<b>Chapter 3 Granularity Studies</b> .....	<b>20</b>
3.1 Background - Prior Research on WE vs. PS vs. FWE vs. CPS .....	21
3.2 Experiment Setup .....	22
3.2.1 Participants .....	22
3.2.2 Conditions .....	22
3.2.3 Procedure and Measures .....	23
3.2.4 High vs. Medium vs. Low Students .....	24
3.3 Fall 2014 Empirical Study Results .....	25
3.3.1 Analysis on the Impact of Granularity .....	26
3.3.2 Analysis on Granularity and Incoming Competence .....	26
3.4 Post-hoc Analysis Results .....	27
3.4.1 Analysis on the Impact of Granularity .....	29
3.4.2 Analysis on Granularity and Incoming Competence .....	29
3.5 Conclusion and Discussion of Chapter 3 .....	31
<b>Chapter 4 Data Collection and Processing for Reinforcement Learning</b> .....	<b>33</b>
4.1 Markov Decision Processes .....	33
4.2 Training Corpus .....	34
4.3 Properties of Steps .....	35
4.3.1 Step Duration .....	36

4.3.2	Step Correctness . . . . .	36
4.3.3	Knowledge Components (KCs) . . . . .	36
4.3.4	Other Elements . . . . .	36
4.4	Environment Features . . . . .	37
4.4.1	Autonomy . . . . .	37
4.4.2	Temporal . . . . .	38
4.4.3	Problem Solving . . . . .	41
4.4.4	Performance . . . . .	45
4.4.5	Hints . . . . .	51
<b>Chapter 5 Reinforcement Learning (RL) Study - Comparing RL Induced Problem-Level vs. Step-Level Policies . . . . .</b>		<b>53</b>
5.1	Background - Prior Research on Applying RL for Pedagogical Policy Induction . . . . .	54
5.2	Method . . . . .	55
5.2.1	Feature Discretization . . . . .	56
5.2.2	Policy Induction . . . . .	57
5.2.3	Feature Selection . . . . .	57
5.2.4	Training Data . . . . .	58
5.3	Empirical Experiment . . . . .	59
5.3.1	Conditions and Participants . . . . .	59
5.3.2	Procedure and Measures . . . . .	60
5.3.3	Results . . . . .	61
5.4	Conclusion and Discussion for Chapter 5 . . . . .	63
<b>Chapter 6 Hierarchical Reinforcement Learning (HRL) Study - Inducing a Hierarchical Pedagogical Policy . . . . .</b>		<b>65</b>
6.1	Background - HRL Approaches and Applications . . . . .	66
6.2	Method . . . . .	67
6.2.1	Training Data . . . . .	67
6.2.2	Challenges for HRL Policy Induction . . . . .	68
6.2.3	GP-Based Immediate Rewards Inference . . . . .	70
6.2.4	HRL for Policy Induction . . . . .	71
6.2.5	DQN for Policy Induction . . . . .	73
6.3	Evaluation Experiment . . . . .	74
6.3.1	Conditions and Participants . . . . .	74
6.3.2	Procedure and Measures . . . . .	74
6.3.3	Results . . . . .	76
6.4	Conclusion and Discussion for Chapter 6 . . . . .	80
<b>Chapter 7 General Discussion, Conclusion, and Future Work . . . . .</b>		<b>82</b>
7.1	The Impact of Granularity on Student Learning . . . . .	82
7.2	The Effectiveness of RL Induced Problem- vs. Step-Level Policies . . . . .	84
7.3	HRL for Pedagogical Policy Induction . . . . .	85
7.4	Conclusion and Future Work . . . . .	87
<b>BIBLIOGRAPHY . . . . .</b>		<b>89</b>

## LIST OF TABLES

Table 2.1	Probability Principles . . . . .	9
Table 2.2	An Example Training Problem . . . . .	10
Table 2.3	Elicit vs. Tell . . . . .	11
Table 2.4	Example Hint/Feedback Sequence . . . . .	12
Table 2.5	Single-principle Problem vs. Multiple-principle Problem . . . . .	15
Table 2.6	Principles Included in Each Pre-test Problem . . . . .	16
Table 2.7	Principles Included in Each Pyrenees Problem . . . . .	17
Table 2.8	An Example Pair of Isomorphic Problems in Pre- and Post-test . . . . .	18
Table 2.9	Principles Included in Each Post-test Problem . . . . .	19
Table 3.1	Procedure of the Granularity Studies . . . . .	23
Table 3.2	Fall 2014 Empirical Study Results . . . . .	25
Table 3.3	Results on High, Medium, and Low Students for Fall 14 . . . . .	28
Table 3.4	Fall 2015-2017 Empirical Studies Results . . . . .	28
Table 3.5	Post-hoc Analysis Results . . . . .	28
Table 3.6	Post-hoc Analysis Results on High, Medium and Low students . . . . .	30
Table 4.1	Training Data Information . . . . .	35
Table 5.1	Procedure of the RL study . . . . .	61
Table 5.2	Results of the Fall 2016 RL Study . . . . .	61
Table 5.3	Tutor Decisions in Terms of Elicits and Tells . . . . .	63
Table 6.1	A Comparison of the Two NLG Definitions . . . . .	68
Table 6.2	Procedure of the HRL study . . . . .	75
Table 6.3	HRL Study Results . . . . .	76
Table 6.4	Step-Level Tutor Decisions . . . . .	79

## LIST OF FIGURES

Figure 2.1	The Interface of Our Probability Tutor - Pyrenees . . . . .	8
Figure 2.2	A Screenshot of the Pyrenees Textbook . . . . .	13
Figure 2.3	An Example of the Practice Problems in the Pyrenees Textbook . . . . .	14
Figure 2.4	Correct Answers for Textbook Practice Problems are Marked in Green . . . . .	14
Figure 2.5	Incorrect answers for Textbook Practice Problems are Marked in Red . . . . .	14
Figure 3.1	Students' Improvement and Time on Task . . . . .	25
Figure 3.2	Adjusted Post-test and Time on Task for Post-hoc Analysis on Granularity and Incoming Competence . . . . .	30
Figure 5.1	Features with Different Distribution . . . . .	56
Figure 5.2	Students' Improvement and Learning Performance . . . . .	62
Figure 6.1	Flow Chart of the HRL Policy Induction Procedure . . . . .	69
Figure 6.2	Students' Improvement Through Training . . . . .	77
Figure 6.3	Students' Learning Performance . . . . .	78
Figure 6.4	Students' Time on Task . . . . .	78
Figure 6.5	Problem level decisions for the HRL policy, with 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 ranks, with the highest on the top. . . . .	79
Figure 6.6	Percentage of Elicit for the a. HRL, b. DQN and c. Baseline policies' step-level decisions on each problem. The higher the percentage, the darker a cell is. Problem-level WEs and PS 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 ranks, with the highest on the top. . . . .	79

## CHAPTER

# 1

# INTRODUCTION

## 1.1 Decision Granularity

Worked examples (WE) and problem solving (PS) have a long history of being used for instructional purposes. In PS, students are given tasks to complete either independently or with assistance; while in WEs, students are given detailed solutions. 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 [Liz06]. At the end of the worked solutions, it would often state “this way you may solve similar problems,” or “by the same method solve all similar problems,” [Swe87] indicating that WEs have been used as an instructional approach for a long time. Similarly, evidence of PS was also found in early Egyptian, Babylonian, and Chinese artifacts [Swe95]. 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. In recent decades, there has been a growing interest in utilizing WEs and PS in e-learning environments such as intelligent tutoring systems (ITSs). While a great deal of research has investigated how they can be best used to improve student learning [Swe85; McL08; McL11; McL14; VG11; Ren02; Sch09; Naj14; Sal10; Zho15; Zho16; Zho17a; Zho17b; Zho19], the ultimate instructional strategy has yet to be found.

In 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. As a result, VanLehn described tutoring in such domains as a two-loop procedure [Van06]. 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, by contrast, 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. Based on this two-loop structure, pedagogical decisions can be categorized into three types of granularity: 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 the student how to solve the next problem directly using a WE or asks the student to solve the problem on their own via PS. Presenting a WE is a passive instructional action where the student is not required to act, while PS is an active event that requires a student to produce a solution based on their own knowledge either with or without help. Prior studies have shown that combining WEs with PS can be more effective than using PS alone [McL08; McL11; McL14; Swe85]. Note that at the problem level, the tutor can also give the student an example with certain steps left out and require the student to complete the solution. In learning literature, such problems are referred to as incomplete examples or faded worked examples (FWEs) [Ren02; Sch09]. Here, we focus only on the decisions that select between WE and PS.

**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 [Sal10], and students sometimes use bottom-out hints as tells to help them learn [Shi11].

**Both:** Finally, a tutor can make decisions at both the problem and the 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 a WE is fundamentally different from an all-tell CPS in that WE is *determined* at the problem level; while tells in CPS are decided at the step level. The same argument applies to PS vs. all-elicite CPS.

## 1.2 Experimental Studies

### 1.2.1 Granularity Studies

In a series of four classroom studies conducted in the Fall 2014-2017 semesters, we investigated the impact of granularity on student learning. In Fall 2014, we directly compared 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 following 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. Then, motivated by the aptitude treatment interaction (ATI) theory, we split students into different incoming competence groups and conducted a two-factor post-hoc analysis on granularity and incoming competence. Results showed that Prob-only can be more effective for Low students, Step-only can be more effective for High ones, and more importantly, Both can be effective for both Low and High students. For time on task, the Prob-Only and Both conditions spent significantly less time than the Step-Only condition. This suggests that granularity indeed can have an impact on student learning, but its impact may depend on students' current competence level. It also provides a possible explanation for the tie in learning performance between the three conditions, in that the random decisions did not align the instructional intervention with students' competence level in an effective way. In order to investigate how adaptive decision-making at different granularity levels would impact student learning, we then applied reinforcement learning (RL) to induce pedagogical policies.

### **1.2.2 Reinforcement Learning (RL) Study**

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 [And95; Pho10]. An abundance of prior research has shown that pedagogical policies can significantly impact [Swe85; McL08; McL11; McL14; Ren02; Sch09; Naj14; Sal10] or improve [Ren02; Naj14; Sal10] student learning. However, currently, there is no existing well-established theory on how to make pedagogical decisions effectively.

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., [Chi11; She16b; Man14; Row15; Row14; Raf16; Cle16; Sta11; Igl09a; Igl09b; Zho17b]). While promising, this work still has a major limitation: the impact of decision granularity has never been considered.

In the RL study, we compared the effectiveness of RL-induced problem-level and step-level policies. The same tabular RL approach was used to induce both policies. To comprehensively represent the learning environment state, we extracted 142 features from student-system interaction

logs. However, since tabular RL cannot handle large continuous state space, we discretized the feature values and performed feature selection to control the size of state space. The training data were collected by training students in our ITS using *random yet reasonable* problem- or step-level policies. 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.

In an empirical classroom study, the RL induced problem- and step-level policies were compared with a random problem-level policy and a random step-level policy. Results showed that there was no significant difference between 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.

### 1.2.3 Hierarchical Reinforcement Learning (HRL) Study

Motivated by the results from the granularity study that Both level decisions can benefit more students than either the problem- or step-level, we applied HRL to induce a policy that makes decisions at both the problem- and step-levels. While a considerable amount of prior research has applied RL for pedagogical policy induction, most of this work treats all system decisions *equally or independently* and does not account for the long-term impact of higher-level actions or the interaction of decisions made at different levels. Human decision-makers treat these distinct levels of granularity differently and are capable of selecting between them [Eve06; Laj13]. Therefore, we apply an offline, off-policy Gaussian Processes-based (GP-based) Hierarchical Reinforcement Learning (HRL) approach to induce hierarchical pedagogical policies.

The policy induction procedure involves three phases: 1) data pre-processing, 2) immediate rewards inference, and 3) policy induction. In the pre-processing stage, similar to the RL study, we extracted 142 features from system logs to represent the learning environment state. Then, we applied a Gaussian Processes based approach to infer “immediate rewards” from delayed rewards, which provides the HRL agent with the reward information at different granularity levels for hierarchical policy induction. Once the rewards were inferred, we then applied a Gaussian Process (GP) based HRL framework [Sut99] to induce the hierarchical policy. We chose GP to approximate the Q-value function because it can handle the noise and uncertainty in our training data. The HRL policy consists of a problem level policy and a step level policy for each individual problem. 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-elicited 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 compared 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 was induced using the same data and inferred immediate rewards that were 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 is 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.

### **1.3 Contributions**

The granularity studies contribute to cognitive science in that they made the first attempt to explicitly investigate the impact of granularity on student learning. A great amount of prior research has involved problem-level WE/PS and step-level elicit/tell, but to the best of our knowledge, no prior work has directly compared the three types of granularity before. Additionally, most prior studies employed effective pedagogical policies to control the instructional intervention, but here, we used random decisions to factor out the impact of pedagogical policies.

The second contribution of the granularity study is that it provided evidence that the granularity indeed can have an impact on student learning. It suggests that step-level elicit and tell are different from problem-level PS and WE, which, in turn, informs researchers and educators that decision granularity should be considered in pedagogical decision-making.

The contribution of the RL study is that it was the first work to compare RL induced problem- and step-level policies. Prior research has applied RL for pedagogical policy induction, but no prior work has investigated whether decision granularity can impact the effectiveness of RL policies. Results in the RL study suggest that decisions made at a single granularity level by RL induced pedagogical policies may not always be effective.

The contribution of the HRL study is that it made the first attempt to apply HRL for pedagogical policy induction. While many RL approaches have been applied for pedagogical policy induction, most of them treat all system decisions equally or independently, disregarding the long-term impact that tutor decisions may have across different levels of granularity. In our HRL approach, problem- and step-level decisions are treated differently, allowing the agent to consider their impact explicitly and separately.

The second contribution is that it provided empirical evidence that HRL can be more effective than flat RL in pedagogical policy induction. For machine learning research, it showed the advantage of HRL in one more area, while for cognitive science research, it supported our hypothesis that taking decision granularity into account can result in more effective decision-making.

### **1.4 Organization**

My dissertation is organized as follows: Chapter 2 describes our ITS and the general experiment procedure. Chapter 3 shows the granularity studies which compared three types of granularity:

problem-level only (Prob-Only), step-level only (Step-Only) and both the problem and step levels (Both). Chapter 4 describes the data collection and processing procedure for RL and HRL policy induction. Chapter 5 shows the RL study where RL induced problem- and step-level policies were compared with random problem- and step-level policies. Chapter 6 describes the HRL study, showing details of our offline off-policy Gaussian Processes based HRL approach and an empirical classroom study where the HRL induced policy was compared with a DQN induced step-level policy and a random step-level baseline policy. Finally, Chapter 7 presents the general discussion, conclusion, and future work. For each of the three studies (Chapter 3, 5, 6), we describe the background, method, results, discussion, and conclusion in turn.

## CHAPTER

# 2

## OUR ITS AND THE GENERAL EXPERIMENT PROCEDURE

This chapter describes our ITS and the general experimental procedure used in all the studies. Our ITS is a web-based application that teaches ten probability principles, such as the Addition Theorem and Bayes' Theorem. It delivers training through guiding students to complete probability problems. Each problem requires the students to go through three stages: 1) problem formalization (defining variables), 2) inference (applying principles to generate equations), and 3) equation solving. Each stage includes multiple steps. In elicit steps, immediate feedback and on-demand hints are provided, while in tell steps, the solution is directly shown.

The experiment procedure includes four phases: 1) textbook 2) pre-test 3) ITS training, and 4) post-test. During **textbook**, all students read a general description of the ten probability principles being taught, 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. During **training on the ITS**, all students received the same 12 problems in the same order. Finally, all students took the 20-problem **post-test**: 14 of the problems were isomorphic to the pre-test, and the remainder were non-isomorphic multiple-principle problems. The remaining chapter elaborates on the details of the ITS and the procedure.

### 2.1 Our ITS - Pyrenees

Our ITS - Pyrenees, is a web-based application that teaches ten probability principles, such as the Addition Theorem and Bayes' Theorem. The training is conducted by guiding students to complete

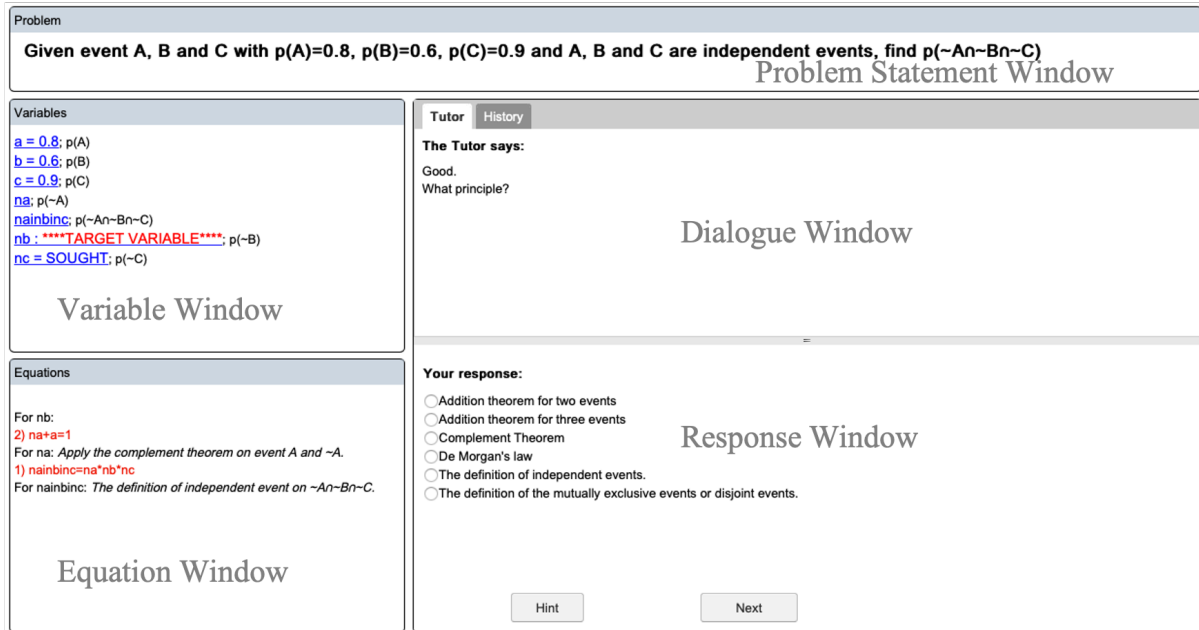


Figure 2.1 The Interface of Our Probability Tutor - Pyrenees

training problems. All instructions, such as explanations, feedback, and hints, are presented using natural language. Figure 2.1 shows the interface of Pyrenees, which is divided into multiple windows.

**Problem Statement window:** shows a statement of the problem for the student to complete.

**Variable window:** shows the variables defined during problem solving, listing one variable per line. Variables whose value is known are shown in the form: "variable name = variable value; the probability event" For example,  $a = 0.8; p(A)$ . Those whose value is unknown and there is currently no equation to determine its value are marked as SOUGHT, showing in the form: "variable name=SOUGHT; the probability event". For example,  $nc = SOUGHT; p(\sim C)$ . For the one that is chosen as the current target variable, the sign **\*\*\*\*TARGET VARIABLE\*\*\*\*** is attached next to its name. For example,  $nb : ****TARGET VARIABLE****; p(\sim B)$ . Once the equation is generated for the current target variable, the sign **\*\*\*\*TARGET VARIABLE\*\*\*\*** will be removed, showing the variable name and the probability event. For example,  $na : p(\sim a)$ .

**Equation window:** lists all the equations generated during problem solving (either by the tutor or the student). Each equation shows the algebraic expression, the target event, and the probability principle behind the equation. For example:

1)  $nainbinc=na*nb*nc$

For nainbinc: *The definition of independent event on  $\sim A \cap \sim B \cap \sim C$ .*

**Dialogue window:** All tutor instructions generated during training, such as explanations, feedback, and hints, are shown in the dialogue window. On the top of the dialogue window, there are two buttons. Clicking the [Tutor] button shows the tutor's current message; while clicking the [History] button shows the history of the dialogue, showing both the tutor's instruction and the student's response.

**Response Window:** Students answer the tutor’s questions in the response window. Questions may be presented in one of the three forms: multiple-choice, event specification, or equation generation. Once the complete answer is entered, students click the [next] button to submit the answer. At any time point, students can ask for help by clicking the [Help] button.

**Table 2.1** Probability Principles

Principle Name	Equation
Addition Theorem for two events (A2)	$p(A \cup B) = p(A) + p(B) - p(A \cap B)$ .
Addition Theorem for three events (A3)	$p(A \cup B \cup C) = p(A) + p(B) + p(C) - p(A \cap B) - p(A \cap C) - p(B \cap C) + p(A \cap B \cap C)$ .
De Morgan’s Theorem (DMT)	$p(\sim (A \cap B)) = p(\sim A \cup \sim B)$ , $p(\sim (A \cup B)) = p(\sim A \cap \sim B)$ .
Independent Events (IE)	$p(A \cap B) = p(A) \times p(B)$ iff $A$ and $B$ are independent events.
Complement Theorem (CT)	$p(A) + p(\sim A) = 1$ .
Mutually Exclusive Theorem (MET)	$p(A \cap B) = 0$ iff $A$ and $B$ are mutually exclusive events.
Total Probability Theorem (TPT)	$p(A) = p(A B_1) + p(A B_2) + \dots + p(A B_n)$ if $B_1, B_2, \dots, B_n$ are mutually exclusive events and $B_1 \cap B_2 \cap \dots \cap B_n = \Omega$ .
Conditional Probability (CP)	$p(A \cap B) = p(A B) \times p(B)$ , $p(B A) \times p(A)$ .
Conditional Independent Events (CIE)	$p(A \cap B C) = p(A C) \times p(B C)$ if $A$ and $B$ are independent events given $C$ .
Bayes Rule (BR)	$p(B_i A) = p(A B_i) \times p(B_i) / p(A B_1) \times p(B_1) + p(A B_2) \times p(B_2) + \dots + p(A B_n) \times p(B_n)$ . if $B_1, B_2, \dots, B_n$ are mutually exclusive events and $B_1 \cap B_2 \cap \dots \cap B_n = \Omega$ .

### 2.1.1 Probability Principles

Pyrenees teaches ten probability principle: including six probability axioms and four conditional probability theorems. Table 2.1 shows 10 principles being taught and the corresponding equation.

### 2.1.2 Problem Solving Stages and Steps

The tutor solves problems using a general, abstract, and domain-independent strategy called the target variable strategy (TVS). TVS solves a problem through three stages: 1) problem formalization, 2) inference (applying principles to generate equations), and 3) equation solving. Each stage involves multiple main steps. Each main step starts with a sub-step of selecting what to do next and then carries it out via one or multiple sub-steps. In our studies, step-level decisions are made at the sub-step level. Next, I will elaborate on the process in detail.

#### 2.1.2.1 Problem Formalization

The problem formalization phase includes two main steps: 1) define a given quantity, and 2) define a sought quantity. A given quantity is defined by giving it a name and specifying its value. In order

to save students time and typing, once the student selected “define a given quantity” in the select-what-to-do sub-step, the tutor will pick an event, give it a name, and specify its value (if there are still undefined event(s)). In this sense, the tutor only requires the student to determine whether there is still an undefined event. A sought quantity is the event that the problem statement requires the student to determine its value (the goal event). Once the student selects “define a sought quantity” in the select-what-to-do sub-step, the tutor will require the student to specify the event, and then the tutor will give it a name and mark the variable as “Sought”. All variables defined in this stage are shown in the variable window.

**Table 2.2** 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 equation 1): $p(\sim A \cap \sim B \cap \sim C) = p(\sim A) * p(\sim B) * p(\sim C)$	-
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: quit	move to next problem

### 2.1.2.2 Inference - Principle Application and Equation Generation

Once all the quantities are defined, students enter the inference stage in which they apply probability principles to determine the value of the goal event. This stage involves the main step “apply a principle”, which includes four sub-steps: 1) select a target variable, 2) select the principle, 3) generation the equation, and 4) update marks. In target variable selection, the tutor lists all the variables that need an equation to determine its value (the variables marked as sought), and the student picks one from them as the current target. Then, the student selects a principle to apply to the current target and writes an equation for the application. If the equation involves any undefined probability events, the student will be asked to define them before writing the equation. Finally, the tutor removes the sought mark for the current target and places a sought mark on the newly defined events. This procedure repeats until there is no sought event left.

**Table 2.3** Elicit vs. Tell

	<b>Elicit</b>	<b>Tell</b>
Select Target Variable	<b>Tutor (T):</b> Please select a target variable. <b>Student (S):</b> $p(\sim AU \sim BU \sim C)$	<b>T:</b> I chose $p(\sim AU \sim BU \sim C)$ as the target variable. <b>S:</b> Go on.
Select Principle	<b>T:</b> I marked $p(\sim AU \sim BU \sim C)$ as the target variable in the variable window. You need to pick a principle to apply on it. Which group of principles will you pick the principle from? <b>S:</b> The axioms of probability. <b>T:</b> What principle? <b>S:</b> Addition theorem for three events.	<b>T:</b> There is 1 principle whose equation contain the target variable: Addition theorem for three events: $\sim A, \sim B, \sim C$ . Therefore, I chose that principle for the target event. <b>S:</b> Go on.
Apply Principle	<b>T:</b> Please enter the equation for Addition theorem for three events: $\sim A, \sim B, \sim C$ . <b>S:</b> $p(\sim AU \sim BU \sim C) = p(\sim A) + p(\sim B) + p(\sim C) - p(\sim A \cap \sim B) - p(\sim A \cap \sim C) - p(\sim B \cap \sim C) + p(\sim A \cap \sim B \cap \sim C)$	<b>T:</b> The equation for applying the Addition Theorem for three events on $p(\sim AU \sim BU \sim C)$ is: $p(\sim AU \sim BU \sim C) = p(\sim A) + p(\sim B) + p(\sim C) - p(\sim A \cap \sim B) - p(\sim A \cap \sim C) - p(\sim B \cap \sim C) + p(\sim A \cap \sim B \cap \sim C)$ <b>S:</b> Go on.

### 2.1.2.3 Equation Solving

Once all the equations are generated, the last stage is to solve them via the main step “solve an equation.” Similar to “define a given quantity,” the tutor only requires the student to determine whether there are still unsolved equations. If there are, the student selects “solve an equation” in the select-what-to-do sub-step, and the tutor will pick one equation and solve it. Once all the equations are solved, the tutor tells the student that the problem is successfully solved. Table 2.2 shows an

example training problem (the steps that the tutor automatically completes for the student are excluded).

### 2.1.3 Elicit vs. Tell

For each step, the system 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. Table 2.3 shows examples of elicit vs. tell.

**Table 2.4** Example Hint/Feedback Sequence

Order	Content
0	[ <i>dialogue</i> ] What principle? [ <i>multiple choices</i> ] <input type="radio"/> Addition theorem for two events; <input type="radio"/> Addition theorem for three events; <input type="radio"/> Complement Theorem;
1	[ <i>dialogue</i> ] ( <b>hint as an instructional message</b> ) The following are short names for probability principles that you can apply to find the target variable. Ask for help again to see longer names. [ <i>multiple choices</i> ] <input type="radio"/> Addition theorem for two events; <input type="radio"/> Addition theorem for three events; <input type="radio"/> Complement Theorem;
2	[ <i>dialogue</i> ] Here are the same choices with longer descriptions. Please choose one. [ <i>multiple choices</i> ] ( <b>hint as a longer description of the choices</b> ) <input type="radio"/> Addition theorem for two events: $p(A \cup B) = p(A) + p(B) - p(A \cap B)$ ; <input type="radio"/> Addition theorem for three events: $p(A \cup B \cup C) = p(A) + p(B) + p(C) - p(A \cap B) - p(A \cap C) - p(B \cap C) + p(A \cap B \cap C)$ ; <input type="radio"/> Complement Theorem: If $A_1$ and $A_2$ is a collection of mutually exclusive and exhaustive events, then $p(A_1) + p(A_2) = 1$ ;
3	[ <i>dialogue</i> ] ( <b>hint as an instructional message</b> ) Selecting a principle is a very difficult step. It needs a lot of practice. The selection is not only based on the variables we already know but also based on how difficult it is for us to find out the value of the new variables produced by applying the principle. Here are all the possible correct choices. Each of the choices does NOT guarantee to put you on the right path. Please think carefully, and you can pick anyone. [ <i>multiple choices</i> ] ( <b>hint as elimination of the choices</b> ) <input type="radio"/> Addition theorem for three events: $p(A \cup B \cup C) = p(A) + p(B) + p(C) - p(A \cap B) - p(A \cap C) - p(B \cap C) + p(A \cap B \cap C)$ ;
4	[ <i>dialogue</i> ] ( <b>bottom-out hint</b> ) Let's just choose Addition theorem for three events.

# 1.1 Events

**Definition 1:** The result of a random trial is called an *outcome*.

**Definition 2:** The set of all the possible outcomes from an experiment is called the *sample space*.

**Definition 3:** A collection of outcomes is called an *event*.

Thus an event is a subset of the sample space (or possibly the whole sample space). Outcomes are denoted by numbers or letters, in this textbook often by  $a_1, a_2 \dots$ . The sample space is denoted by  $\Omega$ . Events are denoted by capital letter  $A, B, C \dots$

**Definition 4.** If the number of outcomes is finite or countable,  $\Omega$  is said to be a discrete sample space. More specifically, if the number of outcomes is finite,  $\Omega$  is said to be a finite sample space. For example: the sample space of tossing a coin is {Head, Tail}. It is discrete and finite because it has only two possible outcomes.

Back Next

**Figure 2.2** A Screenshot of the Pyrenees Textbook

## 2.1.4 Feedback and Hints

In elicited, if the student gives any wrong answer, the tutor will provide feedback on it. The feedback 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. Pyrenees 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. Hints may be delivered as a tutor's instructional message, a longer description of the choices, or elimination of some choices. Table 2.4 shows an example hint sequence, showing the message displayed in the dialogue window and the choices presented in the response window (Since hints are the main focus here, we omitted some choices).

## 2.2 Experimental Procedure

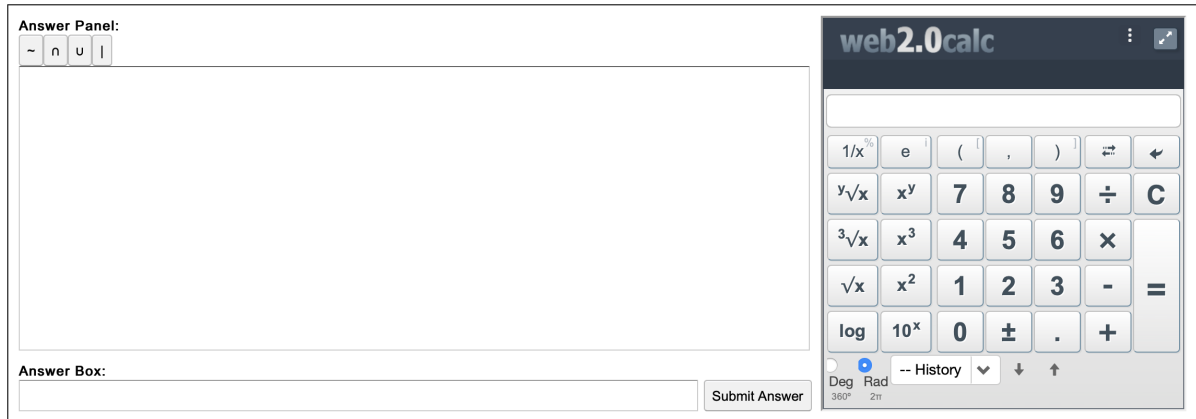
The experimental procedure includes four phases: 1) textbook, 2) pre-test, 3) ITS training, and 4) post-test. This section illustrates them in detail.

### 2.2.1 Textbook

The textbook is shown using ordinary HTML pages. Figure 2.2 shows a screenshot of the textbook. When finishing reading a page, students click the [Next] button to go to the next page. They can also go back to previous pages by clicking the [Back] button.

The textbook consists of three main parts: 1) general information about events and probability, 2) axioms of probability, and 3) conditional probability. The general information section introduces basic probability concepts, such as outcome, sample space, and event, with several examples.

Given A and B are mutually exclusive events such that  $p(A)=0.3$ ,  $p(B)=0.4$ . What is the value of  $p(A \cap B)$



**Figure 2.3** An Example of the Practice Problems in the Pyrenees Textbook

Question: Given A and B are mutually exclusive events such that $p(A)=0.3$ , $p(B)=0.4$ . What is the value of $p(A \cap B)$	
Your Answer	An Example Of Correct Answer
Your answer is: 0	From the problem statement: A and B are mutually exclusive events, so $A \cap B = \text{null set}$ . Apply the definition of mutually exclusive events. : $p(A \cap B)=0$ .

**Figure 2.4** Correct Answers for Textbook Practice Problems are Marked in Green

Question: Given A and B are mutually exclusive events such that $p(A)=0.3$ , $p(B)=0.4$ . What is the value of $p(A \cap B)$	
Your Answer	An Example Of Correct Answer
Your answer is: 0.12	From the problem statement: A and B are mutually exclusive events, so $A \cap B = \text{null set}$ . Apply the definition of mutually exclusive events. : $p(A \cap B)=0$ .

**Figure 2.5** Incorrect answers for Textbook Practice Problems are Marked in Red

The probability axioms section teaches six probability axioms. For each axiom, it first gives a text description of the definition, the corresponding equations, and some examples of its application. Then a practice problem for the axiom is given. If students failed to solve it, they were asked to solve an isomorphic one; this process repeated until they either failed three times or succeeded once. Figure 2.3 shows a screenshot of the textbook practice problem. Students were required to give a detailed step-by-step solution in the large textbox and enter the final answer in the small answer box at the bottom. Once the student submits the answer, the tutor will give feedback based on the answer in the small answer box as well as the correct solution. Correct answers will be marked in green, and incorrect answers will be marked in red, as shown in Figure 2.4 and Figure 2.5, respectively.

Similar to the probability axioms section, the conditional probability section introduces condi-

tional probability theorems by giving their definition, application examples, and practice problems. Since the Total Probability Theorem and Bayes Rule are advanced concepts, two practice problems are given, an easy one and a challenging one.

**Table 2.5** Single-principle Problem vs. Multiple-principle Problem

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

### 2.2.2 Pre-test

The pre-test contains ten single-principle problems, one for each of probability theorems being taught and four multiple-principle problems. Table 2.5 shows a comparison of a single-principle and a multiple-principle problem. Each single-principle problem requires applying a single principle to solve while each multiple-principle problem requires applying multiple principles in a logical order. Table 2.6 shows the probability principles required to solve each problem, with the first column showing the order of the problem and rest ten columns showing the number of times of each principle needs to be applied. A blank cell means the corresponding principle is not required.

The pre-test interface was the same as the one used for practice problems in the textbook, as shown in Figure 2.3. As required in the textbook, for each problem, students need to give a detailed step-by-step solution and provide the final answer separately. However, no feedback was given to their answers, and they were not allowed to go back to earlier problems. This was also true for the post-test.

### 2.2.3 Training on Pyrenees

During training on Pyrenees, all students studied the same 12 problems in the same order (redo or review previously completed problems were not allowed). Each primary domain principle was applied at least twice. The minimum number of steps needed to solve each training problem ranged

**Table 2.6** Principles Included in Each Pre-test Problem

p#	CT	MET	IE	DMT	A2	A3	CIE	CP	TPT	BR
1						1				
2									1	
3								1		
4	1			1	1					
5										1
6	3						2			1
7		3	2			1				
8	3	2		1	2	1				
9	1									
10					1					
11			1							
12				1						
13							1			
14		1								

from 20 to 50. The steps included variable definitions, principle applications, and equation solving. The number of domain principles required to solve each problem ranged from 3 to 11 (with some principles applied more than once). In this phase, the experimental conditions differed only in decision granularity and/or the pedagogical policy used. Table 2.7 shows the probability principles included in each training problem, showing the order of the problem (p#), problem id (pid), and the number of times each principle needs to be applied.

### 2.2.4 Post-test

The post-test contains ten single-principle problems and ten multiple-principle problems. Among them, 14 are isomorphic to the pre-test (ten single- and four multiple-principle). The remaining six are complicated multiple-principle problems. Table 2.8 shows a pair of isomorphic problems (one in the pre-test and one in the post-test.) Table 2.9 shows the probability principles needed to solve each problem, showing the order of the problem in the first column and the number of times each principle needs to be applied.

**Table 2.7** Principles Included in Each Pyrenees Problem

p#	pid	CT	MET	IE	DMT	A2	A3	CIE	CP	TPT	BR
1	ex222	3		1							
2	exc137	1			1	1					
3	ex132a	3	2				1				
4	ex132	3	2			2	1				
5	ex152a	3		1		2	1				
6	ex152b	5	2		2		1				
7	ex152	6		1	3		1				
8	ex144	1								1	
9	ex212	1							5		1
10	ex242	1							2	1	1
11	ex252a	1									1
12	ex252	2						6			1
Total		30	6	3	6	5	6	6	7	2	4

### 2.2.5 Grading

The pre- and post-test problems were graded using three scoring rubrics: 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 twice in a double-blind manner by single or multiple experienced graders. Inconsistent grades will be checked again to resolve the discrepancy. The results presented in the following chapters 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].

**Table 2.8** An Example Pair of Isomorphic Problems in Pre- and Post-test

---

**A Pre-test Problem**

**Question:** Given  $p(A) = 0.7$ ,  $p(B) = 0.8$  and  $p(A \cap B) = 0.6$ , find  $p(\sim A \cap \sim B)$

**Answer:** 1) Apply the Addition Theorem for two events:

$$p(A \cup B) = p(A) + p(B) - p(A \cap B) = 0.7 + 0.8 - 0.6 = 0.9;$$

2) Apply the Complement Theorem:

$$p(\sim(A \cup B)) = 1 - p(A \cup B) = 1 - 0.9 = 0.1;$$

3) Apply the De Morgan's Law:

$$p(\sim A \cap \sim B) = p(\sim(A \cup B)) = 0.1.$$

---

**The Isomorphic Post-test 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:

$$p(A \cup B) = p(A) + p(B) - p(A \cap B), \quad p(A) = 0.13 + 0.03 - 0.06 = 0.1.$$

---

**Table 2.9** Principles Included in Each Post-test Problem

p#	CT	MET	IE	DMT	A2	A3	CIE	CP	TPT	BR
1			1							
2		2	2			1				
3	1			1	1					
4	1									
5								1		
6					1					
7									1	
8							1			
9								2	1	1
10		1								
11										1
12				1						
13							2			1
14	2		1							
15	1			1	1			1		
16	1									1
17						1				
18	1	2		1	1	1				
19	1	2			2	1				
20		2				1			3	

## CHAPTER

# 3

## GRANULARITY STUDIES

This Chapter describes our exploration of the impact of decision granularity on student learning. A series of four classroom studies were conducted in the Fall semester of 2014-2017 following the same 4-phase procedure as described in Chapter 2, using the same materials, and ITS. In Fall 2014, three types of granularity: 1) problem-level only (Prob-Only), 2) step-level only (Step-Only), and 3) both problem- and step-level (Both) as well as two baseline conditions: WE-only (WE) and PS-only (PS) were compared. Results showed that there was no significant difference between the five conditions in learning performance, but for time on task, we have: WE < Prob-Only < Step-Only, Both < PS.

The three following studies involved one or two granularity types. To further investigate how decision granularity may impact student learning, 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 on learning performance. Then, motivated by the aptitude treatment interaction (ATI) theory, we split students into different incoming competence groups and conducted a two-factor post-hoc analysis on granularity and incoming competence. Results showed that Prob-only can be more effective for Low students, Step-only can be more effective for High ones, and more importantly, Both can be effective for both Low and High students. For time on task, we have 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 impact depends on students' current knowledge level. The following of this Chapter describes the background, experiment setup, and results in detail.

### **3.1 Background - Prior Research on 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. (Recall that in FWE, all steps are pre-determined at the beginning of a problem while in CPS, step level decisions are made dynamically inside the problem.) 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 [McL08]. 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 [McL11]. 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 PS, and the former can take significantly less time than the latter. In another study, Van Gog et al. found that studying WEs can lead to better performance [VG11]. The study included four conditions, WE-only, WE-PS pairs, PS-WE pairs (problem-solving followed by an example), and PS-only. Results showed that the WE and WE-PS conditions significantly outperformed the PS-WE and PS-only conditions in learning performance, but no time on task difference was found among the four conditions.

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 [Kal10]. 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 [Ren02]. Students in the WE-FWE-PS condition studied and completed WEs, FWEs, and PS in a fixed order, while those in the WE-PS condition received the same problem 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 [Sch09]. 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 [Naj14]. 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 [Sal10]. The WE-FWE-PS condition gives WEs, FWEs, and PS 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 performance. In addition, adaptive policies can be more effective than fixed ones. However, all previous works focused on either the problem level or the 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 some fixed or some adaptive policies. Therefore, it is not clear whether it was the policy or granularity that impacted student learning. In four classroom studies, we directly compare the three granularity types and use random policies to factor out the impact of pedagogical policy.

## 3.2 Experiment Setup

### 3.2.1 Participants

The participants in all the studies were undergraduate students enrolled in the discrete math class offered by the computer science department at North Carolina State University. Most of them were majored in computer science and in the second year of their undergraduate program. 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.

### 3.2.2 Conditions

The studies compared three types of granularity (Prob-Only, Step-Only, and Both) with two baseline conditions: WE-Only (WE) and PS-Only (PS). Ordered by the amount of work students need to do (increasingly), the five conditions are:

- **WE-Only:** 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-Only:** where all problems are PSs.

### 3.2.3 Procedure and Measures

Students in all four studies went through four phases: 1) textbook, 2) pre-test, 3) training on the ITS, and 4) post-test, as shown in Table 3.1. During **textbook**, all students read a general description of each principle, reviewed some examples, and solved some training problems. The students then took a **pre-test**, which contained a total of 14 single- and multiple-principle 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 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. 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. Conditions differed only in decision granularity. The pre- and post-test were graded by a single experienced grader following the partial credit rubric in a double-blind manner. More details about the experimental procedure are described in Chapter 2.

**Table 3.1** Procedure of the Granularity Studies

Procedure	Conditions				
	WE	Prob	Step	Both	PS
Textbook	Read a probability textbook				
Pre-test	Complete 14 test questions (10 single- and 4 multiple-principle)				
ITS-training	WE-only	WE/PS	Elicit/Tell	WE/PS & Elicit/Tell	PS-only
Post-test	Complete 20 test questions (10 single- and 10 multiple-principle)				

After the studies, we performed a statistical analysis on students' learning performance and time on task. Our analysis on the pre-test showed that there are four very easy questions that almost every student got correct. To increase sensitivity, they were excluded from our subsequent analysis and so were the four corresponding isomorphic questions in the post-test. In the following of this Chapter, the pre- and post-test scores are calculated based on the remaining 10 and 16 problems, respectively. Specific learning performance measures are listed as follows.

- **Pre-test:** calculated based on the 10 pre-test questions;

- **Isomorphic Post-test:** calculated based on the 10 post-test questions that are isomorphic to the pre-test;
- **Full Post-test:** calculated based on the 16 post-test questions;
- **Adjusted Post-test:** calculated following the equation:  $adjusted = post - k * (pre - \overline{pre})$ , where  $k$  is the coefficient of the linear regression model  $post = k * pre + b$  generated by ANCOVA analysis, and  $\overline{pre}$  is the average pre-test score across all students.

The time students spent on the training task (time on task) was calculated based on an analysis of the system logs with idle time excluded. A time interval with no action for over 141 seconds was identified as idle. We got the threshold of 141 seconds based on the analysis of the Fall'14 data that 99% of the actions took less than 141 seconds.

### 3.2.4 High vs. Medium vs. Low Students

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 [Cro77; Sno91], we split students into multiple incoming competence groups based on their pre-test performance. Recall that our pre-test includes two types of problems: single-principle and multiple-principle, as shown in Table 2.5. Here, we argue that multiple-principle problems require more knowledge than single-principle problems to solve, and based on this argument, we split students.

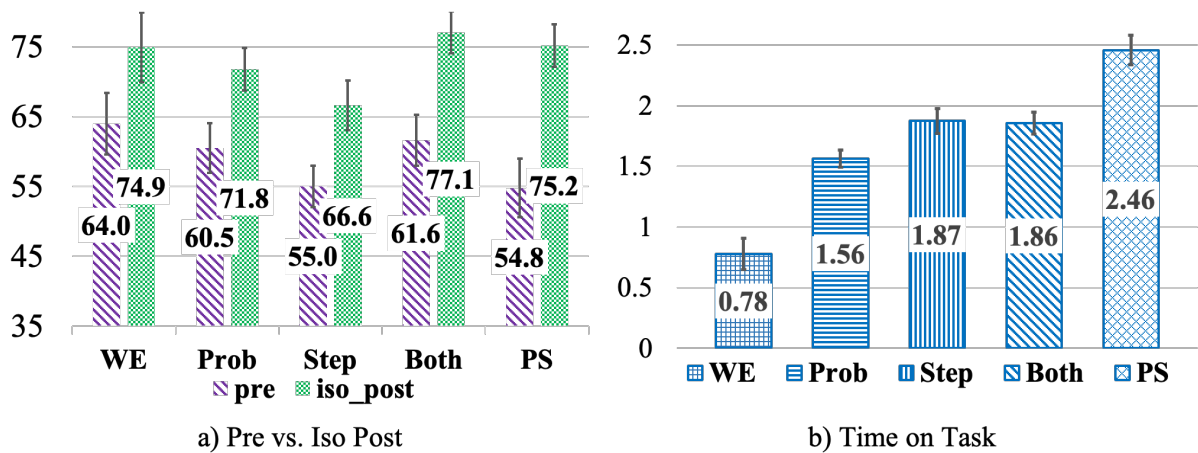
Learning in STEM domains often involves acquiring two types of knowledge: declarative knowledge, which includes facts that we know, such as domain principles and procedural knowledge, which specifies how to retrieve and use declarative knowledge to solve problems [And93]. Procedural knowledge often requires 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 problem mainly involves factors 1-3, while solving a multiple-principle problem involves all five. Thus, students must be able to solve single-principle problems before they can solve multiple-principle problems. Our data supported this point, showing that students who could solve a multiple-principle problem on the pre-test were also able to solve the single-principle problems that require one of the principles involved. Therefore, we argue that it is much easier for students to earn credit from single-principle problems than from multiple-principle problems. In the following, we refer to students who got a pre-test score that requires correctly solving all single-principle problems and at least one multiple-principle problem as High students ( $pre \geq 0.7$ ), those who got a score that requires correctly solving no more than the six single-principle problems as Low students ( $pre \leq 0.6$ ), and the rest as Medium ones ( $0.6 < pre < 0.7$ ).

**Table 3.2** Fall 2014 Empirical Study Results

Condition	Pre	Iso Post	Full Post	Adj Post	Time (hours)
WE(21)	64.0(20.3)	74.9(23.0)	63.6(23.0)	60.6(17.6)	.78(.59)
Prob(35)	60.5(21.0)	71.8(18.2)	59.6(20.6)	58.6(17.3)	1.56(.41)
Step(36)	55.0(17.8)	66.6(21.4)	54.3(19.5)	56.5(14.3)	1.87(.62)
Both(33)	61.6(21.0)	77.1(17.1)	64.7(18.9)	63.1(16.3)	1.86(.51)
PS(28)	54.8(22.2)	75.2(16.1)	62.3(17.7)	64.7(13.7)	2.46(.65)

### 3.3 Fall 2014 Empirical Study Results

In the Fall 2014 study, 266 students were randomly assigned to five conditions: 31 for WE<sup>1</sup>, 58 for Prob-Only, 59 for Step-Only, 59 for Both, and 59 for PS. Due to preparations for exams and the length of the experiment, 162 students completed the study. Nine students who performed perfectly on the pre-test or completed the study in groups were excluded from our subsequent statistical analysis. The remaining 153 students were distributed as follows: 21 for WE, 35 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 differ significantly by condition:  $\chi^2(4) = 4.57, p = 0.335$ .



**Figure 3.1** Students' Improvement and Time on Task

<sup>1</sup>Fewer students were assigned to the WE condition because another purpose of this study was to collect training data for later RL/HRL policy induction.

### 3.3.1 Analysis on the Impact of Granularity

**Incoming Competence:** A one-way ANOVA analysis on the pre-test showed that there was no significant difference among the five conditions:  $F(4, 148) = 1.14$ ,  $p = 0.340$ ,  $\eta = 0.030$ . This suggests that the random assignment indeed balanced students' incoming competence. Table 3.2 shows the mean and standard deviation (SD) of students' learning performance and total training time results across the five conditions. 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 (Full Post), adjusted post-test (Adj Post) (full-post test score adjusted by pre-test score 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 students' learning improvement, we compared their isomorphic post-test scores with their pre-test scores, as shown in Figure 3.1. 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 of test type:  $F(1, 148) = 85.51$ ,  $p < 0.0001$ ,  $\eta = 0.358$  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 than in the pre-test:  $F(1, 20) = 5.89$ ,  $p = 0.025$ ,  $\eta = 0.227$  for WE;  $F(1, 34) = 13.36$ ,  $p = 0.0009$ ,  $\eta = 0.282$  for Prob-Only;  $F(1, 35) = 15.50$ ,  $p = 0.0004$ ,  $\eta = 0.307$  for Step-Only;  $F(1, 32) = 19.23$ ,  $p = 0.0001$ ,  $\eta = 0.375$  for Both and  $F(1, 27) = 44.28$ ,  $p < 0.0001$ ,  $\eta = 0.621$  for PS. This showed 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 performed analysis on the full post-test score which has six additional multiple-principle problems. An ANCOVA analysis on the post-test using the pre-test score as a covariate showed no significant difference among the five conditions:  $F(4, 147) = 1.39$ ,  $p = 0.241$ ,  $\eta = 0.023$ .

**Time on Task:** A one-way ANOVA analysis showed a significant difference among the five conditions:  $F(4, 148) = 28.98$ ,  $p < 0.0001$ ,  $\eta = 0.439$ , as shown in Figure 3.1. 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 ( $p = 0.020$ ), Both ( $p = 0.033$ ), 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.

### 3.3.2 Analysis on Granularity and Incoming Competence

In order to investigate whether the impact of granularity differs for students with different incoming competence, we split them into High, Medium, and Low incoming competence groups based on their pre-test score and conducted a two-factor analysis. Since the baseline WE-only has fewer students and more importantly, our main focus is the impact of granularity, this analysis involved

only the three granularity conditions, resulting in a 3 (Prob vs. Step vs. Both) by 3 (High vs. Medium vs. Low) analysis.

**Incoming Competence:** As expected, one-way ANOVA showed that the High, Medium and Low groups differed significantly in pre-test:  $F(2, 101) = 210.26, p < 0.0001, \eta = 0.806$ . Subsequent contrast analysis revealed that High scored significantly higher than Medium:  $t(101) = 5.50, p < 0.0001, d = 2.67$  and Medium scored significantly higher than Low:  $t(101) = 8.04, p < 0.0001, d = 2.00$ . The first column in Table 3.3 shows the groups with the number of students in the parentheses. A Chi-square test showed that the size of the High, Medium, and Low groups does not differ significantly by conditions:  $\chi^2(4) = 2.98, p = 0.562$ . Fortunately, the random assignment balanced students' incoming competence among the three conditions and this balance persisted even after students were split into High, Medium, and Low groups. More specifically, no significant difference was found in pre-test among the three High groups, the three Medium groups, or the three Low groups.

**Learning Performance:** A two-way ANCOVA analysis on granularity and incoming competence on the full post-test using the pre-test score as a covariate showed no significant interaction effect or main effect. Subsequent contrast analysis revealed that for Low students, there is a trend that Both<sub>L</sub> scored higher than Step-Only<sub>L</sub>:  $t(94) = 1.94, p = 0.056, d = 0.70$ .

**Time on task:** A two-way ANOVA on granularity and incoming competence showed a main effect of granularity:  $F(2, 95) = 3.85, p = 0.025, \eta = 0.072$  in that the Prob-Only spent significantly less time than Step-Only:  $t(101) = -2.51, p = 0.014, d = 0.59$  and Both:  $t(101) = -2.31, p = 0.023, d = 0.64$ . Subsequent contrast analysis revealed that the difference mainly came from High students in that: Prob-Only<sub>H</sub> spent significantly less time than Step-Only<sub>H</sub>:  $t(95) = -2.76, p = 0.007, d = 1.34$  and Both<sub>H</sub>:  $t(95) = 2.20, p = 0.030, d = 1.08$ .

Overall, results from the Fall 2014 study showed that granularity can have an impact on students' time on task in that the Prob condition spent less time than the Step and Both condition. But most of the learning performance results were not significant. One possible reason is that the groups were small in size after the incoming competence split, and thus, there wasn't enough statistical power to get significant results. To increase statistical power, we included the data collected in three following studies and conducted a post-hoc analysis.

### 3.4 Post-hoc Analysis Results

The post-hoc analysis included four studies conducted in the Fall semester of 2014 to 2017. Students were combined according to 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 from 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. 67 students completed the study, and we excluded one student with a perfect pre-test score. The remaining students were distributed as follows:  $N = 31$  for Prob-Only

**Table 3.3** Results on High, Medium, and Low Students for Fall 14

Group	Pre	Iso Post	Full Post	Adj Post	Time (hours)
Prob-Only <sub>H</sub> (11)	86.2(6.6)	82.8(12.6)	71.8(16.5)	56.1(14.9)	1.41(.34)
Step-Only <sub>H</sub> (8)	80.0(8.0)	86.8(15.6)	76.9(15.4)	64.8(13.9)	2.09(.67)
Both <sub>H</sub> (13)	82.6(6.4)	86.3(13.6)	76.7(16.0)	63.1(15.4)	1.89(.50)
Prob-Only <sub>M</sub> (5)	63.6(1.7)	72.8(13.4)	59.8(16.8)	57.1(16.0)	1.61(.29)
Step-Only <sub>M</sub> (6)	64.7(1.7)	68.7(18.7)	55.9(12.4)	52.6(12.7)	2.07(.37)
Both <sub>M</sub> (6)	65.3(2.8)	74.0(16.3)	61.8(18.9)	58.2(19.3)	1.86(.43)
Prob-Only <sub>L</sub> (19)	44.9(11.8)	65.1(19.3)	52.5(21.1)	60.6(19.4)	1.64(.46)
Step-Only <sub>L</sub> (22)	43.3(9.9)	58.8(19.5)	45.6(15.6)	54.6(14.4)	1.74(.65)
Both <sub>L</sub> (14)	40.6(10.9)	69.9(17.5)	54.8(15.9)	65.3(16.6)	1.83(.57)

**Table 3.4** Fall 2015-2017 Empirical Studies Results

Year	Condition	Pre	Iso Post	Full Post	Adj Post	Time (hours)
2015	Prob(38)	73.7(18.9)	83.0(20.5)	74.1(22.1)	74.5(19.8)	1.99(.75)
	Step(34)	75.4(16.7)	90.0(11.6)	83.4(14.5)	83.0(13.3)	2.26(.48)
2016	Prob(31)	65.2(17.4)	81.8(17.4)	71.2(19.1)	69.6(18.3)	1.73(.66)
	Step(35)	60.7(21.5)	76.3(23.1)	63.5(21.9)	64.8(14.7)	1.88(.44)
2017	Both(55)	67.2(20.8)	83.5(20.0)	73.9(22.2)	73.9(16.6)	1.81(.53)

**Table 3.5** Post-hoc Analysis Results

Condition	Pre	Iso Post	Full Post	Adj Post	Time (hours)
Prob(104)	66.7(19.8)	78.9(19.3)	68.4(21.5)	67.3(18.8)	1.77(.65)
Step(105)	63.5(20.5)	77.4(21.5)	66.8(22.3)	67.8(15.6)	2.00(.55)
Both(88)	65.1(20.9)	81.1(19.2)	70.5(21.4)	70.5(16.8)	1.83(.52)

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 two baseline conditions (WE-only and PS-only) were not included in the 2015-2017 studies, the three granularity conditions were much larger in group size than the two baseline conditions. Thus, the post-hoc analysis mainly focuses on comparing the three types of granularity. The final post-hoc analysis data set included 297 students:  $N = 104$  for Prob-Only,  $N = 105$  for Step-Only, and  $N = 88$  for Both. Overall, a Chi-square test showed that

the participants' completion rates did not differ significantly by condition:  $\chi^2(2) = 0.653, p = 0.722$ . Table 3.4 shows the test scores and training time results for the 2015-2017 Fall studies.

### 3.4.1 Analysis on the Impact of Granularity

**Incoming Competence:** A one-way ANOVA analysis on the pre-test score showed no significant difference among the three conditions:  $F(2, 294) = 0.65, p = 0.522, \eta = 0.004$ . This suggests that the three conditions were balanced with respect to incoming competence across the four years. Table 3.5 shows the learning performance and time on task results for the post-hoc analysis.

**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 of test type  $F(1, 296) = 174.04, p < 0.0001, \eta = 0.370$  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:  $F(1, 103) = 42.61, p < 0.0001, \eta = 0.293$  for Prob-Only;  $F(1, 104) = 71.32, p < 0.0001, \eta = 0.407$  for Step-Only and  $F(1, 87) = 64.59, p < 0.0001, \eta = 0.426$  for Both. The results suggested that our tutor was 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 between the three conditions:  $F(2, 293) = 0.93, p = 0.395, \eta = 0.004$ .

**Time on Task:** A one-way ANOVA analysis showed a significant difference among the three conditions:  $F(2, 294) = 4.53, p = 0.012, \eta = 0.030$ . Subsequent contrast analysis revealed that Prob-Only and Both spent significantly less time than Step-Only:  $t(294) = -2.90, p = 0.004, d = 0.39$  and  $t(294) = -2.08, p = 0.038, d = 0.32$ , respectively.

### 3.4.2 Analysis on Granularity and Incoming Competence

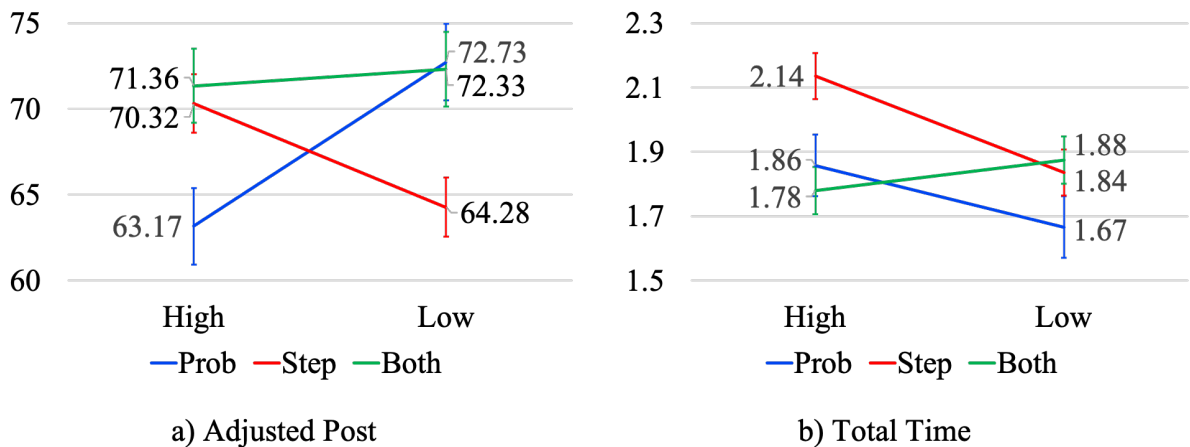
As stated previously, we split students into different incoming competence groups to investigate whether the impact of granularity is contingent on students' competence level. A two-factor analysis on granularity and incoming competence was performed to answer this question.

**Incoming Competence:** As expected, in the pre-test, the High group scored significantly higher than the Medium group:  $t(178) = 11.36, p < 0.0001, d = 2.19$  and the Medium group scored significantly higher than the Low group:  $t(148) = 9.12, p < 0.0001, d = 1.80$ . The first column in Table 3.6 shows the groups with the number of students in the parentheses. As expected, the three High and the three Low groups are of reasonable sizes; while the three Medium groups are small due to its small score range. A Chi-square test showed that the size of the High, Medium, and Low groups did not differ significantly by conditions:  $\chi^2(4) = 1.96, p = 0.744$ . In addition, there was no significant difference in pre-test between the three High groups, the three Medium groups, or the three Low groups.

**Learning Performance:** We then conducted  $3 \times 3$  analyses on the factors of granularity and incoming competence. Table 3.6 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 covariate showed a significant interaction effect:  $F(4, 287) = 3.44, p = 0.009, \eta = 0.028$ , as shown

**Table 3.6** Post-hoc Analysis Results on High, Medium and Low students

Group	Pre	Iso Post	Full Post	Adj Post	Time (hours)
Prob-Only <sub>H</sub> (55)	82.2(8.7)	84.6(13.8)	74.5(17.7)	63.2(16.6)	1.86(.71)
Step-Only <sub>H</sub> (47)	82.7(7.7)	90.8(8.9)	81.9(12.6)	70.3(11.7)	2.14(.49)
Both <sub>H</sub> (45)	82.1(7.2)	90.4(11.0)	82.6(15.5)	71.4(14.6)	1.78(.49)
Prob-Only <sub>M</sub> (10)	65.2(2.1)	81.3(14.8)	68.8(17.3)	68.7(16.3)	1.69(.41)
Step-Only <sub>M</sub> (12)	64.9(2.2)	81.8(19.0)	71.5(19.1)	71.7(19.1)	2.10(.47)
Both <sub>M</sub> (11)	65.2(2.5)	73.6(21.6)	61.6(21.6)	61.6(21.8)	1.89(.44)
Prob-Only <sub>L</sub> (39)	45.3(11.8)	70.1(23.7)	59.6(24.7)	72.7(21.2)	1.67(.59)
Step-Only <sub>L</sub> (46)	43.6(10.9)	62.6(22.1)	50.0(19.2)	64.3(17.5)	1.84(.58)
Both <sub>L</sub> (32)	41.2(11.9)	70.5(21.1)	56.5(18.8)	72.3(17.4)	1.88(.58)



**Figure 3.2** Adjusted Post-test and Time on Task for Post-hoc Analysis on Granularity and Incoming Competence

in Figure 3.2.a. However, there was no significant main effect of granularity or incoming competence. Subsequent contrast analysis revealed that for High students, the Step-Only<sub>H</sub> group and the Both<sub>H</sub> group scored significantly higher than the Prob-Only<sub>H</sub> group:  $t(287) = 2.12, p = 0.035, d = 0.49$  and  $t(287) = 2.42, p = 0.016, d = 0.52$ , respectively, but there was no significant difference between Step-Only<sub>H</sub> and Both<sub>H</sub>:  $t(287) = 0.32, p = 0.749, d = 0.08$ ; for Low students, the Prob-Only<sub>L</sub> group and the Both<sub>L</sub> group scored significantly higher than the Step-Only<sub>L</sub> group:  $t(287) = 2.23, p = 0.026, d = 0.44$  and  $t(287) = 2.15, p = 0.032, d = 0.46$ , respectively, but there was no significant difference between Prob-Only<sub>L</sub> and Both<sub>L</sub>:  $t(287) = 0.04, p = 0.969, d = 0.02$ ; while for Medium students, none of the contrasts was significant:  $t(287) = -0.42, p = 0.677, d = 0.17$  for Prob<sub>M</sub> vs. Step<sub>M</sub>,  $t(287) = 0.96, p = 0.335, d = 0.37$  for Prob<sub>M</sub> vs. Both<sub>M</sub> and  $t(287) = 1.44, p = 0.151, d = 0.49$  for Step<sub>M</sub> vs. Both<sub>M</sub>.

The results suggest that for High students, Step-Only and Both are more effective than Prob-Only, whereas for Low students, Prob-Only and Both are more effective than Step-Only. In other words, problem level decisions can be more effective for students with a lower incoming competence, step level decisions can be more effective for students with a higher incoming competence, and both level decisions can be effective for students that have either a lower or higher incoming competence.

**Time on Task:** A two-way ANOVA analysis on granularity and incoming competence showed a significant main effect of granularity:  $F(2, 288) = 4.60, p = 0.011, \eta = 0.030$  in that Prob-Only and Both spent less time than Step-Only:  $t(294) = -2.90, p = 0.004, d = 0.39$  and  $t(294) = -2.08, p = 0.038, d = 0.32$ , respectively, as shown in Figure 3.2.b. Subsequent contrast analysis revealed that for High students: Prob-Only<sub>H</sub> and Both<sub>H</sub> spent less time than Step-Only<sub>H</sub>:  $t(288) = -2.46, p = 0.015, d = 0.45$  and  $t(288) = -3.00, p = 0.003, d = 0.73$ , respectively, but there was no significant difference between Prob-Only<sub>H</sub> and Both<sub>H</sub>:  $t(288) = 0.68, p = 0.496, d = 0.13$ . While for Medium and Low students, none of the contrast was significant:  $t(288) = -1.68, p = 0.094, d = 0.92$  for Prob<sub>M</sub> vs. Step<sub>M</sub>,  $t(288) = -0.80, p = 0.423, d = 0.47$  for Prob<sub>M</sub> vs. Both<sub>M</sub>,  $t(288) = 0.88, p = 0.378, d = 0.46$  for Step<sub>M</sub> vs. Both<sub>M</sub>,  $t(288) = -1.37, p = 0.172, d = 0.29$  for Prob<sub>L</sub> vs. Step<sub>L</sub>,  $t(288) = -1.54, p = 0.126, d = 0.36$  for Prob<sub>L</sub> vs. Both<sub>L</sub>, and  $t(288) = -0.30, p = 0.767, d = 0.07$  for Step<sub>L</sub> vs. Both<sub>L</sub>.

### 3.5 Conclusion and Discussion of Chapter 3

In four studies, we explored the impact of decision granularity on student learning by comparing three granularity types: 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 showed that Prob-Only can be more effective for Low students, Step-Only can be more effective for High ones, and Both can be effective for both Low and High 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 knowledge level.

One possible reason that decision granularity can have an impact on student learning 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, students learn from co-constructing the solution with the tutor. As a result, the three granularity types may involve students in different learning activities. More specifically, from students' perspective, Prob-Only switches between learning by observing (WE) and learning by doing (PS); Step-Only provides opportunities to co-construct

the solution with the tutor; while Both involve all three of them. Each of the activities may involve different underlying cognitive processes and, in turn, lead to different impacts. However, this is only our hypothesis; more investigation is needed to understand what exact cognitive processes are involved in ITS provided WE, PS, and CPS.

Both learning performance and time on task results suggest that Step-Only might be more difficult than Prob-Only. Our learning performance results showed that Step-Only benefited High students more, while Prob-Only benefited Low students more. Applying the Zone of Proximal Development (ZPD) theory [Cha03], 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, 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 efforts are 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 again only our hypothesis; more research is needed to investigate the exact mental work required by different granularity types.

Our results suggest that Both can be a more robust instructional intervention than Prob-Only and Step-Only, and this may also be explained by our hypothesis that WE, PS, and CPS involve different learning mechanisms. More specifically, using Both-level decisions allows students to experience all of them, and thus, it is less likely that the intervention is ineffective during the entire training.

Overall, the granularity studies showed that decision granularity indeed can have an impact on student learning, and its impact may depend on students' competence level. This suggests that decision granularity should be considered in pedagogical decision-making, and the decisions should be made in an adaptive manner. Though, there is no well-established theory that specifies how problem-level WE/PS and step-level Elicit/Tell can be best used. On the other hand, in recent years, reinforcement learning has been shown to be powerful in handling complex decision-making tasks. Thus, in the RL and HRL studies, we applied RL and HRL to induce pedagogical policies with decision granularity taken into account.

# DATA COLLECTION AND PROCESSING FOR REINFORCEMENT LEARNING

Motivated by the results from the granularity study that decision granularity can have an impact on student learning, we then applied reinforcement learning (RL) and hierarchical reinforcement learning (HRL) to induce pedagogical policies that make decisions at different levels of granularity. This chapter describes how the data were collected and processed for RL/HRL pedagogical policy induction.

## 4.1 Markov Decision Processes

Prior research applying RL to induce pedagogical policies often formalize 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  $\langle S, A, T, R \rangle$ , where

- $S = \{S_1, S_2, \dots, S_n\}$  denotes the state space;
- $A = \{A_1, A_2, \dots, A_m\}$  represents a set of agent's possible actions;
- $T : S \times A \times S \rightarrow [0, 1]$  is a transition probability function indicating the probability of transiting from state  $S_i$  to state  $S_j$  by taking an action  $a$ ;

- $R : S \times A \times S \rightarrow \mathbb{R}$  is a reward function specifying the rewards the agent will receive if it takes action  $a$  in state  $s_i$  and the environment state transits to  $s_j$ .

Additionally, the policy, defined as  $\pi : S \rightarrow A$ , is a mapping from state  $S$  into action  $A$  with the goal of maximizing the expected cumulative reward.

In pedagogical policy induction, states  $S$  are often represented by features that describe the students' learning state or the context of the learning environment, such as the percentage of correct attempts a student has made so far. 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  $\langle S, A, R \rangle$  has been defined, the transition probability function  $T$  is estimated from the training corpus. In order to apply RL for pedagogical policy induction, we transferred student-system interaction logs into trajectories in the following form.

$$S_1 \xrightarrow{A_1, R_1} S_2 \xrightarrow{A_2, R_2} S_3 \xrightarrow{A_3, R_3} \dots \rightarrow S_N$$

Where  $S_i \xrightarrow{A_i, R_i} S_{i+1}$  means that the tutor executed action  $A_i$  and received reward  $R_i$  in state  $S_i$ , and then the environment transferred to the next state  $S_{i+1}$ . In general, the reward can be divided into two categories: immediate rewards are received during the state transition, and delayed rewards are only available after the task is complete. The following of this chapter describes the data collection and processing details.

## 4.2 Training Corpus

In my dissertation, we applied offline RL approaches to induced pedagogical policies from a pre-collected exploratory corpus. As a result, the quality of the policies is heavily dependent on the quality of the exploratory data. Ideally, the exploratory corpus should cover all possible state-action combinations so that the environment is fully explored. However, due to the high cost of collecting student data, it is infeasible to fully explore all state-action combinations. To broaden the exploration, we employed random policies to explore the environment. Additionally, we also included RL-guided exploration data in our exploratory corpus. Unlike random exploration trying to cover as many states as possible, RL-guided exploration visits certain states more often and thus provides better information for them, which also helps improving policy quality.

Our training data were collected by letting students work on our ITS. All the data were collected following the same four-phase procedure using the same materials as described in section 2. In the ITS training section, some students followed random policies, and some students followed RL-induced policies. Students who did not complete the study or performed perfectly in the pre-test were excluded. The entire training data set includes 1118 students. Each of them spent around 2 hours in training and completed around 400 steps (since all students were given the same 12 problems in the same order and were required to follow the same problem solving strategy, most of them completed around 400 steps). Information for the resulting training data set is shown in Table 4.1, showing the semester, tutor decisions, policy, and the number of students included.

**Table 4.1** Training Data Information

Semester	Tutor Decision	Policy	N Students
14'Fall	WE/PS	Random	38
	Elicit/Tell	Random	37
	WE/PS/ & Elicit/Tell	Random	34
15'Spring	WE/PS	Random	37
	Elicit/Tell	Random	41
	WE/PS/ & Elicit/Tell	Random	39
15'Fall	WE/PS	Random	38
	Elicit/Tell	Random	34
	WE/PS	Student Decision	70
	Elicit/Tell	Student Decision	59
16'Fall	WE/PS	Random	31
	Elicit/Tell	Random	35
	WE/PS	Tabular RL1	48
	Elicit/Tell	Tabular RL2a	48
	Elicit/Tell	Tabular RL2b	42
17'Spring	Elicit/Tell	Random	28
	Elicit/Tell	Tabular RL3	103
17'Fall	WE/PS/ & Elicit/Tell	Random	56
	WE/PS/ & Elicit/Tell	POMDP-Tabular RL	77
	Elicit/Tell	POMDP	68
18'Spring	WE/PS/ & Elicit/Tell	Random	74
	WE/PS/ & Elicit/Tell	GP HRL1	81

### 4.3 Properties of Steps

In our ITS, each step contains a series of system actions such as giving tutor instruction or feedback, receiving students' responses, or presenting an equation. The log files record each of the system actions in chronological order. On the other hand, our pedagogical decisions are made at a more coarse granularity for the step or problem level. In order to align state representation with decision granularity, we aggregate the system information at the problem- or step-level granularity and

define some basic properties based on the aggregation. Specific properties are described in the following.

### 4.3.1 Step Duration

For each of the step, there are two rows in the log indicating its start and end. The duration of a step is defined as the interval between its end and start time. For example the log row “Thu Nov 29 20:54:58 2018’, substep, ‘<parameter>begin point; Step mode = problem; Procedure = select\_main\_step(student,\_G14255); </parameter>’ ” indicates that at “20:54:58” the step “select\_main\_step” started. Similarly, there is a row in the log indicating this step ended at “20:55:29”. In this case, the duration of the step would be “20:55:29” - “20:54:58” = 31s.

### 4.3.2 Step Correctness

Recall that our ITS requires the student to provide the correct answer for each elicit step to move on. Though in feature extraction, we count an elicit step as correct if the student gives the correct answer on the first attempt. A hint request does not count as an attempt, but if the student requested more than two hints, the step will be counted as incorrect. This is because the first two hints give only general assistance, such as encouraging the student to think carefully, but the third hint usually gives substantial help that can make the step significantly easier, such as eliminating some choices.

### 4.3.3 Knowledge Components (KCs)

Each step requires applying certain knowledge to solve, such as determining the current problem solving phase, selecting a principle, or applying a principle (refer to section 2.1.2 for more details about the problem solving procedure). Accordingly, we define a knowledge component as the specific knowledge required to solve a step (only the knowledge that is the focus of the training counts). More specifically, the ITS requires students to solve five types of steps, “select main step”, “specify sought”, “select target variable”, “select principle” and “apply principle”. Since the first three do not involve probability principles, we define the KCs needed by them as “select\_main\_step”, “specify\_sought”, “select\_target\_variable”, respectively. For the last two types, the KC is defined as the combination of the step and the probability principle needed. For example, if the step requires the student to select the principle “Addition Theorem”, then the corresponding KC will be “select\_principle\_addition\_theorem”. The ITS teaches 10 probability principles, and combining them with the steps of “select principle” and “apply principle” results in 20 KCs. In sum, our training data has 23 KCs in total.

### 4.3.4 Other Elements

- **Hint count:** A click of the help button in an elicit step counts as a request. Clicking the help button in tell steps functions as submitting the “go on” message and does not count as a hint request.

- **Session:** A session is defined as a period of consecutive time that the student is working on the training. The session ends if the student has not submitted anything for more than 6 hours.
- **Tutor words:** The total number of words the tutor used in its messages to the student, including instructions, questions, feedback, and hints. An equation is counted as one word.
- **Tutor concepts:** The total number of probability concepts the tutor mentioned in its messages to the student. Each word in the following list is considered as a concept: event, events, conditional, independent, mutually, mutual, exclusive, exhausted, quantity, total, Addition, Complement, De Morgan, disjoint, Bayes.

## 4.4 Environment Features

Since there is no existing theory that clearly specifies what information should be considered in pedagogical decision making, we extracted all the information that may be relevant to student learning (based on our understanding) to represent the learning environment state. Considering the step properties described above, we extracted a total number of 142 features in the following 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*;
- **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*;
- **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*;
- **Performance (57):** information about the student's performance so far, such as the percentage of correct elicit steps, *pctCorrect*;
- **Hints (11):** information about the student's hint usage, such as the total number of hints requested, *nHint*.

The following of this section describes each of the features in detail.

### 4.4.1 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, irrespective of the KC involved. For example,  $\text{tellsSinceElicit} = 2$  means that the student has received two tells since the last elicit. This feature reflects how proactive the tutor currently is in solving problems. The more consecutive tells the tutor gives, the more proactive it is.
- **ntellsSinceElicitKC**: The number of tells the student has received since the last elicit for the current KC. Similarly, it reflects how proactive the tutor currently is on the current KC.
- **nElicitSinceTell**: The number of elicits the student has received since the last tell, irrespective of the KC involved. This feature reflects how active a student currently is. The more elicits the student has received, the more active the student is.
- **nElicitSinceTellKC**: The number of elicits the student has received since the last tell 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, irrespective of the KC involved. The feature describes how active the student has been so far. The higher the percentage, the more active the student has been.
- **pctElicitKC**: The total number of elicit steps divided by the total number of steps the students have received so far for the current KC. This feature describes how active the student has been so far for the current KC.
- **pctElicitSession**: The total number of elicit steps divided by the total number of steps the students have received so far for the current session. Similarly, this feature describes the degree of activity for the current session.
- **pctElicitKCSession**: The total number of elicit steps divided by the total number of steps the students have received so far for the current KC and the current session.
- **nTellSession**: The total number of tells the student has received so far in the current session. This feature describes how much work the tutor has done for the student so far in the current session. The more work the tutor has done, the more opportunities the student has got to view the tutor's solution and explanation.
- **nTellKCSession**: The total number of tells the student has received so far for the current KC in the current session.

#### 4.4.2 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. This feature reflects how fresh the student's knowledge of the current KC is. High "durationKCBetweenDecision" means that the tutor has not mentioned the KC recently, so the student's memory about the current KC may have become vague.
- **timeInSession:** The time elapsed since the start of the current session. This feature may be interpreted as the student's fatigue over time.
- **timeBetweenSession:** The time elapsed between the end of the previous session and the beginning of the current one. This feature reflects how likely a student has forgotten what they have learned in previous sessions.
- **timeOnCurrentProblem:** The time elapsed since the start of the current problem. This feature reflects the student's fatigue level on the current problem.
- **timeOnLastStepKCElicit:** The time that the student spent on the last elicit step with the same KC as the current step. This feature may be interpreted as the student's proficiency in applying the current KC. Note that in each elicit step, the student must give the correct answer to move forward. Thus, the faster they give the correct answer, the more proficient they are in applying the current KC.

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 steps that were the focus of the training (the same setting applies to all following temporal features). This feature reflects the amount of effort the student has put on learning.
- **timeOnTutoringTell:** The total time the student has spent on tells. This feature reflects the amount of effort the student has put on studying the tutor's solutions.
- **timeOnTutoringElicit:** The total time the student has spent on Elicits. This feature reflects the amount of effort the student has put on solving problems.
- **timeOnTutoringKC:** The total time the student has spent on the current KC. This feature reflects the amount of effort the student has put on the current KC.
- **timeOnTutoringKCTell:** The total time the student has spent on the current KC with tell. This feature reflects the amount of effort the student has put on studying the tutor's solutions for the current KC.

- **timeOnTutoringKCElicit:** The total time the student has spent on the current KC with elicit. This feature reflects the amount of effort the student has put on solving problems for the current KC.
- **timeOnTutoringSession:** The total time the student has spent on learning for the current session. This feature reflects the total amount of effort the student has put on learning in the current session.
- **timeOnTutoringSessionTell:** The total time the student has spent on studying tell steps for the current session. This feature reflects the total effort the student has put on studying the tutor's solutions in the current session.
- **timeOnTutoringSessionElicit:** The total time the student has spent on solving elicit steps for the current session. This feature reflects the total effort the student has put on solving problems in the current session.
- **timeOnTutoringProblem:** The total time the student has spent on the current problem. This feature reflects the amount of effort the student has put on studying/solving the current problem.
- **timeOnTutoringProblemTell:** The total time the student has spent on tell steps for the current problem. This feature reflects the amount of effort the student has put on studying the tutor's solutions for the current problem.
- **timeOnTutoringProblemElicit:** The total time the student has spent on elicit steps for the current problem. This feature reflects the amount of effort the student has put on solving elicit steps for the current problem.

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

- **avgTimeOnStep:** The average time the student spent on each step. This feature reflects the student's speed on completing the training.
- **avgTimeOnStepTell:** The average time the student spent on each tell step. This feature reflects the student's speed on studying the tutor's solutions.
- **avgTimeOnStepElicit:** The average time the student spent on each elicit step. This feature reflects the student's speed on problem solving.
- **avgTimeOnStepKC:** The average time the student spent on each step for the current KC. This feature reflects the student's speed on studying the current KC.
- **avgTimeOnStepKCTell:** The average time the student spent on each tell step for the current KC. This feature reflects the student's speed on studying the tutor's solutions for the current KC.

- **avgTimeOnStepKCElicit:** The average time the student spent on each elicit step for the current KC. This feature reflects the student's speed on solving the steps that require applying the current KC.
- **avgTimeOnStepSession:** The average time the student spent on each step for the current session. This feature reflects the student's recent working speed.
- **avgTimeOnStepSessionTell:** The average time the student spent on tell steps for the current session. This feature reflects the student's recent speed on studying the tutor's solutions.
- **avgTimeOnStepSessionElicit:** The average time the student spent on elicit steps for the current session. This feature reflects the student's recent problem solving speed.
- **avgTimeOnStepProblem:** The average time the student spent on each step for the current problem. This feature reflects the student's working speed in the current problem.
- **avgTimeOnStepProblemTell:** The average time the student spent on each tell step for the current problem. This feature reflects the student's speed on studying the tutor's solutions in the current problem.
- **avgTimeOnStepProblemElicit:** The average time the student spent on each elicit step for the current problem. This feature reflects the student's speed on problem solving in the current problem.

#### 4.4.3 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. This feature reflects the student's overall progress.
- **stepOrderingSession:** The total number of steps the student has received for the current session. This feature reflects the amount of progress the student has made in the current session.
- **stepOrderingPb:** The total number of steps the student has received for the current problem. This feature reflects the student's progress in the current problem.
- **nKCs:** The number of steps the student has completed for the current KC. This feature reflects the student's familiarity level with the current KC.
- **nKCsAsElicit:** The number of elicit steps the student has completed for the current KC. This feature reflects the amount of practice the student has done for the current KC.

- **nKCsSession**: The number of steps the student has completed for the current KC in the current session. This feature reflects the student's recent familiarity level with the current KC.
- **nKCsSessionElicit**: The number of elicit steps the student has completed for the current KC in the current session. This feature reflects the amount of practice the student has done recently for the current KC.

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 (8th), the value is 1, and for the rest, the value is 0. This feature reflects how well a student might get used to the tutoring system.
- **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. This feature reflects whether the problem is easy to solve.
- **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. If a problem is significantly harder than its predecessors, the student may need to put more effort into studying it.
- **performanceDifficulty**: Students' average performance on the current KC (calculated based on our historical data). More specifically  $\frac{\text{correct elicits}}{\text{total elicits}}$  across all students. This feature reflects the difficulty of the current KC from previous students' performance.
- **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]). More specifically, for a given equation, each probability event is counted 1 point, and each condition probability event is count 2 points. The difficulty of the equation is the sum of the points. For example, the equation for the Conditional Probability Theorem ( $p(A \cap B) = p(A|B) \times p(B)$ ) includes three events:  $p(A \cap B)$  (1 point),  $p(A|B)$  (2 points), and  $p(B)$  (1 point). Therefore, its difficulty is 4. To balance the difficulty among principles, we adjusted the difficulty for three principles that have a long equation: Addition Theorem for Three Events ( $8-2=6$ ), Total Probability Theorem ( $10-1=9$ ), and Bayes' Rule ( $15-3=12$ ). More specifically, the adjustment ignores the target event on the left-hand side of the equation and deduct one more point for non-conditional probability theorems. For example, Addition Theorem for Three Events has 8 events in its equation:  $p(A \cup B \cup C) = p(A) + p(B) + p(C) - p(A \cap B) - p(A \cap C) - p(B \cap C) + p(A \cap B \cap C)$ . The adjustment ignores  $p(A \cup B \cup C)$  (-1) and deducts one more because it is not a conditional probability theorem (-1), resulting in  $8-2=6$ .
- **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 prob-

ability principle the value is 0. This feature describes the category of knowledge (probability axioms vs. conditional probability) needed to solve the current step.

- **problemDifficulty:** The difficulty of the current problem, which is the sum of the difficulty of the principles (defined by “principleDifficulty”) needed to solve the problem. For example, the problem “exc137” requires an Addition Theorem for two events (4 points), a De Morgan’s Theorem (2), and a Complement Theorem (2). Thus, its difficulty is  $4+2+2=8$ . If a principle needs to be applied multiple times, its difficulty will also be counted multiple times accordingly.
- **problemComplexity:** The value of this feature is determined by the number of principle applications needed to solve the current problem. For problems that require at most 2 principle applications (the eighth and eleventh problem), the value is 2. For problems that require 3-6 principle applications, the value is 3 (the second, third, ninth, and tenth problem), and for those requires more than 7 principle applications, the value is 4. Note that we also gave the first problem is a complexity of 2, because the main focus of it was to get the student familiar with the interface.
- **problemCategory:** If the problem does not require any conditional probability principle to solve, the value is 0; otherwise, the value is 1. This feature describes the category of the problem.

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). This feature reflects the complexity of the problem.
- **nDistinctPrincipleInSession:** The total number of distinct principles appeared in the current session. This feature measures the number of principles the student has encountered in the current session.
- **nPrincipleInSession:** The total number of principles appeared in the current session. This feature measures the number of principles the student has studied 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. This feature reflects the amount of probability-related information the tutor has shown.
- **tutAverageConcepts:** The average number of probability concepts the tutor has mentioned in each step. This feature reflects the amount of probability-related information the tutor conveyed in each step.

- **tutAverageConceptsSession:** The average number of probability concepts the tutor has mentioned in each step in the current session. This feature reflects the amount of probability-related information the tutor conveyed in each step recently.
- **tutConceptsToWords:** The number of probability concepts the tutor has mentioned, divided by the total number of words the tutor has used so far. This feature reflects how often the tutor has mentioned probability concepts.
- **tutConceptsToWordsSession:** The number of probability concepts the tutor has mentioned, divided by the total number of words the tutor has used so far in the current session. This feature reflects how often the tutor has mentioned probability concepts recently.
- **tutAverageWords:** The average number of words the tutor used in each step. This feature reflects how verbose the tutor is overall.
- **tutAverageWordsSession:** The average number of words the tutor used in each step in the current session. This feature reflects how verbose the tutor recently is.
- **tutAverageWordsElicit:** The average number of words the tutor used in each elicit step. This feature reflects how well students performed on elicit steps. The better they perform, the fewer words the tutor needs to say.
- **tutAverageWordsSessionElicit:** The average number of words the tutor used in each elicit step in the current session. This feature reflects how well the student performed on elicit steps in 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 received. Select- and apply- principle are considered as quantitative steps because they require using probability equations for inference. This feature measures how quantitative current training is.

The following six features describe the number of each probability axioms 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.

#### 4.4.4 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. This feature reflects the student's overall competence when only elicits are counted as learning opportunities.
- **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 with the following equation  $\frac{tells + correct\ elicits}{steps}$ . This feature reflects the student's overall competence, when both elicits and tells are counted as learning opportunities.
- **nCorrectKC:** The total number of elicit steps the student has correctly solved so far for the current KC. This feature reflects the student's overall competence on the current KC.
- **nIncorrectKC:** The total number of elicit steps the student failed to solve so far on the current KC. This feature reflects the student's overall incompetence on the current KC.
- **pctCorrectKC:** The number of elicit steps the student has correctly solved, divided by the total number of elicit steps the student has received so far on the current KC. This feature reflects the student's overall competence on the current KC when only elicits are counted as learning opportunities.
- **pctOverallCorrectKC:** Calculated as  $\frac{tells + correct\ elicits}{steps}$  for the current KC. This feature reflects the student's overall competence on the current KC, when both elicits and tells are counted as learning opportunities.
- **nCorrectKCSession:** The total number of elicit steps the student has correctly solved on the current KC in the current session. This feature reflects the student's competence on the current KC in the current session when only elicits are counted as learning opportunities.

- **nIncorrectKCSession:** The total number of elicit steps the student failed to solve (on the first attempt) for the current KC in the current session. This feature reflects the student's incompetence on the current KC in the current session.
- **pctCorrectSession:** 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 in the current session. This feature reflects the student's competence in the current session when only elicits are counted as learning opportunities.
- **pctCorrectKCSession:** 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 on the current KC in the current session. This feature reflects the student's competence on the current KC in the current session when only elicits are counted as learning opportunities.
- **pctOverallCorrectSession:** Calculated as  $\frac{\text{tells} + \text{correct elicits}}{\text{steps}}$  for the current session. This feature reflects the student's overall competence in the current session, when both elicits and tells are counted as learning opportunities.
- **pctOverallCorrectKCSession:** Calculated as  $\frac{\text{tells} + \text{correct elicits}}{\text{steps}}$  on the current KC for the current session. This feature reflects the student's competence on the current KC for the current session, when both elicits and tells are counted as learning opportunities.

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

- **nStepSinceLastWrong:** The number of steps the student has completed since the last wrong elicit step. This feature reflects the student's recent competence when both elicits and tells are counted as learning opportunities.
- **nStepSinceLastWrongKC:** The number of steps the student has completed since the last wrong elicit step on the current KC. This feature reflects the student's recent competence on the current KC when both elicits and tells are counted as learning opportunities.
- **nTellsSinceLastWrong:** The number of tell steps the student has received since the last wrong elicit step. This feature reflects the opportunities the student has received to learn from tell steps since the last time a wrong answer was given to an elicit step (irrespective of the KC involved).
- **nTellsSinceLastWrongKC:** The number of tell steps the student has received since the last wrong elicit step on the current KC. This feature reflects the opportunities the student has received to learn from tell steps since the last time a wrong answer was given to an elicit step on the current KC.

- **nStepSinceLastWrongSession:** The number of steps the student has completed since the last wrong elicit step in the current session. This feature reflects the student's recent competence in the current session when both elicits and tells are counted as learning opportunities.
- **nStepSinceLastWrongKCSession:** The number of steps the student has completed since the last wrong elicit step on the current KC in the current session. This feature reflects the student's recent competence on the current KC in the current session when both elicits and tells are counted as learning opportunities.
- **nTellsSinceLastWrongSession:** The number of tell steps the student has received since the last wrong elicit step in the current session. This feature reflects the opportunities the student has received to learn from tell steps since the last time a wrong answer was given to an elicit step in the current session (irrespective of the KC involved).
- **nTellsSinceLastWrongKCSession:** The number of tell steps the student has received since the last wrong elicit step on the current KC in the current session. This feature reflects the opportunities the student has received to learn from tell steps since the last time a wrong answer was given to an elicit step on the current KC in the current session.
- **timeSinceLastWrongStepKC:** The time elapsed since the last wrong elicit step on the current KC. This feature reflects how long it has been since the last time a wrong answer was given to an elicit step on the current KC.
- **nCorrectElicitsSinceLastWrong:** The number of elicit steps the student has successfully solved since the last wrong elicit step. This feature reflects the student's recent competence in solving elicit steps.
- **nCorrectElicitsSinceLastWrongKC:** The number of elicit steps the student has successfully solved since the last wrong elicit step on the current KC. This feature reflects the student's recent competence in solving elicit steps for the current KC.
- **nCorrectElicitsSinceLastWrongKCSession:** The number of elicit steps the student has successfully solved since the last wrong elicit step on the current KC in the current session. This feature reflects the student's recent competence in solving elicit steps 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:** The number of elicit steps the student has correctly solved, divided by the total number of elicit steps the student has received so far on the steps that require a probability principle. This feature reflects the student's overall competence in solving probability-principle-involved steps when only elicits are counted as learning opportunities.

- **pctCorrectPrinSession:** The number of elicit steps the student has correctly solved, divided by the total number of elicit steps the student has received so far on the steps that require a probability principle in the current session. This feature reflects the student's overall competence in solving probability-principle-involved steps when only elicits are counted as learning opportunities in the current session.
- **nStepSinceLastWrongPrin:** The number of steps the student has completed since the last wrong elicit step for the steps that require a probability principle. This feature reflects the student's recent competence on probability-principle-involved steps when both elicits and tells are counted as learning opportunities.
- **nTellsSinceLastWrongPrin:** The number of tell steps the student has received since the last wrong elicit step for the steps that require a probability principle. This feature reflects the opportunities the student has received to learn how the tutor uses probability principles since the last time a wrong answer was given to a principle-involved elicit step.
- **nStepSinceLastWrongPrinSession:** The number of steps the student has completed since the last wrong elicit step for the steps that require a probability principle in the current session. This feature reflects the student's recent competence on probability-principle-involved steps in the current session when both elicits and tells are counted as learning opportunities.
- **nTellsSinceLastWrongPrinSession:** The number of tell steps the student has received since the last wrong elicit step for the steps that require a probability principle in the current session. This feature reflects the opportunities the student has received to learn how the tutor uses probability principles since the last time a wrong answer was given to a principle-involved elicit step in the current session.
- **nCorrectElicitsSinceLastWrongPrin:** The number of elicit steps the student has successfully solved since the last wrong elicit step for the steps that require a probability principle. This feature reflects the student's recent competence in solving principle-involved elicit steps.
- **nCorrectElicitsSinceLastWrongPrinSession:** The number of elicit steps the student has successfully solved since the last wrong elicit step for the steps that require a probability principle in the current session. This feature reflects the student's recent competence in solving principle-involved elicit steps in the current session.

The following four features describe students' performance on the first occurred select- and apply-principle steps in each problem. Since our ITS requires students to solve problems following the target variable strategy, the selection and application of the first principle require considering many more things than the rest. Therefore, we extracted these features to observe students' competence in handling complicated situations.

- **pctCorrectFirst:** The number of elicit steps the student has correctly solved, divided by the total number of elicit steps the student has received so far for the first occurred select- and

apply- principle steps in each problem. This feature reflects the student's competence in handling difficult steps.

- **nStepsSinceLastWrongFirst:** The number of steps the student has completed since the last wrong elicit step for the first occurred select- or apply-principle steps in each problem. This feature reflects the student's recent competence in handling difficult steps, when both elicits and tells are counted as learning opportunities.
- **nTellsSinceLastWrongFirst:** The number of tell steps the student has received since the last wrong elicit step for the first occurred select- or apply-principle steps in each problem. This feature reflects the opportunities the student has received to learn how the tutor solves difficult steps since the last time a wrong answer was given to a first occurred select- or apply-principle steps in a problem.
- **nCorrectElicitsSinceLastWrongFirst:** The number of elicit steps the student has successfully solved since the last wrong elicit step for first occurred select- or apply-principle steps in each problem. This feature reflects the student's recent competence in solving difficult steps.

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

- **pctCorrectLastProb:** The number of elicit steps the student has correctly solved divided by the total number of elicit steps the student has received for the elicit steps in the last problem. This feature reflects the student's demonstrated competence in the last problem.
- **pctCorrectLastProbPrin:** The number of elicit steps the student has correctly solved divided by the total number of elicit steps the student has received for the principle-involved elicit steps in the last problem. This feature reflects the student's demonstrated competence in using probability principles in the last problem.

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

- **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 for the Addition Theorem for two events.
- **pctCorrectAdd3All:** pctCorrect for the select- or apply-principle steps for the Addition Theorem for three events.
- **pctCorrectCompAll:** pctCorrect for the select- or apply-principle steps for the Complement Theorem.
- **pctCorrectDeMorAll:** pctCorrect for the select- or apply-principle steps for the De Morgan's Law.
- **pctCorrectIndeAll:** pctCorrect for the select- or apply-principle steps for the Independent Theorem.
- **pctCorrectMutualAll:** pctCorrect for the select- or apply-principle steps for 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.

#### 4.4.5 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. This feature reflects the amount of help the student has proactively requested from the system.
- **nTotalHintSession:** The total number of hints the student has requested in the current session. This feature reflects the amount of help the student has proactively requested from the system in the current session.
- **nHintKC:** The total number of hints the student has requested so far for the current KC. This feature reflects the amount of help the student has proactively requested from the system for the current KC.
- **nHintSessionKC:** The total number of hints the student has requested so far for the current KC in the current session. This feature reflects the amount of help the student has proactively requested from the system for the current KC in the current session.
- **nTotalHintProblem:** The total number of hints the student has requested in the current problem. This feature reflects the amount of help the student needed to reach the current point in 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. This feature reflects the student's working speed on the hint-requested steps. The slower they are, the more thinking they have done before or after requesting hints.
- **durationSinceLastHint:** The time elapsed since the last hint was requested. This feature reflects the student's recent struggling level. The longer they did not request hints, the less struggling they were.
- **stepsSinceLastHint:** The number of steps the student has completed since the last hint-requested step. This feature also reflects the student's recent struggling level, when both elicit and tell are counted as learning opportunities. The more steps the student has completed without hints, the less struggling they were.
- **stepsSinceLastHintKC:** The number of steps the student has completed since the last hint-requested step for the current KC. This feature reflects the student's recent struggling level on the current KC.
- **totalTimeStepsHint:** The total time the student has spent on hint-requested steps. This feature reflects the total amount of thinking the student has done before or after requesting hints.

- **totalStepsHint:** The total number of steps where hints were requested. This feature reflects the student's overall struggling level. The more steps that they needed hints to complete, the more struggling they were.

## CHAPTER

# 5

# REINFORCEMENT LEARNING (RL) STUDY - COMPARING RL INDUCED PROBLEM-LEVEL VS. STEP-LEVEL POLICIES

Results from the granularity studies suggest that decision granularity can have an impact on student learning. However, there was no significant difference between the three granularity conditions in learning performance, and none of them significantly outperformed the WE-only and PS-only baselines. One possible reason is that the pedagogical decisions were randomly made, and thus, the instructional interventions were not adapted to improve students' learning performance. This raises the question of whether adaptively making decisions at different levels of granularity would impact students' learning differently.

To investigate this question, we applied reinforcement learning (RL) to induce a problem-level policy and a step-level policy using the same algorithm and following the same procedure. The effectiveness of them was evaluated in a classroom study where they were compared with each other as well as a problem-level and a step-level random baseline policies. Results showed that there was no significant difference between the two RL conditions, and none of them significantly outperformed the two baseline conditions. This chapter describes the policy induction procedure and the empirical evaluation study in detail.

## 5.1 Background - Prior Research on Applying RL for Pedagogical Policy Induction

A lot of prior research has applied reinforcement learning (RL) to induce pedagogical policies for intelligent tutoring systems (ITSs) [Bec00; Igl09a; Igl09b; Raf16; Wan17; She16b; Chi11; Zho17b; Man14]. 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. 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, can provide exploration opportunities at a relatively low cost. Moreover, simulation models may also be used for fast and inexpensive policy evaluations [Dor17]. On the other hand, the success of these approaches is heavily dependent on the accuracy of the simulations, which can often be hard to achieve. Beck et al. [Bec00] applied an online approach, temporal difference, with off-policy  $\epsilon$ -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 [Igl09b]. 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 [Raf16]. 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" [Sch17b]. These approaches avoid the possible errors and bias generated by simulation, 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 [She16b]. 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 [She18]. 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 [Chi11]. 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 [Man14]. 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.

While a lot of prior works have applied RL for pedagogical policy induction, they either focused on a single granularity level or treated all system decisions equally. For example, Chi et al. focused on the step-level decisions of elicit vs. tell [Chi11]. Shen et al. focused on the problem-level decisions of WE vs. PS [She16b; She18]. Beck et al. included decisions at three levels of granularity: topic, problem, and step in RL policy induction, but treated them equally.

In sum, previous research has shown that both online and offline RL approaches can induce effective pedagogical policies that can improve student learning or behavior. However, the necessity for accurate simulations (online) or large training corpora (offline) has limited the wide use of RL for pedagogical policy induction. Additionally, prior research in RL policy induction generally focused on making decisions at a single level of granularity or treated all system decisions equally. To the best of our knowledge, no prior study has directly compared RL induced problem-level and step-level policies.

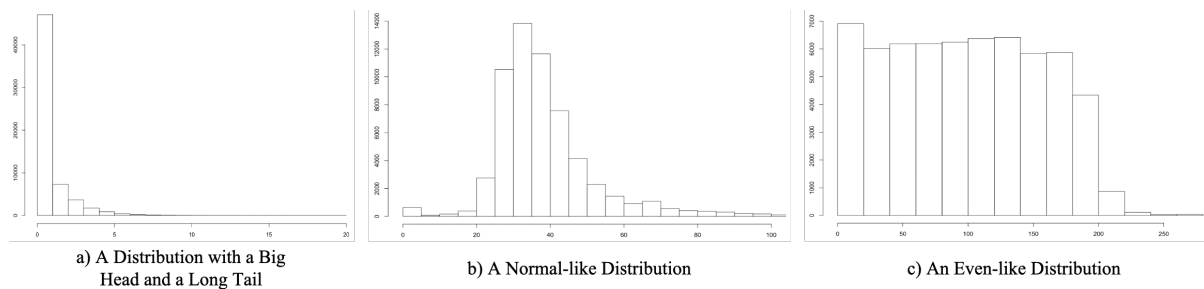
## 5.2 Method

In this study, we applied an offline model-based tabular RL to induce the policies, which were represented using state-action look-up tables. During learning, the table was updated by the RL agent using the transition and reward information  $S_i \xrightarrow{A_i, R_i} S_{i+1}$  contained in our exploratory trajectories. Since this approach requires a finite state space, we discretized all integer- or real-value features so that each feature only has several possible values. Additionally, we performed feature selection to balance the informative degree of the states, and the quality of the transition and reward information. More specifically, the more features we use to represent the states, the more informative each state will be. Though on the other hand, the more states there are, the less often each state will be covered by the exploratory data and, in turn, the quality of the transition and reward information will be

lower. Feature selection balances the trade-off by using the most informative features to represent the states.

### 5.2.1 Feature Discretization

The feature discretization process transfers numerical values, such as integer and real number, into ordinal values where each number represents an independent category, and there is an order among the categories. More specifically, this process segments the value range of a feature into several intervals (2-6 in our case) and assigns a number for each of them. This defines a mapping from numerical values to ordinal values, and, based on it, we discretize the features.



**Figure 5.1** Features with Different Distribution

The discretization is a generalization process, where similar values are grouped together. On the one hand, we want to segment the range into more groups so that significantly different values will not be grouped together. On the other hand, we have to control the number of groups to be small so that each group will have a considerable amount of visits in our exploratory data. To pick a reasonable group number for each feature, we checked the value distribution of each feature and, based on it, determined the group number and the discretization strategy. More specifically, for each feature, we first determine whether to create independent groups for its head and tail and then segment the body. An independent group was created for the head (the small value end) when it was big. For the tail, an independent group was created when it was long. Figure 5.1.a shows an example that has both a big head and a long tail. In this example, we set a threshold 0 for the head and 4 for the tail. That means values  $\leq 0$  belongs to the head group, and values  $\geq 4$  belongs to the tail group. Once the head and tail are determined, we check the distribution of the remaining data and choose the strategy, between median split and k-mean clustering, for segmentation. Generally speaking, we apply median split for normal-like distributions, as shown in Figure 5.1.b. That's because most of the data are in a small interval, and thus, k-means would cluster them into the same group. In this case, the difference between samples will be eliminated by the discretization. For all other distributions, such as even-like distributions, as shown in Figure 5.1.c, we applied k-means for the segmentation. When applying k-means, we manually set the number of groups for each feature based on the distribution of its value.

## 5.2.2 Policy Induction

When the state representation (a set of discretized features) is determined, we can apply the value iteration algorithm to induce the policy. This algorithm reaches the optimal policy by finding the optimal value for each state  $V^*(s)$ , and the optimal value function  $Q^*(s, a)$ . The optimal state value  $V^*(s)$  indicates the expected discounted reward that the agent will receive if it starts in  $s$  and follows the optimal policy to the end, while the optimal value function  $Q^*(s, a)$  specifies the expected discounted reward the agent will receive if it takes action  $a$  in state  $s$  and follows the optimal policy to the end. The value iteration algorithm finds  $V^*(s)$  and  $Q^*(s, a)$  by iteratively updating  $V(s)$  and  $Q(s, a)$  following equations 5.1 and 5.2 until they converge:

$$Q(s, a) = R(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(S_j | S_i, a) V(s') \quad (5.1)$$

$$V(s) = \max_a Q(s, a) \quad (5.2)$$

$p(S_j | S_i, a)$  and  $R(s, a)$  are estimated from our training data set and  $0 \leq \gamma \leq 1$  is a discount factor.

Once a policy is induced, we can estimate its effectiveness by calculating the Expected Cumulative Reward (ECR) following Equation 5.3 based on our training data set.

$$ECR_\pi = \sum_{i=1}^n \frac{N_i}{N_1 + \dots + N_n} \times V(s_i) \quad (5.3)$$

$s_1, \dots, s_n$  denotes the start state of each trajectory,  $V(s_i)$  are the V-values for state  $s_i$ , determined by RL,  $N_i$  is the number of times that  $s_i$  is the start state in our data set, and  $\frac{N_i}{N_1 + \dots + N_n}$  indicates the estimated probability that  $s_i$  is the start state. This equation calculates the weighted average of the V-values for all start states. The higher the ECR value of a policy, the better it is expected to be.

## 5.2.3 Feature Selection

We adopted the feature selection approach proposed by Shen et al. [She16a] in the general policy induction procedure. This approach selects features one-by-one following a forward stepwise selection procedure. More specifically, given the currently selected feature set, it selects the next one based on the correlation between each remaining feature and the selected feature set. Five correlation metrics were explored, and for each metric, high and low correlations were considered, which results in 10 methods in total.

Algorithm 1 (from [She16a]) shows the detailed feature selection procedure, which includes three main phases: 1) determining an initial feature (line 3-5), 2) determining a set of candidate features based on correlations (line 7-9) and 3) selecting the one that results in the highest ECR (10-12). Phase 2 and 3 repeats recursively until the maximal capacity is reached.

Besides the correlation based methods, we also explored an ensemble approach [She16a], which combines the 10 correlation-based methods and 4 *RL-based* methods [Chi11]. This approach employs a procedure similar to Algorithm 1, except phase 2, where the feature pool includes all the

---

**Algorithm 1** Correlation-based Feature Selection Algorithm

---

```
1: Inputs  $\Omega$ : Feature space;  $\mathcal{D}$ : Training data;  $\mathcal{N}$ : Maximum number of selected features;  $m$ : Correlation metric
2: Output  $\mathcal{S}^*$ : Optimal feature set
3: for  $f_i$  in  $\Omega$  do
4:    $ECR_i \leftarrow \text{CALCULATE-ECR}(\mathcal{D}, f_i)$ 
5: Add  $f^*$  with highest  $ECR$  to  $\mathcal{S}^*$ 
6: while  $\text{SIZE}(\mathcal{S}^*) < \mathcal{N}$  do
7:   for  $f_i$  in  $\Omega - \mathcal{S}^*$  do
8:      $C_i \leftarrow \text{CALCULATE-CORRELATION}(\mathcal{S}^*, f_i, m)$ 
9:    $\mathcal{F} \leftarrow \text{SELECTTOP}(C, 5, \text{reverse})$  ▷ Select top 5 features based on correlation metrics
10:  for  $f_i$  in  $\mathcal{F}$  do
11:     $ECR_i \leftarrow \text{CALCULATE-ECR}(\mathcal{D}, \mathcal{S}^* + f_i)$ 
12:  Replace  $\mathcal{S}^*$  by  $\mathcal{S}^* + f_i$  with highest  $ECR$ 
```

---

candidate features determined by the 14 feature selection methods (each method proposes 5 candidates).

### 5.2.4 Training Data

Our training data set was collected by training students on Pyrenees using random policies in the Fall 2014, Spring 2015, and Fall 2015 semesters. All students followed the same general procedure, used the same ITS, and studied the same training materials. The problem-level policy was induced from problem-level random exploratory data, which includes 112 students while the step-level policy was induced from step-level random exploratory data, which includes 149 students. From the student-system interaction logs, we extracted 142 features (described in Section 4.4) for feature selection and state representation. The problem-level actions were WE vs. PS, and the step-level actions were elicit vs. tell. The rewards were student normalized learning gain (NLG), defined as  $\frac{\text{posttest} - \text{pretest}}{1 - \text{pretest}}$ . Since NLG is available only when the whole training is complete, only the last state in each trajectory has a non-zero reward.

Both the problem- and step-level policies were induced as part of the feature selection procedure. For each candidate feature set, the algorithm induces a policy based on it to calculate ECR. The feature space for the step-level policy included all 142 features (described in Section 4.4), while for the problem-level policy, 12 features that describe the student’s learning state in the current problem were excluded because there is no chance to observe the state at the beginning of the problem. The max feature capacity was set to be 8 (for both the problem- and step-level policies) – that is, the feature selection process terminates when 8 features are selected. When the feature selection terminates, we pick the policy with the best balance between ECR (the higher, the better) and the size of the feature set (the smaller, the better) for empirical evaluation.

The final resulting problem-level policy involves six features and 352 rules. The following lists the six selected features in the order they were selected.

- **pctCorrectAdd3Select:** 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 for the select principle steps that require selecting the Addition Theorem for three events.
- **nCorrectElicitsSinceLastWrong:** The number of elicit steps the student has successfully solved since the last wrong elicit step.
- **AvgTimeOnStepWithHint:** The average time the students spent on each hint-requested step.
- **nPrincipleInProblem:** The number of principles needed to solve the current problem (some principles may be applied more than once).
- **tutAverageWords:** The average number of words the tutor used in each step.
- **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 received. Select- and apply- principle are considered as quantitative steps because they require using probability equations for inference.

The final resulting step-level policy involves seven state features and 3706 rules. The seven selected features are listed below.

- **avgTimeOnStepElicit:** The average time the student spent on each elicit step.
- **avgTimeOnStepSessionElicit:** The average time the student spent on elicit steps in the current session.
- **ntellsSinceElicit:** The number of tells the student has received since the last elicit, irrespective of the KC involved.
- **nStepSinceLastWrongKC:** The number of steps the student has completed since the last wrong elicit step on the current KC.
- **timeInSession:** The time elapsed since the start of the current session.
- **nTellsSinceLastWrong:** The number of tell steps the student has received since the last wrong elicit step.
- **nCorrectElicitsSinceLastWrongKCSession:** The number of elicit steps the student has successfully solved since the last wrong elicit step on the current KC in the current session.

## 5.3 Empirical Experiment

### 5.3.1 Conditions and Participants

The effectiveness of the RL induced policies was evaluated in a classroom study by comparing against problem-level and step-level random baseline policies. Since problem-level WE and PS

and step-level elicit and tell are always considered to be reasonable interventions in our learning context, our baseline policies are “*random yet reasonable*”.

The participants in the study were undergraduate students enrolled in the discrete math class in the Fall 2016 semester, and this study was given as one of their regular homework assignments. They had one week to complete the study and were graded (in class) based on their demonstrated effort, rather than performance. 203 students were randomly assigned to four conditions:  $N = 61$  for problem-level RL ( $RL_{Prob}$ ),  $N = 61$  for step-level RL ( $RL_{Step}$ ),  $N = 40$  for problem-level Baseline  $Base_{Prob}$ , and  $N = 41$  for step-level Baseline  $Base_{Step}$ . Since the primary goal of this study was to examine the effectiveness of the RL induced policies, we assigned more students to the two RL conditions than the two Baseline conditions. Due to preparations for exams and length of the experiment, 164 students completed the study. 8 students who performed perfectly in the pre-test or completed the study in groups were excluded from our subsequent analysis. The remaining 156 students were distributed as follows:  $N = 45$  for  $RL_{Prob}$ ;  $N = 45$  for  $RL_{Step}$ ;  $N = 31$  for  $Base_{Prob}$ ; and  $N = 35$  for  $Base_{Step}$ . A  $\chi^2$  test showed that the completion rate did not differ significantly between the conditions:  $\chi^2(3) = 2.33, p = 0.507$ .

### 5.3.2 Procedure and Measures

All students went through four phases: 1) textbook, 2) pre-test, 3) training on the ITS, and 4) post-test, as shown in Table 3.1. During **textbook**, all students read a general description of each principle, reviewed some examples, and solved some training problems. The students then took a **pre-test**, which contained a total of 14 single- and multiple-principle 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 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. 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. Conditions differed in pedagogical policy (RL vs. random) and decision granularity (problem- vs. step-level). The pre- and post-test were graded by a single experienced grader following the partial credit rubric in a double-blind manner. More details about the experimental procedure are described in Chapter 2.

After the study, we performed statistical analyses on students’ learning performance, time on task, and the policies’ pedagogical decision-making. Specific learning performance measures are listed below.

- **Pre-test**: calculated based on the 14 pre-test questions;
- **Isomorphic Post-test**: calculated based on the 14 post-test questions that are isomorphic to the pre-test;
- **Full Post-test**: calculated based on the 20 post-test questions;

**Table 5.1** Procedure of the RL study

Procedure	Conditions			
	RL <sub>Prob</sub>	RL <sub>Step</sub>	Base <sub>Prob</sub>	Base <sub>Step</sub>
Textbook	Read a probability textbook			
Pre-test	Complete 14 test questions (10 single- and 4 multiple-principle)			
ITS-training	RL WE/PS	RL elicit/tell	random WE/PS	random elicit/tell
Post-test	Complete 20 test questions (10 single- and 10 multiple-principle)			

- **Adjusted Post-test:** calculated following the equation:  $adjusted = post - k * (pre - \overline{pre})$ , where  $k$  is the coefficient of the linear regression model  $post = k * pre + b$  generated by ANCOVA analysis, and  $\overline{pre}$  is the average pre-test score across all students;

Similar to the granularity studies, the time students spent on the training task (time on task) was calculated based on an analysis of the system logs with the idle time excluded. We still used the 141-second threshold for determining idle intervals.

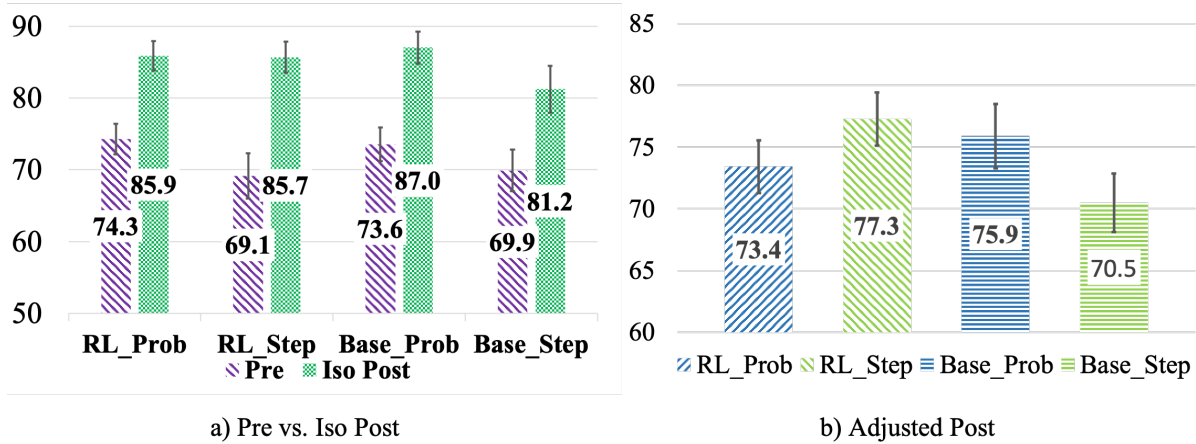
### 5.3.3 Results

**Table 5.2** Results of the Fall 2016 RL Study

Condition	Pre	Iso Post	Full Post	Adj Post	Time (hours)
RL <sub>Prob</sub> (45)	74.3(14.3)	85.9(13.7)	74.8(16.3)	73.4(14.4)	1.76(.61)
RL <sub>Step</sub> (45)	69.1(21.1)	85.7(14.6)	75.9(17.2)	77.3(14.4)	1.96(.57)
Rand <sub>Prob</sub> (31)	73.6(12.9)	87.0(12.4)	76.9(15.3)	75.9(14.5)	1.73(.66)
Rand <sub>Step</sub> (35)	69.9(17.1)	81.2(19.5)	69.5(19.3)	70.5(14.1)	1.88(.44)

**Incoming Competence:** A one-way ANOVA analysis on the pre-test score showed that there was no significant difference among the four conditions:  $F(3, 152) = 0.95$ ,  $p = 0.420$ ,  $\eta = 0.018$ . This suggests that our random assignment balanced students' incoming competence quite well. Table 5.2 shows the mean and standard deviation (SD) of students' learning performance and time on task results across the four conditions. From left to right, it shows the conditions with the number of students in the parentheses, pre-test (Pre), isomorphic post-test (Iso Post), full post-test (Full Post), adjusted post-test (Adj Post), and total training time in hours (Time).

**Students' Improvement Through Training:** To measure the improvement students made through training, we compared their pre-test with isomorphic post-test, as shown in Figure 5.2.a. A repeated



**Figure 5.2** Students' Improvement and Learning Performance

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 of test type:  $F(1, 152) = 104.26, p < 0.0001, \eta = 0.402$ . More specifically, all four conditions scored significantly higher in the isomorphic post-test than in the pre-test:  $F(1, 44) = 27.25, p < 0.0001, \eta = 0.382$  for  $RL_{Prob}$ ;  $F(1, 44) = 34.07, p < 0.0001, \eta = 0.436$  for  $RL_{Step}$ ;  $F(1, 30) = 26.27, p < 0.0001, \eta = 0.467$  for  $Base_{Prob}$  and  $F(1, 34) = 18.30, p = 0.0001, \eta = 0.350$  for  $Base_{Step}$ . This suggests that our ITS training is effective regardless of the pedagogical policy and decision granularity.

**Students' Learning Performance:** To measure students' learning performance, we conducted an analysis on their full post-test score. A two-way ANCOVA analysis on the factors of granularity (Prob vs. Step) and policy (RL vs. Baseline) using the pre-test score as a covariate showed a significant interaction effect:  $F(1, 151) = 3.92, p = 0.049, \eta = 0.018$ , as shown in Figure 5.2.b for the adjusted post-test score. However, subsequent contrast analysis revealed that most of the between-group contrasts were not significant. Specifically, there was no significant difference between the two RL groups ( $RL_{Prob}$  vs.  $RL_{Step}$ ):  $t(151) = -1.26, p = 0.209, d = 0.27$  or between the two baselines ( $Base_{Prob}$  vs.  $Base_{Step}$ ):  $t(151) = 1.52, p = 0.131, d = 0.38$ . At the problem-level granularity, there was no significant difference between ( $RL_{Prob}$  and  $Base_{Prob}$ ):  $t(151) = 0.73, p = 0.464, d = 0.17$ . Only at the step level,  $RL_{Step}$  outperformed  $Base_{Step}$  significantly:  $t(151) = 2.09, p = 0.038, d = 0.47$ . However, the difference between  $RL_{Step}$  and  $Base_{Prob}$  was not significant:  $t(151) = 0.41, p = 0.683, d = 0.10$ . Overall the results suggest that RL induced problem- and step-level policies may not always differ significantly in effectiveness and RL induced policies that make decisions at a single level of granularity may not always outperform random baselines.

**Time on task:** A two-way ANOVA analysis on students' time on task showed a marginally significant main effect of granularity in that the two problem-level conditions spent significantly less time than the two step-level conditions:  $F(1, 152) = 3.59, p = 0.060, \eta = 0.023$ . However, subsequent contrast analysis showed no significant difference between the two RL groups ( $RL_{Prob}$  vs.  $RL_{Step}$ ) or the two Baseline groups ( $Base_{Prob}$  vs.  $Base_{Step}$ ).

**Table 5.3** Tutor Decisions in Terms of Elicits and Tells

Condition	Elicit	Tell	Pct Tell
RL <sub>Prob</sub>	167.9(109.9)	227.0(112.1)	57.3(28.3)
RL <sub>Step</sub>	205.7(49.3)	183.8(51.0)	47.1(12.9)
Base <sub>Prob</sub>	195.5(65.8)	194.4(65.2)	49.9(16.7)
Base <sub>Step</sub>	198.2(13.8)	197.3(18.8)	49.8(2.7)

**Tutor Decisions:** Since two of the 12 ITS training problems were fixed to be PS, we excluded them from our statistical analysis, and all the tutor decision results were generated based on the remaining 10 problems. At the problem level, students in the RL<sub>Prob</sub> condition received 5.91 WEs ( $SD = 2.79$ ) and 4.09 PSs ( $SD = 2.79$ ) on average; while the Base<sub>Prob</sub> condition received 5.06 WEs ( $SD = 1.59$ ) and 4.94 PSs ( $SD = 1.59$ ) on average. A t-test showed that there was no significant difference between the two conditions on the number of WE received :  $t(74) = -1.52, p = .132, d = 0.36$ . At the step level, the RL<sub>Step</sub> condition received 183.8 ( $SD = 51.0$ ) tells and 205.7 ( $SD = 49.3$ ) elicits on average (47.1% tells,  $SD = 16.7$ ); while Base<sub>Step</sub> received 197.3 ( $SD = 18.8$ ) tells and 198.2 ( $SD = 13.8$ ) on average (49.8% tells,  $SD = 2.7$ ). A t-test showed that there is no significant difference between them on tell percentage:  $t(78) = 1.22, p = 0.228, d = 0.27$ .

In order to conduct an analysis across the four conditions, we transferred problem-level decisions into step-level ones. Table 5.3 shows the number of step-level decisions students received across the four conditions, showing the number of Elicit steps (Elicit), Tell steps (Tell), and the percentage of tell (Pct Tell). As we can see in the table, the two baseline policies (Base<sub>Prob</sub> and Base<sub>Step</sub>) balanced elicit and tell well; while RL<sub>Prob</sub> selected more tells than elicits (57.3% tells) and RL<sub>Step</sub> selected slightly more elicits than tells (47.1% tells). A one-way ANOVA analysis on the percentage of tell showed a marginally significant difference among the four conditions:  $F(3, 152) = 2.52, p = 0.060, \eta = 0.047$ . Subsequent contrast analysis revealed that RL<sub>Prob</sub> selected significantly more tells than RL<sub>Step</sub>:  $t(152) = 2.63, p = 0.009, d = 0.46$  and marginally more tells than Base<sub>Prob</sub>:  $t(152) = 1.74, p = 0.084, d = 0.31$ . Additionally, results showed that the two problem-level conditions had a larger SD in tell percentage than the two step-level conditions. This is not surprising because the problem-level decisions are more coarse than the step-level ones in that one problem-level decision determines many steps. Moreover, the two RL conditions had a larger SD in tell percentage than the two Baseline conditions. This suggests that the RL policies made more personalized decisions.

## 5.4 Conclusion and Discussion for Chapter 5

The RL study investigated the effectiveness of RL-induced problem- and step-level policies. The two policies were induced using the same algorithm and following the same procedure. In a classroom study, the two RL policies were compared with two baselines: problem-level random and step-level

random. Results showed that there was no significant difference between the two RL conditions, and none of them significantly outperformed the two baseline conditions.

There are many possible reasons that the RL policies did not significantly outperform the random baselines. Here, we discuss some of the reasons related to decision granularity. For the RL induced problem-level policy, one possible reason is that problem-level decisions are too coarse and, thus, not flexible enough. In other words, each training problem requires applying multiple probability principles to solve, but students' proficiency in the principles may differ. This may result in the situation that they need a tell for some of the steps, and an elicit for some others. Since problem-level decisions cannot provide step-level adaptation, it is possible that students' learning needs were not well-met.

However, the flexibility hypothesis does not explain why the step-level RL policy did not significantly outperform random baselines. Theoretically, step-level decisions allow the system to fine-tailor its instructional interventions, and many researchers believe that the effectiveness of ITS comes from fine-grained step-by-step support [Van06]. However, our results showed that this is not always the case in that it did not significantly outperform the problem-level random baseline. One possible reason is that problem- and step-level granularities involve different learning mechanisms. As discussed in the granularity study, problem-level granularity switches between observing (WE) and doing (PS), while step-level granularity involves students in co-constructing the solution with the tutor. Thus, focusing on a single level of granularity does not cover all of them, and in turn, may not always be able to provide the best intervention.

In sum, results showed that there was no significant difference between the RL-induced problem- and step-level policies in effectiveness, and none of them significantly outperformed the two random baselines. This suggests that RL induced pedagogical policies that make decisions at a single level of granularity may not always be effective. On the other hand, the granularity study results showed that making Both-level decisions can benefit more students than either the problem- or step-level. Motivated by the findings, we then applied hierarchical reinforcement learning to induce a policy that makes decisions at both the problem- and step-level.

## CHAPTER

# 6

# HIERARCHICAL REINFORCEMENT LEARNING (HRL) STUDY - INDUCING A HIERARCHICAL PEDAGOGICAL POLICY

Results from the RL study showed that the RL induced problem-level and step-level policies did not differ significantly in effectiveness, and none of them significantly outperformed the two random baselines. This suggests that RL induced policies that make decisions at a single granularity level may not always be effective. On the other hand, results from the granularity study showed that Both-level decisions can benefit more students than either the problem- or step-level. Thus, it is possible that RL induced pedagogical policies that make decisions at both levels can be effective. In this study, we apply a Gaussian Processes (GP) based hierarchical reinforcement learning (HRL) approach to induce a pedagogical policy that makes decisions first at the problem-level and then the step-level.

More specifically, the policy first makes a problem-level decision to decide whether the next problem should be WE, PS or CPS. If WE is selected, an all-tell step-level policy will be carried out; if PS is selected, an all-elicited step-level policy will be executed; finally, if CPS is selected, the tutor will decide whether to elicit or tell each step based on the corresponding HRL induced step-level policy. In an empirical evaluation study, the HRL policy was compared with a Deep Q-Network (DQN) induced step-level policy and a *random yet reasonable* step-level baseline policy. Results showed that the HRL policy was significantly more effective than the DQN and the random baseline policy. For time on task, no significant difference was found between the HRL and Random conditions, but

the former (HRL) spent significantly more time than DQN. Finally, the HRL induced policy is more likely to select PS and CPS than WE.

## 6.1 Background - HRL Approaches and Applications

A lot of prior research has shown that hierarchical reinforcement learning (HRL) can be more effective and data-efficient than flat (standard) RL approaches [Cua10; Rya00; Pen17; Wan18; Kul16]. 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 fewer 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 [Pen17]. Results showed that HRL policies can complete tasks that flat RL policies could not complete.

In recent years, inspired by the successes of Deep RL, there is a growing trend in incorporating deep RL approaches into HRL. Most of deep RL approaches use a deep neural network to approximate the Q-value function or the policy, which allows the agent to handle complicated environments, such as the screen of video games [Kul16]. Kulkarni et al. proposed one of the earliest Deep HRL approaches in 2016, right after Deep Mind showed the power of Deep RL in Atari games [Mni15]. Similar to traditional HRL approaches, the Deep HRL framework includes a meta-controller for high-level decision-making and a controller for low-level decision-making. Though unlike most traditional HRL approaches using tabular Q-value functions, this framework employs deep neural networks based Q-value functions. This combination allows HRL to handle complicated environments while keeping its traditional advantages. In two experiments, the Deep HRL approach has been shown to outperform traditional Q-learning and DQN.

Since then, a variety of Deep HRL approaches were proposed, enhancing the original Deep HRL framework from different perspectives [Vez17; Haa18b; Flo17]. For example, Vezhnevets et al. proposed a FeUdal Networks HRL approach [Vez17] that can automatically discover goals for the high-level agent. This function avoids the necessity of manually setting the goals for the agent, which reduced the requirement of domain knowledge for using HRL. Similarly, Haarnoja proposed another Deep HRL approach that makes and passes hierarchical decisions in latent spaces [Haa18b]. Again, this approach releases users from setting the hierarchy manually. In another work, Florensa proposed a two-phase learning procedure for HRL, which improves data-efficiency. In the first phase, the agent learns general skills, while in the second one, it learns task-specific skills. Similar to other approaches, deep neural networks were used in the learning process.

As HRL approaches became more and more powerful, there is a growing trend to apply them in real-world applications. Wang et al. applied HRL for video captioning, which automatically generates text descriptions for videos [Wan18]. They proposed a two-agent model with the manager determin-

ing the context of the video and the worker generating specific text following the manager’s guidance. Results showed that the HRL based approach achieved state-of-the-art results. Zhang et al. applied HRL to revise user profiles for course recommendation in MOOC [Zha19]. The recommendation system recommends courses based on the user’s previous course-taking profile, and HRL removes the courses that are unlikely to impact the user’s next decision. The manager determines whether a user’s profile needs to be revised, and the worker agent removes the specific courses. Results showed that the HRL-involved model significantly outperformed the state-of-the-art recommendation model. In another work, Kao et al. applied HRL for question-based disease diagnosis [Kao18]. They trained a two-agent system for efficient question asking, with the high-level agent locating the symptom and the low-level agent identifying the exact disease. Results showed that the HRL-based system achieved a significantly higher symptom diagnosis accuracy over traditional systems.

While prior research has proposed many HRL approaches, most of them are online. That’s because 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. Recently, there are few works focusing on extending existing HRL approaches for offline uses, but they often build on assumptions that do not hold in our domain. For example, Schwab et al. proposed to simulate the transition and reward of higher-level actions based on lower-level information. However, given that human learning has not been well-studied yet, it is still hard to precisely predict the changes of the learning state after a series of actions. Fortunately, Hamoon Azizoltani (a post-doc in our group) helped us tackle this challenge by incorporating Gaussian Processes in the HRL policy induction procedure, which allowed us to induce effective pedagogical policies offline. To the best of our knowledge, this is the first attempt to apply HRL for pedagogical policy induction.

## 6.2 Method

### 6.2.1 Training Data

The training corpus for HRL policy induction contains 1118 students’ interaction logs, Each of them spent around 2 hours in training and completed around 400 steps. To prepare the data for HRL, we transferred the logs into a dataset  $\mathcal{D}$  consisting of trajectories  $d$  in the form of  $s_1 \xrightarrow{a_1, r_1} s_2 \xrightarrow{a_2, r_2} \dots s_n \xrightarrow{a_n, r_n}$ . Here  $s_i \xrightarrow{a_i, r_i} s_{i+1}$  indicates that in 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  $s_{i+1}$ . The environment states were represented using 142 features, as described in Chapter 4, the actions were 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 chose Normalized Learning Gain (NLG) as the reward because it measures students’ improvement *irrespective of their incoming competence*.  $NLG = \frac{posttest - pretest}{\sqrt{1 - pretest}}$  where *pretest* and *posttest* refer to the students’ test scores before and after the ITS training respectively and 1 is the maximum score. Different from the the widely

used NLG equation  $NLG = \frac{posttest - pretest}{1 - pretest}$ , our equation has an extra square root in the denominator, which was used to reduce the variance.

To verify whether the square root can actually reduce the variance, we compared the two NLG equations using the data collected in the granularity studies. Table 6.1 shows a comparison of the two NLG scores for the High, Medium, and Low students respectively (to be consistent with the range of other score measures, all numbers are timed 100). As expected, the square root indeed can reduce the variance, especially for the High group (from 117.6 to 38.9). In addition, the square root rose the average of High from -30.8 to -9.4 and reduced the average of Low from 20.4 to 15.3, which reduced the difference between the High and Low groups.

**Table 6.1** A Comparison of the Two NLG Definitions

Definition	High	Medium	Low
$\frac{posttest - pretest}{1 - pretest}$	-30.8(117.6)	6.72(55.03)	20.4(34.78)
$\frac{posttest - pretest}{\sqrt{1 - pretest}}$	-9.4(38.9)	3.91(32.42)	15.3(25.44)

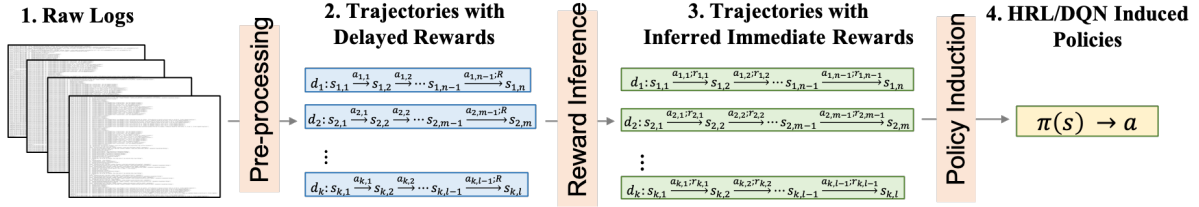
## 6.2.2 Challenges for HRL Policy Induction

Applying our training data for offline HRL policy induction, we face at least two challenges: 1) delayed rewards and 2) 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 example, 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 [Wil08].

On the other hand, the most appropriate reward to use in ITS (student NLG) are 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 caused by delayed rewards is the induction of hierarchical policies. Since each training problem in our ITS covers different knowledge components,



**Figure 6.1** Flow Chart of the HRL Policy Induction Procedure

we induce an independent step-level sub-policy for each of them, and this requires us to assign a reward for each problem (or each step). To tackle this challenge, we applied a Gaussian Processes based (GP-based) approach, proposed by Hamoon Azizsoltani [Azi19], 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 since 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. Fortunately, Hamoon Azizsoltani was familiar with Gaussian Processes (GP), which is a powerful tool for handling uncertainty. Thus, we then worked together to incorporate GP into the HRL policy induction procedure.

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 [Ras04]. 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 6.1 shows an overview of the HRL policy induction procedure. The pre-processing procedure was described in Chapter 4, and the reward inference and policy induction procedure are described in the following.

### 6.2.3 GP-Based Immediate Rewards Inference

The GP-Based immediate rewards inference approach takes state-action trajectories  $s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \dots s_n \xrightarrow{a_n}$  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 [Ras04; Azi18] 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 [Azi19]. The learning objective is to minimize the additive error of its output rather than the direct output error [Guo05].

More specifically, in the context of GP, a function can be specified by a mean and a covariance (kernel) 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 [Eat83], as shown in the following two equations [Azi19]:

$$\mathbb{E}[\mathbf{r}|\mathbf{R}] = \mathbb{E}[\mathbf{r}] + \mathbf{C}_{\mathbf{rR}}\mathbf{C}_{\mathbf{RR}}^{-1}(\mathbf{R} - \mathbb{E}[\mathbf{R}]) \quad (6.1)$$

$$\mathbb{C}[\mathbf{r}|\mathbf{R}] = \mathbf{C}_{\mathbf{rr}} - \mathbf{C}_{\mathbf{rR}}\mathbf{C}_{\mathbf{RR}}^{-1}\mathbf{C}_{\mathbf{Rr}} \quad (6.2)$$

In these equations,  $\mathbb{E}[\mathbf{r}]$  is the mean of the prior function;  $\mathbf{C}_{\mathbf{rR}}$  and  $\mathbf{C}_{\mathbf{Rr}}$  are the cross-covariance between the delayed rewards and the immediate rewards;  $\mathbf{C}_{\mathbf{RR}}$  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  $\mathbf{C}_{\mathbf{rr}}$  is the covariance matrix of the immediate rewards. Here,  $\mathbb{E}[\mathbf{r}] = \mu_{\mathbf{r}}$  and  $\mathbf{C}_{\mathbf{rr}} = \mathbf{C}_{\mathbf{r}}$  are determined by the prior reward function and  $\mathbf{C}_{\mathbf{RR}}$  and  $\mathbf{C}_{\mathbf{rR}}$  are estimated using our data based on the definition of covariance and cross-variance respectively [Fel08]. Next, we will describe how  $\mathbf{C}_{\mathbf{RR}}$  and  $\mathbf{C}_{\mathbf{rR}}$  are calculated with the assumption that the sum of the immediate rewards equals to the delayed reward.

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



$Q(s, a)$ , denoted as 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[Sut99]. In SMDPs, the Bellman equation can be rewritten as:

$$Q^*(s, a) = R(s, a) + \sum_{s', t'} \gamma^{t'} P(s', t' | s, a) \max_{a' \in A} Q(s', a'), \quad (6.6)$$

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 that the objective is to discover actions that maximize the long-term cumulative reward. Local learning focuses not on learning the best policy for the ultimate 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 following equation 6.6. The algorithm is shown in Algorithm 2. To induce the hierarchical policy, we defined 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 defined a step-level semi-MDP for inducing a step-level policy to determine elicit vs. tell if a complex activity CPS was selected for that training problem. Inferred immediate rewards were used for all semi-MDPs. The states were represented using 142 features. To equalize the impact of each feature, all features were normalized to the range of [0,1].

Here, the standard GP Regression was used to approximate the Q-value function. Remind that in the context of GP, a function can be specified by the mean and covariance function. In Q-function approximation, it takes state-action-Q observations  $(S, A) \rightarrow Q$  and a prior covariance function (kernel) as input and specifies the Q-function's posterior mean and covariance:  $(\hat{\mathbf{Q}}^\pi)' \sim \mathcal{N}(\overline{(\hat{\mathbf{Q}}^\pi)'}, \text{COV}((\hat{\mathbf{Q}}^\pi)'))$ . To model possible uncertainty, we add an Independent and Identically-Distributed noise to the prior covariance function:  $\mathcal{E} \sim \mathcal{N}(0, \sigma_n^2)$ . According to the theorem of conditional probability density functions for multivariate Gaussians [Ras04], the mean,  $\overline{(\hat{\mathbf{Q}}^\pi)'}$ , and covariance  $\text{COV}((\hat{\mathbf{Q}}^\pi)')$  of the posterior distribution can be calculated using the following two equations [Gol98; Ras04]:

$$\overline{(\hat{\mathbf{Q}}^\pi)' } = K(\mathbf{X}', \mathbf{X}) [K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}]^{-1} \overline{(\hat{\mathbf{Q}}^\pi)} \quad (6.7)$$

$$\begin{aligned} \text{COV}((\hat{\mathbf{Q}}^\pi)') &= K(\mathbf{X}', \mathbf{X}') + \mathbf{C}_{\mathbf{Q}^\pi} - K(\mathbf{X}', \mathbf{X}) \\ &\quad [K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}]^{-1} K(\mathbf{X}, \mathbf{X}'). \end{aligned} \quad (6.8)$$

where  $\mathbf{X}$  is the observation points (the state-action pairs  $(S, A)$  in our training data),  $\overline{(\hat{\mathbf{Q}}^\pi)}$  and  $\mathbf{C}_{\mathbf{Q}^\pi}$  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),  $\sigma_n^2$  is a parameter,  $K(\mathbf{X}, \mathbf{X})$  is a covariance matrix evaluated on observation points,  $K(\mathbf{X}', \mathbf{X}')$  is a covariance matrix

---

**Algorithm 2** Learning Algorithm for GP based HRL

---

```
1: # Input: a dataset  $\mathcal{D}$  consisting of trajectories  $s_1 \xrightarrow{a_1, r_1} s_2 \xrightarrow{a_2, r_2} \dots s_n \xrightarrow{a_n, r_n}$ 
2: # Denote the collection of state-action pairs  $(s_i, a_i)$  appeared in  $\mathcal{D}$  as  $(\tilde{S}, \tilde{A})$ 
3: Give each  $(s_i, a_i)$  in  $(\tilde{S}, \tilde{A})$  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  $(\tilde{S}, \tilde{A}; \tilde{Q})$ 
4: Converged  $\leftarrow$  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  $(\tilde{S}, \tilde{A}; \tilde{Q})$ 
8:   Generate a policy  $\pi$  based on  $GP(s, a)$  where  $\pi(s) = \operatorname{argmax}_a Q(s, a)$ 
9:   if  $QUEUE.length == m$  then
10:      $QUEUE.popFirst()$ 
11:   Add  $\pi$ 's decisions on  $\tilde{S}$  to  $QUEUE$ 
12:   if  $QUEUE$  is full and all elements are identical then
13:     Converged  $\leftarrow$  True
14:   Update  $(\tilde{S}, \tilde{A}; \tilde{Q})$  following the bellman equation and using the current GP model, setting  $\dot{Q}(s_i, a_i) = r(s_i, a_i) + \max_{a' \in A} GP(s_{i+1}, a')$ 
15: return  $\pi$ 
```

---

evaluated on approximation points, and  $K(\mathbf{X}, \mathbf{X}')$  is a covariance matrix evaluated on observation and approximation points [Ras04].

### 6.2.5 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 the RL study, 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 achieved superhuman performance in many complex tasks, including Atari games [Mni15], Go [Sil16], Chess/Shogi [Sil18], Starcraft II [Vin19], and robotic control [And18]. 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 applied 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 [Mni15], Double DQN [VH16] or Dueling DQN [Wan15]; policy-based methods such as Trust Region Policy Optimization [Sch15b] or Proximal Policy Optimization [Sch17a]; and Actor-

Critic methods such as Deep Deterministic Policy Gradients [Lil15] or Soft Actor-Critic [Haa18a]. Each of these methods has its own advantages and drawbacks. In this work, we used the offline Double-DQN algorithm [VH16] with prioritized experience replay [Sch15a]. More specifically, it used a multi-layer perceptron neural network to approximate the Q-function. The inputs to the neural network were the last 3 step observations of a student and the outputs were 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 were used to stabilize the training. The data and immediate rewards used for DQN policy induction were identical to those used for HRL.

## 6.3 Evaluation Experiment

### 6.3.1 Conditions and Participants

The effectiveness of the HRL policy was evaluated in a classroom study where it was compared to a DQN induced step-level policy and a random yet reasonable baseline step-level policy. We choose step level decisions for the two flat policies because WE and PS can be viewed as two extreme cases of CPS, with WE as an all-tell step-level policy, and PS being an all-elicited step-level policy. In addition, since both *elicit* and *tell* are always considered to be reasonable educational interventions in our learning context, our baseline policy is “*random yet reasonable*”.

Similar to the prior studies, participants in the study were undergraduate students enrolled in the discrete math class in the Fall 2018 semester where the study was given as one of the regular homework assignments. Again they had one week to complete the study and were graded (in class) based on their demonstrated effort rather than performance. 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 study, covered the same content and used the same ITS.

### 6.3.2 Procedure and Measures

Similar to previous studies, all students went through four phases: 1) textbook, 2) pre-test, 3) training on the ITS, and 4) post-test, as shown in Table 3.1. During **textbook**, all students read a general description of each principle, reviewed some examples, and solved some training problems. The students then took a **pre-test**, which contained a total of 14 single- and multiple-principle 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 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. 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. Conditions differed only in the pedagogical policy applied (HRL vs. DQN vs. random). The pre- and post-test were graded by experienced graders following the partial credit rubric in a double-blind manner. More details about the experimental procedure are described in Chapter 2.

**Table 6.2** Procedure of the HRL study

Procedure	Conditions		
	HRL	DQN	Baseline
Textbook	Read a probability textbook		
Pre-test	Complete 14 test questions (10 single- and 4 multiple-principle)		
ITS-training	HRL WE/PS/CPS - elicit/tell	DQN elicit/tell	random elicit/tell
Post-test	Complete 20 test questions (10 single- and 10 multiple-principle)		

After the study, we performed statistical analyses on students' learning performance, time on task and the policies' pedagogical decision-making. Specific learning performance measures are listed below.

- **Pre-test**: calculated based on the 14 pre-test questions;
- **Isomorphic Post-test**: calculated based on the 14 post-test questions that are isomorphic to the pre-test;
- **Full Post-test**: calculated based on the 20 post-test questions;
- **Adjusted Post-test**: calculated following the equation:  $adjusted = post - k * (pre - \overline{pre})$ , where  $k$  is the coefficient of the linear regression model  $post = k * pre + b$  generated by ANCOVA analysis, and  $\overline{pre}$  is the average pre-test score across all students.
- **Isomorphic NLG**: calculated based on the pre- and isomorphic post-test score:  $\frac{(isomorphic\_posttest) - pretest}{\sqrt{1 - pretest}}$ ;
- **NLG**: calculated based on the pre- and full post-test score:  $\frac{posttest - pretest}{\sqrt{1 - pretest}}$ .

**Table 6.3** HRL Study Results

Condition	Improvement			Learning Performance			Time (h)
	Pre	Iso Post	Iso NLG	Full Post	Adj Post	NLG	
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)

Similar to the granularity and RL studies, the time students spent on the training task (time on task) was calculated based on an analysis of the system logs with the idle time excluded. We still used the 141-second threshold for determining idle intervals.

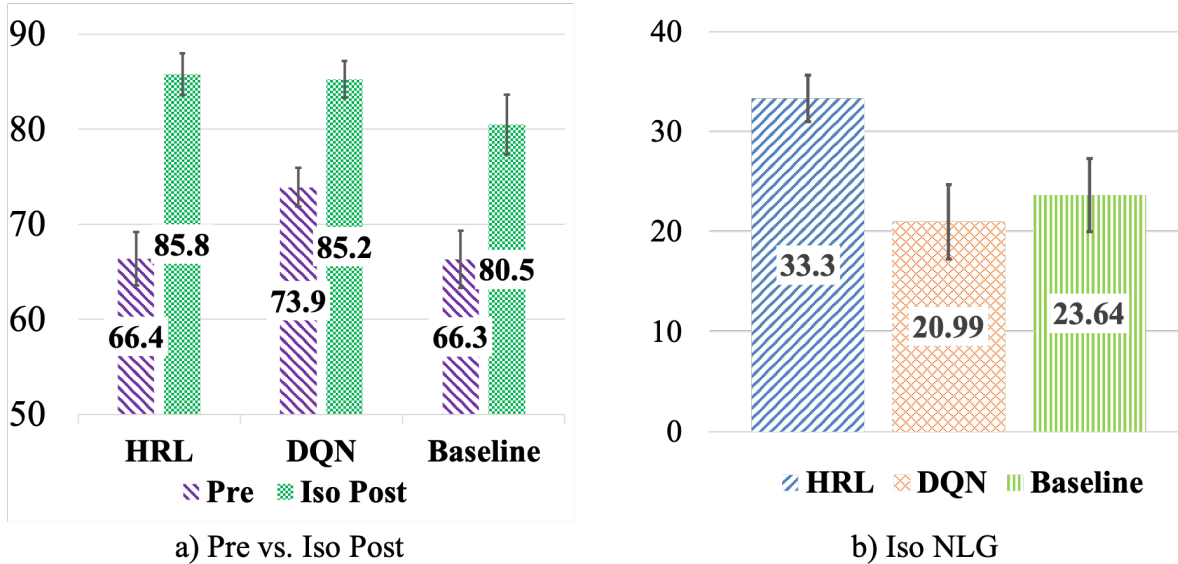
### 6.3.3 Results

**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 condition, it did not do so for the DQN condition. Therefore, we mainly focus on comparing learning performance measures that consider the pre-test differences, that is, the adjusted post-test, isomorphic NLG, and NLG especially the last because it is the reward we used for policy induction.

Table 6.3 shows the mean and standard deviation (SD) of students' learning performance and time on task results across three conditions. From left to right, it shows the conditions with the number of students in the 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 with their pre-test, as shown in Figure 6.2.a. 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 of 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 that our ITS training is effective, regardless of how the pedagogical decisions were made.

*Isomorphic NLG:* The Isomorphic NLG also showed that all three conditions achieved a great



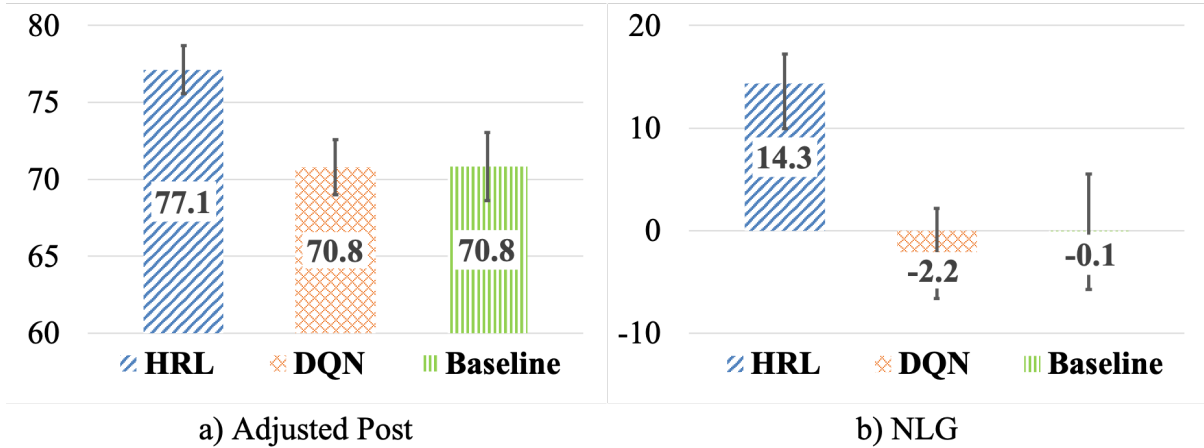
**Figure 6.2** Students' Improvement Through Training

improvement through training, with the lowest one scoring 21.0, as shown in Figure 6.2.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:  $t(125) = 2.06$ ,  $p = 0.042$ ,  $d = 0.50$ , but there was no significant difference between DQN and Baseline. The results suggest that the HRL policy is significantly more effective than the DQN and Baseline policies.

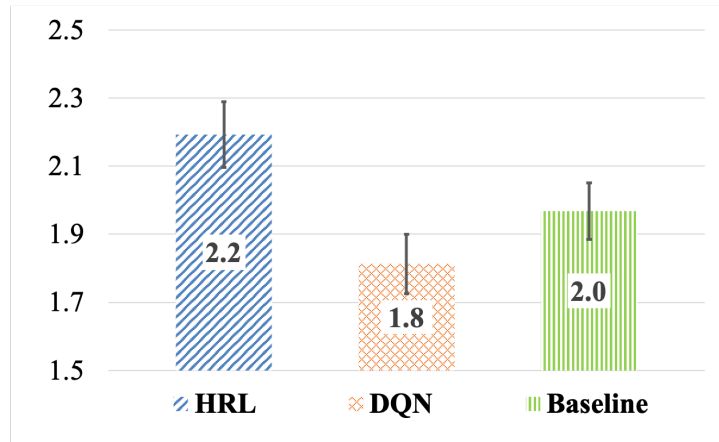
**Learning Performance: Adjusted Post-test:** To comprehensively evaluate students' learning performance, we conducted analyses on their post-test and NLG. An ANCOVA analysis on the full post-test using the pre-test score as a covariate showed a significant difference among the three conditions:  $F(2, 124) = 3.86$ ,  $p = 0.024$ ,  $\eta = 0.030$ . Figure 6.3.a shows a comparison of the adjusted post-test scores across the three conditions. Contrast analysis on the adjusted post-test scores showed that the HRL condition scored significantly higher than the DQN condition:  $t(125) = 2.53$ ,  $p = 0.013$ ,  $d = 0.57$  and the Baseline condition:  $t(125) = 2.36$ ,  $p = 0.020$ ,  $d = 0.52$ . 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 6.3.b. 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 random Baseline policy.

**Time on Task:** A one-way ANOVA analysis showed a significant difference among the three conditions:  $F(2, 125) = 4.74$ ,  $p = 0.010$ ,  $\eta = 0.071$ , as shown in Figure 6.4. More specifically, the HRL



**Figure 6.3** Students' Learning Performance



**Figure 6.4** Students' Time on Task

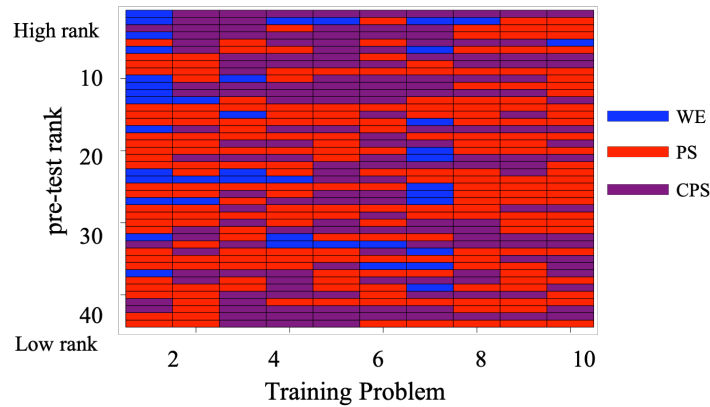
condition ( $M = 2.19$ ,  $SD = .64$  in hours) spent significantly more time than the DQN condition ( $M = 1.81$ ,  $SD = .58$ ):  $t(125) = 3.07$ ,  $p = 0.003$ ,  $d = 0.62$  and marginally significantly more time than the Baseline condition ( $M = 1.97$ ,  $SD = .52$ ):  $t(125) = -1.75$ ,  $p = 0.082$ ,  $d = 0.39$ .

**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 6.5 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 made personalized decisions.

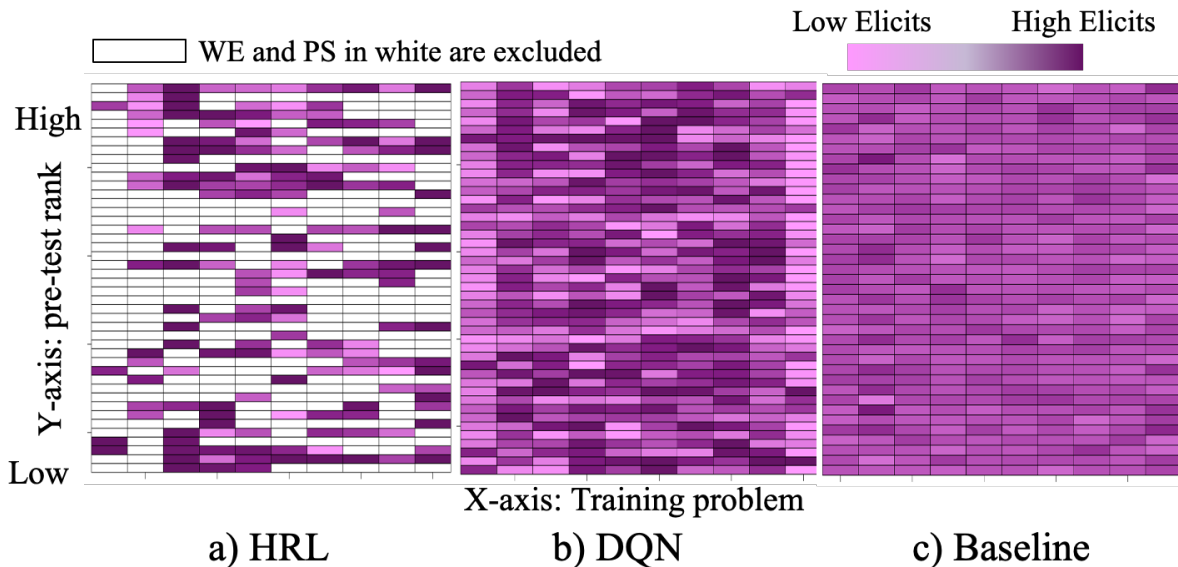
The step-level analysis focuses on the elicits and tells students received. To conduct an analysis across the three conditions, we transferred problem level WE and PS into step level tells and elicits. Table 6.4 shows the number of step-level decisions students received across the three conditions.

**Table 6.4** 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)



**Figure 6.5** Problem level decisions for the HRL policy, with 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 ranks, with the highest on the top.



**Figure 6.6** Percentage of Elicit for the a. HRL, b. DQN and c. Baseline policies' step-level decisions on each problem. The higher the percentage, the darker a cell is. Problem-level WEs and PS 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 ranks, with the highest on the top.

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 policy 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 6.6 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. Note that 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 PS and 3 (1.7%) turned into WEs. For the DQN policy, 31 (6.9%) out of 450 CPS turned into PS and 19 (4.2%) turned into WEs. But the random policy did not turn any CPS into WE or PS.

## 6.4 Conclusion and Discussion for Chapter 6

In this study, we applied an offline, off-policy GP-based HRL approach to induce a hierarchical pedagogical policy that makes decisions first at the problem level and then at the step level. At the problem level, it decides whether the next problem should be WE, PS, or CPS. If CPS is selected, a corresponding step-level policy will be activated to decide whether to elicit or tell each step. In a classroom study, we evaluated the effectiveness of the HRL policy by comparing it 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 and Baseline policy, but there was no significant difference between DQN and Baseline. For time on task, there was no significant difference between the HRL and Baseline condition, but the former spent significantly more time than the DQN condition. Finally, log analysis showed that the HRL policy was more likely to choose PS and CPS than WE. Additionally, log analysis suggested that HRL and DQN made more personalized decisions than the random Baseline policy.

The results supported our hypothesis that making adaptive decisions at both levels of granularity can be more effective than focusing on a single level. This is in line with the results from the granularity studies that Both-level decisions benefited more students than either the problem- or step-level. The HRL study results can also be explained by our hypothesis that WE, PS, and CPS involve different learning mechanisms. More specifically, by making both-level decisions, the HRL policy is able to explicitly select the most appropriate activity for students at the problem level, and,

if necessary, fine tailor the intervention at the step level; but the step-level DQN policy was not able to explicitly select the problem-level activity..

From the machine learning perspective, the fact that HRL has an explicit problem-level vision may explain why it can be more effective than flat RL in pedagogical policy induction. At the problem level, HRL views a problem as an atomic action, and this abstraction has two potential advantages: 1) it aggregates the effects of all steps in a problem and 2) it converts a long step-level sequence into a short problem-level sequence. The aggregation of steps across a problem may provide HRL with a better estimation of the effect of taking a series of steps; while the problem sequence may give HRL a better view of the long-term effects of each problem. Theoretically, flat RL could learn the impact of a problem by aggregating step-level information, but there is no guarantee that it would. 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.

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 activities.

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 the 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 was 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 for the long term learning. 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.

## CHAPTER

# 7

# GENERAL DISCUSSION, CONCLUSION, AND FUTURE WORK

My dissertation consists of three major studies: 1) the granularity studies that explicitly investigated the impact of decision granularity on student learning, 2) the reinforcement learning (RL) study that investigated the effectiveness of RL-induced problem- and step-level policies, and 3) the hierarchical reinforcement learning (HRL) study that applied HRL for pedagogical policy induction and evaluated its effectiveness. This chapter presents a general discussion across the studies, the conclusion, and future work.

## **7.1 The Impact of Granularity on Student Learning**

The granularity studies explored the impact of decision granularity on student learning by comparing three granularity types: problem-level only (Prob-Only), step-level only (Step-Only) and both problem- and step-levels (Both). A series of four classroom studies were conducted for the exploration. Overall, results showed that Prob-Only can be more effective for students with a lower incoming competence (Low), Step-Only can be more effective for students with a higher level of incoming competence (High), and Both is effective for students of either incoming competence. In terms of time on task, the Step-Only condition spent significantly less time than the Prob-Only condition.

The studies contribute to cognitive science in that it made the first attempt to explicitly investigate the impact of granularity on student learning. While problem-level worked example (WE) / problem solving (PS) and step-level elicit / tell were widely used in prior studies, they were often

provided following pre-determined fixed or adaptive pedagogical policies. Since both decision granularity and pedagogical policy may impact student learning, it is not clear which one actually led to the learning outcomes. In our granularity studies, random policies were employed to factor out the impact of pedagogical policy, allowing us to directly observe the impact of granularity.

The second contribution of the granularity studies is that it extended the current understanding of problem-level WE/PS and step-level elicit/tell. Our results showed that problem-level WE/PS and step-level elicit/tell can have different impacts on student learning. This informs researchers and educators that problem- and step-level decisions should be viewed differently in pedagogical decision-making. One possible reason that elicit and tell are different from PS and WE is that they involve a different information processing procedure. From the problem solving perspective, steps inside a problem are logically connected, while problems are independent of each other. Thus, when elicit and tell are interleaved, students need to integrate the tutor's solution with their solution, but such integration is not needed between WEs and PS. This suggests that information dependency and integration should be considered in pedagogical decision making. However, this is only our hypothesis, more research is needed to understand the exact difference between problem- and step-level granularities.

Researchers often use the cognitive load theory [Swe98] to explain the impact of WEs or tells, but this theory does not explain out results perfectly. Under this theory, WEs have the least problem solving load, PS has the most, and CPS is in the middle. As a result, the problem-level granularity switches between light and heavy load, while the step-level granularity keeps a moderate amount of load. From the students' perspective, it would be difficult for students with a lower incoming competence to benefit from problem solving; its heavier cognitive load would occupy most of their cognitive capacity, leaving them with only a small amount remaining for knowledge acquisition. The step-level granularity, on the other hand, would benefit them more because the problem solving load is reduced. However, our results showed the opposite in that the problem-level granularity benefited Low students more and step-level benefited High students more. One possible explanation is that it requires extra cognitive capacity to integrate the tutor's solution with students' own solution, which causes CPS to require more cognitive capacity than PS. As a result, in terms of cognitive load demand, we have the order:  $WE < PS < CPS$ . However, this is only our hypothesis and much more research is needed to explore the possible underlying cognitive procedures.

Our results showed that the step-level granularity benefited High students more than the problem-level. A possible explanation is that High students learned from integrating the tutor's solution with their own solution. This is in line with Chi's interaction-constructive-active-passive (ICAP) theory [Chi09], which states that collaborative work is beneficial because it gives the student another information source. Collaborators' information provides students with extra opportunities to infer new knowledge, integrate new information with existing knowledge, as well as organize, repair and restructure their own knowledge. Our findings may extend this theory by suggesting that effective collaborative work requires adequate domain knowledge. However, much more research is needed to verify this hypothesis.

Note that the above statement which states that effective collaborative work requires adequate domain knowledge does not mean interactive learning is not beneficial for Low students. The ICAP theory specified different types of interactivity, such as interacting with an expert and interacting with a peer. Our CPS can be viewed as interacting with a peer because the student and the tutor relied on each others' inputs to co-construct the solution. In this sense, our results suggest that the interactivity of co-constructing the solution with peers may not always be effective for Low students. It does not suggest that interacting with an expert may be ineffective.

As the underlying mechanism for the impact of decision granularity remains unknown, a future direction is to study the mechanism based on existing cognitive science theory. For example, based on the cognitive load theory, we can investigate whether CPS indeed demands more cognitive capacity. Based on the Zone of Proximal Development (ZPD) theory [Cha03], we can investigate whether the problem- and step-level granularities are the development zone for Low and High students respectively. Based on the ICAP theory, we can investigate which type of interactivity is effective/ineffective for Low/High students.

One limitation of the granularity study is that its impact on Low and High students was not evaluated in a single empirical study. In our analysis, we combined the data collected from four studies and split students into High, Medium and Low groups based on their pre-test score. Although all four studies employed an identical procedure and used the same materials, other factors, such as the schedule of the class and the schedule of students, may still impact students' test scores and learning outcomes. Therefore, conducting a single study to investigate the interaction between the impact of granularity and students' incoming competence would be a more strict design.

## **7.2 The Effectiveness of RL Induced Problem- vs. Step-Level Policies**

In the RL study, we investigated the effectiveness of RL induced problem- and step-level policies. We applied the same RL approach to induce a problem- and a step-level policy, and then compared them with a problem- and a step-level random baseline policy in a classroom study. Results showed that there was no significant difference between the two RL-induced policies and none of them significantly outperformed the two random baseline policies.

The RL study contributes to current RL pedagogical policy induction research in that it made the first attempt to compare RL induced problem- and step-level policies. While a lot of prior works have applied RL for pedagogical policy induction, none of them has investigated the impact of decision granularity on the effectiveness of RL induced policies. Results in this study suggest that RL induced pedagogical policies that make decisions at a single granularity level may not always be effective.

The result that the RL-induced problem- and step-level policies were not effective may be explained by our hypothesis that problem- and step-level granularities involve different learning mechanisms. In particular, when learning with the problem-level granularity, students switch between observing (WE) and doing (PS); when learning with the step-level, they co-construct the

solution with the tutor (CPS). Each of these activities can be effective in some states and ineffective in some others. As a result, if the policy makes decisions at a single granularity level, it may not always be able to provide students with the most appropriate intervention. However, more research is needed to reveal the actual underlying mechanisms.

### 7.3 HRL for Pedagogical Policy Induction

Given that RL induced problem- and step- level policies may not always be effective, and motivated by the results from granularity studies that both-level decisions can benefit more students than either the problem- or step-level, we then applied HRL to induce hierarchical policies. The HRL policy first makes a problem-level decision to decide whether the next problem should be WE, PS or CPS. If WE is selected, an all-tell step-level policy will be carried out; if PS is selected, an all-elicited policy will be executed; finally, if CPS is selected, the tutor will decide whether to elicit or tell each step based on the corresponding HRL induced step-level policy. In an empirical evaluation study, the HRL policy was compared 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 and the random baseline policy.

The contribution of the HRL study is that it made the first attempt to apply HRL for pedagogical policy induction. While much prior research has applied RL for pedagogical policy induction, most of them treat all system decisions equally or independently, disregarding the long-term impact that tutor decisions may have across different levels of granularity. In our HRL approach, problem- and step-level decisions are treated differently, allowing the agent to consider their impact explicitly and separately.

The second contribution is that it provided empirical evidence that HRL can be more effective than flat RL in pedagogical policy induction. For machine learning research, it showed the advantage of HRL in one more area, while for cognitive science research, it supported our hypothesis that taking decision granularity into account can result in more effective decision-making.

The results that the HRL policy outperformed the DQN policy can be explained by our hypothesis that WE, PS and CPS involve different learning mechanisms. As mentioned before, making both-level decisions allow the agent to select the most appropriate learning activity for students, which can hardly be done using a single level of granularity.

The HRL study results are in line with the prior findings that adaptive decision-making can improve student learning [Naj14; Sal10]. 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. 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 actual 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 [Naj14; Sal10].

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 effect. Results from the granularity studies suggest that step-level decisions can be more effective for High competence students, while problem-level decisions can be 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 students with high pre-test scores 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 high competence at the beginning of training may not retain that level of competence after several problems and similarly, those who had low competence in the early stage of training may not still have a lower competence in later stages. Additionally, the training was done with changing content. In the training phase, the 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.

A limitation of the HRL study is that the HRL and DQN induced policies are not interpretable. Both policies are represented by a complicated non-linear model with a lot of parameters, which performs a lot of computation to make each decision. A benefit of complex models is that they have greater representation power, which makes it more likely to result in effective policies. However, it can be difficult to extract simple and human-understandable decision-making rules from models with a complex structure. As a result, it is hard to tell how the features are used in decision-making. Moreover, since HRL and DQN used different function approximators with different structure, it is hard to directly compare these two policies and examine how they differ in decision-making.

The second limitation is that we did not compare our data-driven HRL policy with existing learning theory based policies, such as the one designed by Najjar et al. [Naj14]. One reason that we did not include such policies in our HRL study is that they often require a different procedure. These policies usually make decisions based on students' cognitive state, which may require extra measurements. Since our training data is collected with a procedure that does not have such measurements, currently, we cannot directly compare a HRL policy with a learning theory based policy using the same procedure.

## 7.4 Conclusion and Future Work

My dissertation involves three major studies. The granularity studies investigated the impact of three types of granularity on student learning: 1) problem-level only (Prob-Only); 2) step-level only (Step-Only) and 3) both the problem- and step-level (Both). Results showed that problem-level is more effective for Low incoming competence students, step-level is more effective for High ones, and Both can be effective for both Low and High students. In terms of time on task, students in the step-level condition spent more time on training than those in the problem-level condition. The results suggest that granularity indeed can have an impact on student learning and its impact differs for students with different incoming competence.

Motivated by the results from the granularity studies, we then investigated the effectiveness of reinforcement learning (RL)-induced problem- and step-level policies by comparing them with each other and with two random baseline policies: one problem-level and one step-level. Results showed that there was no significant difference between the two RL-induced policies and none of them significantly outperformed the two random baselines. This suggests that RL induced policies that make decisions at a single level of granularity may not always be effective.

Given the RL study results, we then applied a Gaussian Process (GP) based HRL approach to induce a policy that makes decisions first at the problem-level and then at the step-level. In a classroom study, the HRL policy was compared with a Deep Q-Network (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 and the random baseline policy. This suggests that taking decision granularity into account in RL policy induction indeed results in more effective policies.

One future direction is to investigate the underlying cognitive processes behind each granularity type. Our study showed that decision granularity can have an impact on student learning, but there are still many questions that remain open. Why does decision granularity cause an impact on student learning? Why is problem-level WE/PS different from step-level elicit/tell? Why does problem-level granularity benefit students with lower incoming competence? Similarly, why does step-level granularity benefit students that have a higher level of incoming competence? Though, more importantly, why is it that both levels of granularity benefit students of either competence level? Does working on step-level granularity require more cognitive capacity, or are there other mechanisms that make it harder?

For HRL, one future direction is to involve more granularity levels and instructional interventions. My Ph.D. work mainly focused on the problem- and step-level granularities, but there are more that can be involved, such as the topic level (multiple problems) and the subgoal level (multiple steps). For instructional interventions, the HRL study focused on problem level WE/PS and step level elicit/tell. It remains an open question on whether HRL can induce effective policies for other types of interventions.

Another future direction is to explain the policy's decisions. As discussed above, the HRL policy is represented by complex non-linear models, which makes it hard to extract simple and human-

understandable decision-making rules from the policy. A possible way to explain the HRL policy is to train other explainable models (such as decision trees) based on the HRL policy's decision-making, and then rely on the explainable models to analyze and interpret the HRL policy's decision-making. Explaining the HRL policy's decisions allows us to examine where its effectiveness comes from, which can potentially be used to advance existing learning theories. It can also help us catch the points where the HRL policy's decisions are ineffective and in turn, improve the policy induction process.

One more future direction is to compare our HRL policy with state-of-the-art theory-based policies. In the RL and HRL studies, we compared different data-driven policies. However, we have never compared data-driven policies with theory-driven policies. Comparing these two types of policies would reveal how effective current data-driven and theory-driven policies can be. Additionally, observing and comparing their impacts on student learning allows us to discover potential ways to improve them.

## BIBLIOGRAPHY

- [And93] Anderson, J. R. “Problem solving and learning.” *American Psychologist* **48.1** (1993), p. 35.
- [And95] Anderson, J. R. et al. “Cognitive tutors: Lessons learned”. *The journal of the learning sciences* **4.2** (1995), pp. 167–207.
- [And18] Andrychowicz, M., Baker, B., et al. “Learning dexterous in-hand manipulation”. *arXiv preprint arXiv:1808.00177* (2018).
- [Azi18] 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 (2018), pp. 146–165.
- [Azi19] Azizsoltani, H. et al. “Unobserved Is Not Equal to Non-existent: Using Gaussian Processes to Infer Immediate Rewards Across Contexts”. *IJCAI* (2019), pp. 1974–1980.
- [Bar03] Barto, A. G. & Mahadevan, S. “Recent advances in hierarchical reinforcement learning”. *Discrete event dynamic systems* **13.1-2** (2003), pp. 41–77.
- [Bec00] Beck, J. et al. “ADVISOR: A machine learning architecture for intelligent tutor construction”. *AAAI/IAAI 2000* (2000), pp. 552–557.
- [Cha03] 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** (2003), pp. 39–64.
- [Chi09] Chi, M. T. “Active-constructive-interactive: A conceptual framework for differentiating learning activities”. *Topics in cognitive science* **1.1** (2009), pp. 73–105.
- [Chi11] Chi, M. et al. “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** (2011), pp. 137–180.
- [Cle16] Clement, B. et al. “A Comparison of Automatic Teaching Strategies for Heterogeneous Student Populations”. *EDM, 2016*. 2016.
- [Cro77] Cronbach, L. J. & Snow, R. E. *Aptitudes and instructional methods: A handbook for research on interactions*. Irvington, 1977.
- [Cua10] Cuayáhuitl, H. et al. “Generating adaptive route instructions using hierarchical reinforcement learning”. *International Conference on Spatial Cognition*. Springer. 2010, pp. 319–334.
- [Dor17] Doroudi, S. et al. “Robust evaluation matrix: Towards a more principled offline exploration of instructional policies”. *Proceedings of the fourth (2017) ACM conference on learning@ scale*. 2017, pp. 3–12.
- [Eat83] Eaton, M. L. “Multivariate statistics: a vector space approach.” *JOHN WILEY & SONS, INC., 605 THIRD AVE., NEW YORK, NY 10158, USA, 1983, 512* (1983), pp. 116–117.

- [Eve06] Evens, M. & Michael, J. *One-on-one tutoring by humans and computers*. Psychology Press, 2006.
- [Fel08] Feller, W. *An introduction to probability theory and its applications*. Vol. 2. John Wiley & Sons, 2008.
- [Flo17] Florensa, C. et al. “Stochastic neural networks for hierarchical reinforcement learning”. *arXiv preprint arXiv:1704.03012* (2017).
- [Gol98] Goldberg, P. W., Williams, C. K., et al. “Regression with input-dependent noise: A Gaussian process treatment”. *NIPS*. 1998, pp. 493–499.
- [Guo05] Guo, D. et al. “Mutual information and minimum mean-square error in Gaussian channels”. *IEEE Transactions on Information Theory* **51.4** (2005), pp. 1261–1282.
- [Haa18a] Haarnoja, T., Zhou, A., et al. “Soft Actor-Critic Algorithms and Applications”. *arXiv:1812.05905* (2018).
- [Haa18b] Haarnoja, T. et al. “Latent space policies for hierarchical reinforcement learning”. *arXiv preprint arXiv:1804.02808* (2018).
- [Igl09a] Iglesias, A. et al. “Learning teaching strategies in an adaptive and intelligent educational system through reinforcement learning”. *Applied Intelligence* **31.1** (2009), pp. 89–106.
- [Igl09b] Iglesias, A. et al. “Reinforcement learning of pedagogical policies in adaptive and intelligent educational systems”. *Knowledge-Based Systems* **22.4** (2009), pp. 266–270.
- [Kal10] Kalyuga, S. & Renkl, A. “Expertise reversal effect and its instructional implications: Introduction to the special issue”. *Instructional Science* **38.3** (2010), pp. 209–215.
- [Kao18] Kao, H.-C. et al. “Context-aware symptom checking for disease diagnosis using hierarchical reinforcement learning”. *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.
- [Kul16] Kulkarni, T. D. et al. “Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation”. *Advances in neural information processing systems*. 2016, pp. 3675–3683.
- [Laj13] Lajoie, S. P. & Derry, S. J. “Motivational techniques of expert human tutors: Lessons for the design of computer-based tutors”. *Computers as cognitive tools*. Routledge, 2013, pp. 83–114.
- [Lil15] Lillicrap, T. P., Hunt, J. J., et al. “Continuous control with deep reinforcement learning”. *arXiv preprint arXiv:1509.02971* (2015).
- [Liz06] Liz, B. et al. “Exemplification in mathematics education”. *Proceedings of the 30th Conference of the International Group for the Psychology of Mathematics Education*. Vol. 1. ERIC. 2006, pp. 126–154.

- [Man14] Mandel, T. et al. “Offline policy evaluation across representations with applications to educational games”. *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems. 2014, pp. 1077–1084.
- [McL11] McLaren, B. M. & Isotani, S. “When is it best to learn with all worked examples?” *International Conference on Artificial Intelligence in Education*. Springer. 2011, pp. 222–229.
- [McL08] McLaren, B. M. et al. “When and how often should worked examples be given to students? New results and a summary of the current state of research”. *Proceedings of the 30th annual conference of the cognitive science society*. 2008, pp. 2176–2181.
- [McL14] McLaren, B. M. et al. “Exploring the Assistance Dilemma: Comparing Instructional Support in Examples and Problems”. *Intelligent Tutoring Systems*. Springer. 2014, pp. 354–361.
- [Mni15] Mnih, V. et al. “Human-level control through deep reinforcement learning”. *Nature* **518**.7540 (2015), p. 529.
- [Naj14] Najar, A. S. et al. “Adaptive Support versus Alternating Worked Examples and Tutored Problems: Which Leads to Better Learning?” *User Modeling, Adaptation, and Personalization*. Springer, 2014, pp. 171–182.
- [Pen17] Peng, X. B. et al. “Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning”. *ACM Transactions on Graphics (TOG)* **36**.4 (2017), p. 41.
- [Pho10] Phobun, P. & Vicheanpanya, J. “Adaptive intelligent tutoring systems for e-learning systems”. *Procedia-Social and Behavioral Sciences* **2**.2 (2010), pp. 4064–4069.
- [Raf16] Rafferty, A. N. et al. “Faster teaching via pomdp planning”. *Cognitive science* **40**.6 (2016), pp. 1290–1332.
- [Ras04] Rasmussen, C. E. “Gaussian processes in machine learning”. *Advanced lectures on machine learning*. Springer, 2004, pp. 63–71.
- [Ren02] Renkl, A. et al. “From example study to problem solving: Smooth transitions help learning”. *The Journal of Experimental Education* **70**.4 (2002), pp. 293–315.
- [Row14] Rowe, J. et al. “Optimizing player experience in interactive narrative planning: a modular reinforcement learning approach”. *Tenth Artificial Intelligence and Interactive Digital Entertainment Conference*. 2014.
- [Row15] Rowe, J. P. & Lester, J. C. “Improving student problem solving in narrative-centered learning environments: A modular reinforcement learning framework”. *International Conference on Artificial Intelligence in Education*. Springer. 2015, pp. 419–428.
- [Rya00] Ryan, M. & Reid, M. “Learning to fly: An application of hierarchical reinforcement learning”. *In Proceedings of the 17th International Conference on Machine Learning*. Citeseer. 2000.

- [Sal10] Salden, R. J. et al. “The expertise reversal effect and worked examples in tutored problem solving”. *Instructional Science* **38.3** (2010), pp. 289–307.
- [Sch15a] Schaul, T. et al. “Prioritized experience replay”. *arXiv preprint arXiv:1511.05952* (2015).
- [Sch15b] Schulman, J. et al. “Trust region policy optimization”. *International conference on machine learning*. 2015, pp. 1889–1897.
- [Sch17a] Schulman, J. et al. “Proximal policy optimization algorithms”. *arXiv preprint arXiv:1707.06347* (2017).
- [Sch17b] Schwab, D. & Ray, S. “Offline reinforcement learning with task hierarchies”. *Machine Learning* **106.9-10** (2017), pp. 1569–1598.
- [Sch09] Schwonke, R. et al. “The worked-example effect: Not an artefact of lousy control conditions”. *Computers in Human Behavior* **25.2** (2009), pp. 258–266.
- [She16a] Shen, S. & Chi, M. “Aim Low: Correlation-based Feature Selection for Model-based Reinforcement Learning”. *EDM* (2016).
- [She16b] Shen, S. & Chi, M. “Reinforcement Learning: the Sooner the Better, or the Later the Better?” *UMAP*. ACM. 2016, pp. 37–44.
- [She18] Shen, S. et al. “Improving Learning & Reducing Time: A Constrained Action-Based Reinforcement Learning Approach”. *Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization*. ACM. 2018, pp. 43–51.
- [Shi11] Shih, B. et al. “A response time model for bottom-out hints as worked examples”. *Handbook of educational data mining* (2011), pp. 201–212.
- [Sil16] Silver, D. et al. “Mastering the game of Go with deep neural networks and tree search”. *nature* **529.7587** (2016), pp. 484–489.
- [Sil18] Silver, D. et al. “A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play”. *Science* **362.6419** (2018), pp. 1140–1144.
- [Sno91] Snow, R. E. “Aptitude-treatment interaction as a framework for research on individual differences in psychotherapy.” *Journal of consulting and clinical psychology* **59.2** (1991), p. 205.
- [Sta11] Stamper, J. C. et al. “Experimental evaluation of automatic hint generation for a logic tutor”. *International Conference on Artificial Intelligence in Education*. Springer. 2011, pp. 345–352.
- [Sut99] Sutton, R. S. et al. “Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning”. *Artificial intelligence* **112.1-2** (1999), pp. 181–211.
- [Swe85] Sweller, J. & Cooper, G. A. “The use of worked examples as a substitute for problem solving in learning algebra”. *Cognition and Instruction* **2.1** (1985), pp. 59–89.

- [Swe98] Sweller, J. et al. “Cognitive architecture and instructional design”. *Educational psychology review* **10.3** (1998), pp. 251–296.
- [Swe95] Swetz, F. “To know and to teach: Mathematical pedagogy from a historical context”. *Educational Studies in Mathematics* **29.1** (1995), pp. 73–88.
- [Swe87] 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.
- [VG11] Van Gog, T. et al. “Effects of worked examples, example-problem, and problem-example pairs on novices’ learning”. *Contemporary Educational Psychology* **36.3** (2011), pp. 212–218.
- [VH16] Van Hasselt, H. et al. “Deep Reinforcement Learning with Double Q-Learning.” *AAAI*. Vol. 2. Phoenix, AZ. 2016, p. 5.
- [Van06] Vanlehn, K. “The behavior of tutoring systems”. *International journal of artificial intelligence in education* **16.3** (2006), pp. 227–265.
- [Vez17] Vezhnevets, A. S. et al. “Feudal networks for hierarchical reinforcement learning”. *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR.org. 2017, pp. 3540–3549.
- [Vin19] Vinyals, O. et al. “Grandmaster level in StarCraft II using multi-agent reinforcement learning”. *Nature* **575** (2019), p. 350.
- [Wan17] Wang, P. et al. “Interactive narrative personalization with deep reinforcement learning”. *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*. 2017.
- [Wan18] Wang, X. et al. “Video captioning via hierarchical reinforcement learning”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4213–4222.
- [Wan15] Wang, Z. et al. “Dueling network architectures for deep reinforcement learning”. *arXiv preprint arXiv:1511.06581* (2015).
- [Wil08] Williams, J. D. “The best of both worlds: unifying conventional dialog systems and POMDPs.” *INTERSPEECH*. 2008, pp. 1173–1176.
- [Zha19] Zhang, J. et al. “Hierarchical Reinforcement Learning for Course Recommendation in MOOCs”. *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 435–442.
- [Zho17a] Zhou, G. & Chi, M. “The Impact of Decision Agency & Granularity on Aptitude Treatment Interaction in Tutoring”. *CogSci*. 2017, pp. 3652–3657.
- [Zho15] Zhou, G. et al. “The Impact of Granularity on Worked Examples and Problem Solving”. *CogSci*. 2015, pp. 2817–2822.

- [Zho16] Zhou, G. et al. “The Impact of Granularity on the Effectiveness of Students’ Pedagogical Decisions”. *CogSci*. 2016, pp. 2801–2806.
- [Zho17b] Zhou, G. et al. “Towards Closing the Loop: Bridging Machine-induced Pedagogical Policies to Learning Theories”. *The 10th International Conference on Educational Data Mining*. 2017, pp. 112–119.
- [Zho19] Zhou, G. et al. “Hierarchical Reinforcement Learning for Pedagogical Policy Induction”. *International Conference on Artificial Intelligence in Education*. 2019.