ABSTRACT

CODY, CHRISTA NICOLE. Applied Analytics and Machine Learning to Improve and Inform Adaptive Assistance in a Tutoring System. (Under the direction of Tiffany Barnes.)

This work applies analytics and machine learning to improve problem solving through proactive assistance in an Intelligent Tutoring System (ITS). Proactive assistance, where the ITS decides to provide students with a hint when it is most needed, has shown to have significant positive effects on learning outcomes within ITSs compared to unsupported problem solving. Intelligent tutoring systems typically achieve this support using an expert-defined set of rules, a set of constraints, or a data-driven system to provide feedback and hints to help students achieve a particular problem-solving step. This is called an ITS's inner loop, which provides support during the steps of a single problem, while the outer loop selects problems for the student to solve. A crucial component of effective tutoring includes both determining when intervention is needed and adapting the feedback or hints to specific learner's needs and determining. Although research has explored how to generate hints, and whether they can improve learning, there is little research that explores the impact of different hint content or timing. In other words, while step-level hints can be expert-authored or automatically generated from data, few studies have varied the content or level of student choice in the timing or presentation of hints. Furthermore, there is very little research exploring how to use reinforcement learning (RL) to induce effective proactive hint policies to decide what type of hint to provide and when for open-ended problem solving.

In this dissertation, I present three studies exploring different aspects of what makes proactive assistance effective using the data-driven Deep Thought Intelligent Tutoring System for logic proofs. In the Hint Content Study 1, I explore hint content, adding a new, higher-level hint generator to Deep Thought and comparing its impact on learning with data-driven next-step hints. In the Autonomy Study 2, I explore how the need for autonomy may affect how students respond to proactive intervention, in terms of learning outcomes and hint usage behavior. The results of these two studies demonstrate the effectiveness of the hint type chosen and the positive receptiveness of proactive intervention by students. In the RL Study 3, I investigate the effectiveness of two intelligent proactive hint policies learned via a Deep Q-Network and Long Short-Term Memory (LSTM) Neural Network. The contributions of this dissertation explore what level of assistance produces the best learning outcomes, how students respond to proactive versus autonomous, student-chosen hints, and how to derive RL policies that determine when proactive assistance would produce beneficial learning outcomes.
BIOGRAPHY

Christa Nicole Cody was born in 1994 and grew up in Morristown, Tennessee raised by single mother, Mickie Cody. With much support and encouragement, Christa started her undergraduate career at Tennessee Technological University in 2012 and received her Bachelor of Science in Computer Science in 2015. After spending a couple years exploring Smart Meter energy consumption research, she decided to apply to graduate school and discovered the field of educational systems. She attended North Carolina State University where she received her Master's and Doctorate in Computer Science. She currently resides in Raleigh, North Carolina, but intends to move wherever the next thing in life takes her.
ACKNOWLEDGEMENTS

Dr. Tiffany Barnes, thank you for your guidance and support. You have helped me learn how to speak my opinion louder and provided me the support I needed to succeed.

Zach Cleghern, I would definitely not have succeeded without your support and security. You have read almost every line I have ever written, at least twice, and helped me believe in myself. I, also, can't leave out our very loving and supportive grad school kitty, Balerion (RIP). He stuck with me through the worst of it and was always by my side.

Mom, I would not be the person I am today without you. You helped me grow into an independent person and have always been supportive of my decisions.

And a big thanks to

• the Game2Learn lab members for constantly finding new and unique ways to provide support to each other,

• my friends who provided love, support, and a very needed distraction from work through board games, DnD, and outdoor adventures,

• and my family, who never understand what I am doing but always have my back: Dad, Mammy & Pappy, Nana, and everyone else.

This material is based on work supported by the National Science Foundation under Grant No. 1726550, Integrated Data-driven Technologies for Individualized Instruction in STEM Learning Environments, led by Min Chi and Tiffany Barnes.
# TABLE OF CONTENTS

**LIST OF TABLES** .................................................................................................................. vi

**LIST OF FIGURES** .................................................................................................................. viii

**Chapter 1  INTRODUCTION** .................................................................................................. 1
  1.1 Deep Thought, a logic proof tutor .................................................................................... 3
    1.1.1 Design ..................................................................................................................... 3
    1.1.2 Assistance ................................................................................................................ 4

**Chapter 2  Related Works** ..................................................................................................... 6
  2.1 Intelligent Tutoring Systems ............................................................................................. 6
  2.2 Effective Tutoring ............................................................................................................ 6
    2.2.1 The Need for Assistance .......................................................................................... 8
  2.3 Assistance in ITS ............................................................................................................. 9
    2.3.1 Types of Assistance ................................................................................................. 9
    2.3.2 Data-Driven Assistance ............................................................................................ 12
    2.3.3 Proactive Assistance ............................................................................................... 13
  2.4 Aptitude-Treatment Interaction ........................................................................................ 14

**Chapter 3  Study 1 - Hint Content** ....................................................................................... 15
  3.1 Abstract ........................................................................................................................... 15
  3.2 Introduction ...................................................................................................................... 16
  3.3 Related Works .................................................................................................................. 17
    3.3.1 Approaches to Assistance in Tutoring Systems ....................................................... 18
  3.4 The Deep Thought Logic Proof Tutor ............................................................................. 20
    3.4.1 Assistance ............................................................................................................... 23
  3.5 Method ............................................................................................................................. 26
    3.5.1 Hypotheses .............................................................................................................. 27
    3.5.2 Performance Evaluation Metrics ............................................................................. 27
  3.6 Results ............................................................................................................................... 29
    3.6.1 Effects on High- and Low- Pretest Groups ............................................................... 30
    3.6.2 Did Waypoints help with strategy for those who could utilize them? .................... 33
    3.6.3 What are the circumstances when Assistance was not used? ................................. 34
  3.7 Discussion ........................................................................................................................ 36
    3.7.1 Waypoint hints ....................................................................................................... 37
    3.7.2 Next-Step hints ...................................................................................................... 38
  3.8 Contribution ...................................................................................................................... 38

**Chapter 4  Study 2 - Autonomy** ............................................................................................ 40
  4.1 Abstract ............................................................................................................................ 40
  4.2 Introduction ...................................................................................................................... 41
  4.3 Background ...................................................................................................................... 42
  4.4 Deep Thought, a logic proof tutor .................................................................................. 43
  4.5 Methods ........................................................................................................................... 45
    4.5.1 Performance Metrics .............................................................................................. 46
  4.6 Results & Discussion ........................................................................................................ 48
4.6.1 Hint Usage and Help Need ............................................. 48
4.6.2 Evaluating Students' Performance Across the Tutor .................. 51
4.6.3 Did the Choice group make good help-seeking decisions? ............... 52
4.7 Contribution ............................................................... 56

Chapter 5 Study 3 - RL-Step Policies ............................................. 57
  5.1 Background ............................................................... 58
  5.2 The Deep Thought Logic Proof Tutor .................................... 60
    5.2.1 Assistance .......................................................... 63
  5.3 Method ................................................................. 65
    5.3.1 DQN Data Used for Training .................................... 65
    5.3.2 DQN Reward Generation ........................................ 66
    5.3.3 DQN Policy Induction ........................................... 67
  5.4 LSTM Data Used for Training .......................................... 68
  5.5 LSTM Reward Generation ............................................. 69
  5.6 LSTM Policy Induction .............................................. 69
  5.7 Analysis and Performance Metrics .................................... 70
  5.8 Results & Discussion ................................................ 73
  5.9 Effects on Students with High and Low Prior Proficiency ................. 79
  5.10 Investigating Policy Carry-out in the Historical Random Assertion data ... 80
  5.11 Exploring the Behavior of the Proactive Hint Policies ................. 81
  5.12 Behavior of DQN Policy ............................................. 81
  5.13 Behavior of LSTM Policy ........................................... 83
  5.14 Discussion of Commonalities and Differences in Proactive Hint Policy Behavior ... 85
  5.15 Discussion of Overall Results ..................................... 88
  5.16 Limitations ........................................................... 89
  5.17 Contribution .......................................................... 89

Chapter 6 Contributions ......................................................... 91
  6.1 Contributions .......................................................... 91

BIBLIOGRAPHY ................................................................. 92
**LIST OF TABLES**

Table 3.1  Hint metrics during training. For ANCOVA results controlling for the pretest score, p-values that are at least marginally significant are **bolded** and significant values also have an asterisk*. ................................................................. 30

Table 3.2  Performance metrics for each group on the pretest, training, and posttest; p-values that are at least marginally significant when applying ANCOVA controlled for pretest are **bold** and those that are significant also have an asterisk. 31

Table 3.3  Performance metrics between the NS and WP **High** proficiency groups for the pretest, training, and posttest of the tutor. ANOVA results are reported for the pretest. ANCOVA results, controlling for the pretest, are reported for the training and posttest, with p-values that are at least marginally significant in **bold** and significant p-values also have an asterisk *. ........................... 32

Table 3.4  Performance metrics between the NS and WP **Low** proficiency groups for the pretest, training, and posttest. ANOVA results are reported for the pretest. ANCOVA results, controlling for the pretest are reported for the training and posttest; p-values that are at least marginally significant are in **bold** and significant p-values also have an asterisk *. ........................... 33

Table 3.5  Significant correlations between hint Justification and Adoption rates with posttest performance metrics for each hint type group and pretest group . . . 34

Table 3.6  The total unused (unjustified) hints, percentage of hints unused out of all hints added, and the percentage of the unused hints that were attempted to be derived between the NS and WP group. For ANCOVA results controlling for the pretest score, p-values that are at least marginally significant are in **bold** and significant values also have an asterisk *. ........................... 35

Table 3.7  Comparison of unused hints of each subtype by amount, percentage that were attempted, and steps before the action occurred. ........................... 36

Table 4.1  Mean and Standard Deviation(SD) of the Hint Usage Metrics over the Training portion of the tutor for the Control, Choice, and Assertions groups. 49

Table 4.2  Mean and Standard Deviation(SD) of the Help Need Metrics over the Training portion of the tutor for the Control, Choice, and Assertions groups. 50

Table 4.3  Pretest, Training and Posttest performance metrics for the Assertion, Choice, and Control groups. ANOVA for the pretest and ANCOVA using pretest metrics as the covariate was used to determine significance between group means. P-values are reported in text and **bold** in table. 52

Table 4.4  The Posttest Performance Metrics for the Yes and No groups within the Choice condition. 55

Table 4.5  The Posttest Performance Score for the High- Low- Proficiency groups within the Yes and No groups within the Choice condition. 55

Table 5.1  Mean and Standard Deviation(SD) for Assertion Metrics in Training between the DQN and LSTM groups. Significance tests’ p-values that are at least marginally significant are **bolded** and significant values also have an asterisk(*). 74
LIST OF FIGURES

Figure 1.1  On the left of the screen is the Deep Thought workspace. Below the workspace are are the hint button and hint message box, the rules are in the middle, and to the right is the Dialogue Box where messages related to unsolicited hints as well as problem information are given. ............................................. 4

Figure 1.2  A hint statement being justified. (1) Shows a hint appearing in the workspace. (2) Two statements are selected (highlighted in blue) and the rule “Modus Ponens” applied. (3) The hint has been justified and is connected to the student proof. ................................................................. 5

Figure 2.1  Example of minimal feedback using red coloring in the Andes physics tutor. 10
Figure 2.2  A worked example in Deep Thought. ............................................................... 11
Figure 2.3  A two-layer approach to sequencing hints [Ano07] ........................................... 12

Figure 3.1  On the left of the screen is the Deep Thought workspace. Below the workspace are are the hint button and hint message box, the rules are in the middle, and to the right is the Dialogue Box where messages related to unsolicited hints as well as problem information are given. ............................................. 21

Figure 3.2  Deriving a new justified node. (1) Selecting the node “I ∧ F” to use (2) Selecting the rule “Simplification” to apply (3) The screen after the rule was clicked showing “F” as a justified node ......................................................... 22

Figure 3.3  The left side shows the justification of a Next-Step hint and the right side shows the justification of a Waypoint hint. ................................................................. 24

Figure 3.4  A completed problem with nodes that were used to derive the conclusion (justified and adopted) and one node that was not used to derive the conclusion (justified but not adopted). ................................................................. 29

Figure 4.1  The logic tutor, showing the workspace on the left with Givens (1,2,3,4) at the top and the Conclusion (Q → R) at the bottom. Rules are in the middle, and the Info box is to the right, where problem information and notices to attempt to justify Assertions appear. ......................................................... 44

Figure 4.2  A hint statement being justified. (1) Shows a hint appearing in the workspace. (2) Two statements are selected (highlighted in blue) and the rule “Modus Ponens” applied. (3) The hint has been justified and is connected to the student proof. ................................................................. 45

Figure 4.3  Random walks of the Low and High Choice groups are labelled. In each plot, lines are colored red for Low-posttest and blue for High-posttest. Low/High posttest splits were determined via median split of the pretest score among all students in the study. ................................................................. 53

Figure 5.1  On the left of the screen is the Deep Thought workspace. Below the workspace are are the hint button and hint message box, the rules are in the middle, and to the right is the Dialogue Box where messages related to Assertions as well as problem information are given. ......................................................... 61

Figure 5.2  Deriving a new justified node. (1) Selecting the node “I ∧ F” to use (2) Selecting the rule “Simplification” to apply (3) The screen after the rule was clicked showing “F” as a justified node ................................................................. 62
Figure 5.3  A hint statement being justified. (1) Shows a hint appearing in the workspace. (2) Two statements are selected (highlighted in blue) and the rule “Modus Ponens” applied. (3) The hint has been justified and is connected to the student proof. ................................................................. 64

Figure 5.4  Neural Network architecture diagram for the DQN model. ....................... 68

Figure 5.5  Neural Network architecture diagram for the LSTM model. ..................... 70

Figure 5.6  A completed problem with nodes that were used to derive the conclusion (justified and adopted) and one node that was not used to derive the conclusion (justified but not adopted). ........................................... 72

Figure 5.7  Regression plot of the Relative Score between the Control, DQN, and LSTM groups with the Posttest Relative Score plotted against the Pretest Relative Score. ............................................................... 77

Figure 5.8  Optimal number of clusters determined by test 1-5 clusters and comparing the average silhouette width. The number of clusters that was optimal for the students in the DQN group was 4. .................................................. 81

Figure 5.9  Bivariate plot visualizing the clusters based on two principal components for the DQN group. ................................................................. 82

Figure 5.10 Each student in the DQN group is plotted based on their Pretest Total Time, Pretest Accuracy, Total Assertions Received, and total Hint Requests. Lines are colored according to their cluster assignment. ................................. 83

Figure 5.11 Optimal number of clusters determined by test 1-5 clusters and comparing the average silhouette width. The number of clusters that was optimal for the students in the LSTM group was 3. .................................................. 84

Figure 5.12 Bivariate plot visualizing the clusters based on two principal components for the LSTM group. ................................................................. 84

Figure 5.13 Each student in the LSTM group is plotted based on their Pretest Total Time, Pretest Accuracy, Total Assertions Received, and total Hint Requests. Lines are colored according to their cluster assignment. .................. 85

Figure 5.14 The mean Total Hints received, Total Assertions given, and Total Msg Hints for the Control, DQN, and LSTM groups. .................................................. 87
Intelligent tutoring systems provide a unique advantage over human tutors by allowing the system to teach at any time or place, provide a one-on-one experience while being able to teach numerous students at a time, and adapt to the student's learning needs for a customized experience. Research has shown providing supported problem solving produces better learning outcomes to unsupported problem solving [Swe88]. However, providing the student assistance without providing unnecessary or inadequate guidance is argued to be crucial to producing good learning outcomes [Bur79; Sch89; Shu88; Sch87], a problem often referred to as the assistance dilemma [Koe07].

Multiple types of assistance have been implemented and combined to provide adequate support while a novice is learning a new domain, including different forms of feedback [Van06; McK90], worked examples [Swe06], and hints [Van06; Hum96; Eag14a]. Research in ITSs has shown that students' assistance needs vary depending on multiple factors, such as prior ability and current progress [Arr00; Ste96; Van06]. Therefore, some ITSs have explored the effects of providing multiple levels of hints to allow for a more adaptive range of assistance [Hum96; Woo78; Ste96; MS11; Sta08]. However, little research has explored the impact of the level of assistance and how types of assistance affect both learning outcomes and help-seeking behavior.

Despite assistance proving to produce better learning outcomes, there is a pervasive problem within ITSs called help avoidance, where students do not use help available within the system [Ale04; Ale00]. One potential reason for why this may occur is that certain students may lack specific meta-cognitive skills to know when asking for help would be beneficial for them [Ale00]. One remedy to this issue some ITSs employ is proactive assistance [Van06].

Despite being unsolicited and potentially frustrating or distracting for a student, proactive assistance in ITSs has shown some promising results when compared to unsupported problem
solving [Ran11]. On the other hand, similar research exploring proactive assistance found the opposite to be true, showing that on-demand hints produced better learning outcomes than the condition with proactive interventions [Raz10]. These studies suggest that there is more research to be done to determine policies for when, how, and what hints are presented to students and how these impact learning outcomes. Researchers have explored the creation of effective proactive hint policies through probabilistic models with promising results [Mur04; Uen17]. However, little research has explored the use of RL for creating effective proactive hint policies, and the research completed was in a problem space constrained by sequentially solving the problem given a specific strategy [Zho19], not in a tutor with an open-ended problem space such as Deep Thought.

Given the relative importance of providing effective, customized assistance, the issue of help avoidance with ITSs, and the lack of research exploring proactive hint policies, my work seeks to explore what level of assistance produces the best learning outcomes, how students respond to proactive hints, and when proactive assistance would produce beneficial learning outcomes.

My studies seek to answer the following research questions:

• RQ1) How do higher-level data-driven hint types affect learning outcomes compared to next-step hints derived from a similar data-driven method and are there subgroups of students which benefit more from one type over another?

• RQ2) How do on-demand only systems compare with a proactive hint system in terms of student hint usage, learning, and performance?

• RQ3) How can Reinforcement Learning be used to create an effective proactive hint policy for producing better learning outcomes compared to unsupported problem solving?

The following are my dissertation’s contributions:

• Insight into how existing data-driven methods for next-step hints can be used to generate higher-levels of hints (RQ1)
  
  Design suggestions for scaffolding higher-level hints for future studies

• Greater understanding of how different hint types affect learning outcomes, especially between high and low proficiency students (RQ1)

• Insight into how different levels of autonomy regarding assistance affects learning outcomes (RQ2)

• Insight into reward generation and policy induction methods for creating a proactive hint policy (RQ3)

In the next section, Section 1.1, I discuss the design and assistance of Deep Thought. In Chapter 2: Related Works I provide background in effective tutoring, the need for assistance, types of assistance commonly found in ITSs, data-driven assistance, and the need for proactive assistance. In Chapter
3: Study 1 - I describe a Fall 2018 study comparing two data driven hint types and the evaluation of their effects on learning outcomes to answer RQ1 and RQ2. In Chapter 4: Study 2 - Assistance Autonomy, I describe a Spring 2019 study comparing test conditions with varying levels of student autonomy regarding tutor-initiated, proactive assistance to answer RQ3. In Chapter 5: Study 3, I describe a study testing the effectiveness of a Deep Q-network proactive hint policy deployed in the Fall of 2019 and a Long Short-Term Memory (LSTM) Neural Network hint policy deployed in the Spring of 2020.

1.1 Deep Thought, a logic proof tutor

In this section, I will discuss the design of Deep Thought and its assistance. Then, I will describe the data collected and the students who use the tutor. Last, I will discuss how the Deep Thought tutoring system is comparable to other systems and how studies conducted in Deep Thought are generalizable to the educational technology and learning community.

1.1.1 Design

Deep Thought, a propositional logic tutor [Mos17], is used in the Discrete Mathematics course for computer scientists at North Carolina State University (CSC 226), which is used by approximately 250-300 students per semester. Deep Thought is released as a mandatory homework assignment two weeks before their first exam, which contains logic proof questions. Each semester the tutor is updated based on the current studies’ needs. Click-based action logs are recorded for every action a student takes in the tutor, with problem-level and step-level models being generated and stored as students work through the tutor. The system tracks every click-based action, such as how many hints are requested, how many times a student views the rule descriptions, how many times the student tries to incorrectly use a rule, etc. In Deep Thought, students solve proofs in a graphical interface (see Figure 1.1). The tutor presents proof problems as a set of logical premises with a conclusion derivable using logical axioms. Each statement, given or derived, is represented by a node, which students can click to select and apply a logical axiom/rule to derive a new node. Students engage with the system through three phases; pretest, training, and posttest. The pretest contains a short tutorial (i.e. fixed set of worked examples) followed by two unassisted problems, the last being used as the pretest assessment. Students solve 15 problems in the training section. Students receive hints during training, including both proactive hints generated by the system and on-demand hints upon student request, all generated using the same Hint Factory-type approach described below. After completing training, students take a more difficult non-isomorphic posttest, where they must solve four problems without any help or assistance. Note, in Fall 2019 two problems were included from the pretest into the posttest; therefore, I will be able to make an isomorphic comparison in Fall 2019 studies and forward.
1.1.2 Assistance

The tutor uses a data-driven approach to generate assistance from historical student data. This algorithm results in assistance based on the most frequent and efficient paths available based on the student’s current proof.

Assistance provided by the tutor in the form of hints can either be initiated by the student, in which case they are called on-demand hints, or they can be initiated by the tutor, in which case they are called proactive hints, or Assertions. In training problems, the student’s may request on-demand, next-step, in-workspace hints, see Figure 1.2 for an example of the in-workspace hint. Note, this is different than the procedure described in Chapter 3 Study 1, which provided text-based, on demand hints when requested. I made this change due to work that investigated found the in-workspace hints to result in the hints being derived more often. Furthermore, students in conditions that receive proactive hints (Assertions) receive the same format and content. Pretest and posttest problems disallow any Assertions or hints. In training problems, Assertions are nodes that the tutor adds to the workspace, containing a goal statement for the student to derive. Both on-demand hints and Assertions appear in the workspace labeled as “Subgoal” and are incorporated directly into the student’s solution when the student selects the rule and source statement(s) that can be used to derive the proposition. Within logic, using the right rules to derive a target proposition from the given source statements means that the new statement is justified. Assertions do not tell students which rules to use to derive the new goal proposition. Rather, Assertions are designed to help students solve problems by suggesting a goal proposition that helps them break down these multi-step problems.
All hints are next-step, which suggest the next, best proposition that can be derived in one step from the student's current proof. For example, one tutor assertion could add an $E$ node, and the student would select $\neg D \land E$ and choose the “Simplification” rule to justify/derive the assertion node.

![Figure 1.2](image)

**Figure 1.2** A hint statement being justified. (1) Shows a hint appearing in the workspace. (2) Two statements are selected (highlighted in blue) and the rule “Modus Ponens” applied. (3) The hint has been justified and is connected to the student proof.

Due to Deep Thought’s evolution over the years, each chapter contains more detailed information about the tutor design for that particular study.
CHAPTER

2

RELATED WORKS

2.1 Intelligent Tutoring Systems

2.2 Effective Tutoring

Human tutors have the unique ability to relate to the student and have a variety of techniques available for assisting the student in problem solving, such as asking questions to determine what the student does not understand or adapting their level of guidance as the student responds to each hint. Compared to traditional instruction, one-on-one tutoring is well-accepted to promote greater student learning and increase student motivation to learn [Sla87]. The well-known Bloom's 2 sigma problem concludes that one-to-one tutoring performs, on average, two standard deviations above conventional instruction methods and one standard deviation above mastery learning. This research also suggested that tutoring reduced the effect of the students' pre-test scores being suggestive of their post-test scores. This implies that tutoring allows students at any aptitude level to achieve similar performance as their initially higher performing peers [Blo84]. There has been significant research outlining the good and bad traits of human tutors and in what ways each trait affects the learning process. One model for tutoring success has been created that outlines traits found in expert tutors, INSPIRE [Lep97]. The traits of the INSPIRE model are described as follows:

1. Intelligent
   (a) Expert knowledge of domain
   (b) Wide variety of explanations
2. Nurturant
   (a) Highly supportive
   (b) Empathic

3. Socratic
   (a) Questions, not directions
   (b) Hints, not answers

4. Progressive
   (a) Problem progression: **appropriate increase in difficulty**
   (b) Systematic debugging of student errors

5. Indirect
   (a) Subtle negative and positive feedback

6. Reflective
   (a) Make students articulate, explain, and generalize concepts

7. Encouraging
   (a) Build and maintain confidence
   (b) **Challenge**
   (c) **Allow control**

This model outlines traits observed in effective tutors, but can also be seen in the design of tutoring systems. For example, tutoring systems incorporate a wide variety of explanations through different forms of feedback and support through assistance, discussed further in Section 2.3.1. These different forms can include support such as worked examples, hints, and immediate feedback. Furthermore, a large body of research has studied how to implement different levels of feedback and assistance to provide in tutoring systems [Van06; Cat98] and, generally, aims at providing direction rather than answers [McK90]. Another important component of making tutoring systems intelligent is providing the appropriate level of content, which has been done through problem selection [Mos16a; Koe10] but also through supported problem-solving through assistance [Hum96; Pri17c; Ste96].

Multiple studies have shown that human tutors carefully follow student's problem solving and intervene to various degrees from very subtly to more directive feedback [Fox91; Mer92], especially because research has shown the context of the error is critical in determining the appropriate
feedback to give the student [Lit90] Tutoring systems often consist of complex student modelling and logging, so following the student’s problem solving approach can allow for similar contextualized assistance. This knowledge can help contextualize the feedback to provide the student with guidance that aligns with the student’s learning ability, progress, and affect. In the next section, I discuss why providing the appropriate level of challenge to a student is crucial and how assistance has been shown to effectively scaffold problems to multiple levels of learner needs.

2.2.1 The Need for Assistance

Determining how much assistance to provide is a complex task requiring a delicate balance of the student being in control of the learning process and not becoming too frustrated, confused, or off-task. This dilemma is referred to as the assistance dilemma [Koe07]. Allowing the student to have control over the learning domain without providing unnecessary or inadequate guidance is argued to be crucial to learning [Bur79; Sch89; Shu88; Sch87]. Negativity bias suggests that solving the problem and overcoming obstacles, while a possibly negative experience, are more memorable to students than being directly given the answer [Ito98]. This is because students typically exhibit negative emotions when unable to accomplish a task, such as frustration or confusion. Allowing the student to struggle can produce positive benefits, such as remembering the approach to take on certain problems or what paths are unproductive. However, this effect has its limits and leaving a student to unrestrictedly explore a learning domain without any guidance may lead to less effective learning or a student not remembering the path to the solve the problem [Swe88]. This is particularly important for domains with complex problem solutions because a student may take multiple initial paths to solving a problem before realizing the correct path. If the student is continually backtracking or getting frustrated, the full path to a solution may not be realized and the student might not be able to apply the way the solution was achieved to similar problems. Guiding the students through the learning process without providing direct answers has been shown to have positive learning outcomes [Kir06; May04]. Another set of research advocating the need for adequate guidance is the theory of the “zone of proximal development” (ZPD), which represents the area between steps or problems a student can do independently and those they can only do with support [Vyg78]. Vygotsky theorized that providing challenges to students that are within this area produces the best learning outcomes. The reasoning is that, when students are in this zone, they will struggle just enough to remember the concept in the future, but not to a point that they become so frustrated, which interferes with developing their mental model. However, the cognitive load of a learner is affected by both the difficulty of the task and their own proficiency with the topic [Swe88]. Therefore, not only is assistance crucial to providing the best learning experience, but the amount, content, and timing are dependent on each learner’s needs.
2.3 Assistance in ITS

This section describes assistance in intelligent tutoring systems. First, I describe different forms of assistance within intelligent tutoring systems with the purpose of outlining the advantages and disadvantages of each type and relation to forms used in Deep Thought. The forms of assistance I describe are feedback, worked examples, and hints. Then, I discuss the variety of ways that data-driven methods are used to derive new forms of assistance. Lastly, I discuss research involving proactive assistance, focusing towards the end on proactive hinting policies.

2.3.1 Types of Assistance

This section describes 3 forms of assistance, each with a range of implementations: feedback, worked examples, and hints.

2.3.1.1 Feedback

Feedback was one of the first forms of assistance implemented in the tutoring systems, or at the time references as computer assisted instruction systems [Shu94; Fry60]. There are multiple approaches to feedback in tutoring systems which face three main questions: how to give feedback, when to give feedback, and what to give feedback on. For the first question – how to give feedback, there are many techniques depending on the goal the feedback is pushing towards. In the work by Kurt Vanlehn, "The Behavior of Tutoring systems, " multiple forms of feedback are discussed at length, including minimal feedback, error-specific feedback, coarse-grained assessment, and reviewing student's solutions [Van06].

Each category of feedback is distinguished by the type of feedback given and when it is given, which could be at multiple points during the problem solving process, such as each step, the solution approach, or the overall performance of the student (time to correctly solve, number of mistakes, number of hints requested). Minimal feedback is often utilized by informing a student whether a step they just completed is correct. An example of this type of feedback was discussed earlier when I described how students can tell a step is completed, in Deep Thought, via arrows connecting their step to the rest of the solution. In other systems, this feedback can be as simple as a red/green flag indicating correctness. Similarly, error-specific feedback provides feedback regarding an error the student has made, which could include violating a rule or miscalculations. Andes, a physics tutor, focuses on being minimally invasive and uses an expert model that incorporates physics domain knowledge and pedagogical strategies as the foundation [Van05]. Andes provided on-demand hints and immediate feedback to the student. Feedback in Andes includes coloring entries in the user interface red or green for incorrect or correct entries, respectively. Students can request hints to help correct red entries or further solve the problem [Van04]. In Figure 2.1, an example of Andes feedback is shown. In Deep Thought, error-specific feedback is provided to the student any time they attempt to use a rule incorrectly. Research in error-specific feedback found that goal-directed feedback about what to do after making an error was more effective than reasoning about the error.
alone [McK90], which gives more reason to pursue hints as an additional action for support. Coarse-grained assessment provides a measure of progress to the student and can include the percentage of the tutor they have completed or the overall accuracy or mastery they have achieved. Another form of feedback is when tutor’s review student’s solutions after they have been submitted, a form of on-demand feedback. The review may include information about which steps were not necessary or incorrect and can also be considered a form of delayed feedback. Other research has used feedback to improve student’s meta-cognitive skills regarding help-seeking by providing them feedback on bad help-seeking behaviors [Rol11].

![Figure 1.12: Refer to Example](image)

Figure 2.1 Example of minimal feedback using red coloring in the Andes physics tutor.

2.3.1.2 Worked Examples

A worked example (WE) teaches students how to solve a problem through direct guidance of each step of the solution. For novices with little experience in a domain, a worked example would help the novices see how a problem is solved and arrive to the solution without much frustration. Worked examples have been widely studied and shown to simplify complex problems and improve learning outcomes [Swe06]. Studies have shown that WEs produce better learning outcomes, helping students solve problems quicker with fewer errors, compared to unsupported problem solving [Swe85]. In Deep Thought, Liu et. al investigated whether WEs are beneficial to students. and found that they were beneficial when utilized earlier in the tutor, but did not find those same benefits when implemented in later problems [Liu16]. Research has also investigated whether intelligently pro-

10
viding worked examples over problem solving improves learning, and found that effective policies with better learning outcomes can be produced [She18]. Figure 2.2 shows an example of a worked example in Deep Thought, where the student can click through the arrows to see more steps worked out by the tutor.

![Figure 2.2 A worked example in Deep Thought.](image)

### 2.3.1.3 Hints

Hints are a common approach to supporting students during problem solving. Although feedback is often considered a form of hint, this section focuses on applications in which the hint is providing guidance on future steps, such as next-step hints and subgoal hints. Different levels of content and presentations for hints have been explored in various research [Hum96; Woo78; Ste96; Mor15; MS11]. Next-step hints are a common way to provide hints as seen in tutors such as cognitive tutors [And95], which cover domains such as algebra and computer programming, the Snap/textit! block-based programming tutor [Pri17c], and the Pyrenees probability tutor [Chi10]. Another type of hint is a subgoal, which is a type of hint that seeks to mimic how experts breakdown problems into easier to achieve sub-components by providing a subgoal within a problem for a student to work towards [Cat98]. Subgoals have been created from expert labels [Kim13] and from data-driven generation [Eag14a].

A common approach to providing hints is to have a sequence of hints ranging from more general hint then transition to more specific and directive hints. A standard goal-directed hint sequence within tutoring systems is Point, Teach and Bottom-out [Hum96]. Pointing hints attempt to remind
the student of relevant material. Teaching hints describe how to apply the relevant material. Bottom-out hints tell the student the next step and specifically how to implement it. However, students may benefit from receiving a certain hint type over another, so providing hints in a strict sequence might limit the effectiveness of receiving hints. Contingent tutoring is another policy that attempts to remedy this issue, by mimicking how a human tutor will tailor the type of hint they give a student based on the student's previous behavior [Woo78]. An example of this type of tutoring can be found in the tutoring system Quadratic where the tutor will give more specific or general hints depending on the student’s success with the previous hint type given [Woo99]. Some research individualizes hint policies based on student behavior and ability, allowing the student to receive the appropriate hint without stepping through several levels of hints [Ste96]. For example, in a tutoring system for teaching the Min-max algorithm, Alla Anohina created a two-layer model of hints to be implemented based on the idea that intelligent tutors do not provide enough adaptive abilities [Ano07]. The two layers consist of one layer with general hint types and a layer of hints within the general types, shown in Figure 2.3. Research in ITSs has shown that assistance needs vary depending on multiple factors, such as prior ability and current progress [Arr00; Ste96; Van06].

![Figure 2.3 A two-layer approach to sequencing hints](image)

Although these tutors tailor the specific content of the hint to the level of the learner, the timing of the hint is not proactively determined. Proactive assistance is discussed in Section 2.3.3. In the next section, I will discuss data-driven methods for generating assistance.

### 2.3.2 Data-Driven Assistance

Data-driven assistance provides the ability to generate new methods of assistance from historical student data without the need for an expert, saving time and resources. In complex problem solving spaces, there are often many ways a correct solution may be achieved and not always a clear optimal path. Defining the best solution or encompassing all the solutions can be difficult to achieve, especially when this is to be done by a human expert. Thus, historical data can be used to assess states, construct a student model, and provide contextualized feedback with just the desired solution of a problem. The Hint Factory is a data-driven approach developed to generate Next-Step hints.
for students applying rules to solve open-ended logic problems [Sta08; Sta13]. New innovations in generating assistance from individual pieces of previous student’s solutions have helped researchers extend the ideas of the Hint Factory to generate Next-Step hints for new domains including novice programming and linked list construction [Fos10; Riv17; Pri16]. Researchers have also successfully generated effective, data-driven hints using other methods with historical problem solving data [Pie15; Gla16]. Approaches to assistance that can use data-driven hints effectively demonstrate that effective tutoring can be achieved without expert-created hints.

2.3.3 Proactive Assistance

Despite this considerable amount of research on assistance, there is a pervasive problem within ITSs called help avoidance, where students do not leverage the intelligence within the system for help [Ale04]. There are many reasons for help avoidance, one of which is certain students may lack specific meta-cognitive skills like knowing when to ask for help [Ale00]. In a study using the Andes physics tutor, researchers found that a large number of students would guess rather than ask for hints [RAN14]. Help refusal is a large problem in tutors that provide on-demand hints because students can be too stubborn or unaware of the hints to effectively use them [Ale00]. Another study with Andes also tested unsolicited hints or meta-hints (asking the student to ask for a hint) [Ran11]. The results of the experiment suggested that unsolicited hints were more effective than on-demand hints. Thus, although students need to have control in the learning process, the tutoring system also needs to provide the guidance necessary to prevent unbeneﬁcial struggle, sometimes presenting hints to the student regardless of them requesting it. This is similar to how human tutors interact with students because they tend to probe or question the student when they notice the student struggling with a problem.

To overcome this challenge, some ITSs employ proactive assistance to prevent help avoidance [Van06]. In a paper by Brawner et al, the difﬁculties and advantages of producing proactive agents are discussed [Bra11]. The review covers how human tutors are inherently proactive when helping a student solve a problem, but that proactive-ness comes at a cost in ITS due to the additional resources required to model and implement proactive policies. However, the authors conclude that to reach the same level of success that has been shown with human tutoring, advancing knowledge on proactive assistance is crucial.

One group created a tutoring policy, DT Tutor that applies decision theory to make its choice about whether to give a hint [Mur04]. The tutor uses a probabilistic model of the student to predict the student’s possible reactions and uses this probability for each action the tutor can make (e.g. giving a hint, what type of hint, etc.). The prediction includes the likelihood of learning, becoming frustrated, entering the next step correctly, and other states. Then, the tutor evaluates the utility of each of the predicted student states takes the action with the highest utility. Although advances in probabilistic reasoning make these calculations feasible, a large amount of student data is needed. Data-intensiveness is a problem in creating an ITS because you don’t initially begin with a data set. However, after the data is collected, it can provide a multitude of insights into student behavior.
and remove the burden of creating comprehensive knowledge and student models from human developers.

A study by Razzaq and Heffernan in the ASSISTment tutoring system found that on-demand hints produced better results than having proactive hints for students who request a high number of hints (HNH), but no difference for students who request a low number of hints (LNH) [Raz10]. However, the authors suggested these results may be due to the HNH group having better help-seeking than the LNH group. The authors suggested that, therefore, the LNH group may have benefitted from a proactive hint at the right time.

Ueno and Miyazawa studied the effects of providing proactive hints via a probabilistic model, item response theory (IRT) [Uen17]. The model determines whether a hint should be provided based on the predicted probability of the student’s successful performance. The study found that a predicted probability of success of 0.5 resulted in the optimal amount of assistance being provided.

These studies demonstrate both the difficulties in creating an effective proactive policy as well as the potential for providing better learning outcomes. However, proactive hints are relatively unexplored, and with results showing conflicting results, there still needs to be more investigation into what circumstances surround effective proactive hint policies. Due to the policy being affected by hint content, incoming student proficiencies, and timing, more research is necessary to determine what defines successful proactive assistance.

2.4 Aptitude-Treatment Interaction

Aptitude-treatment interactions have been studied widely in the educational domain. Prior research in instructional strategies [Cro77; Sno91] has shown the existence of aptitude-treatment interaction (ATI), where certain students are more sensitive to variations in the learning environment and may be affected differently by the treatment compared to less sensitive students who perform regardless of the treatment. Educational researchers have discovered ATI effects based on prior experience level, prior working memory, and incoming self-regulated learning ability [Kal01; Leh16; Fuc19; Yeh15]. For example, Lehmann et al explored the effect of working memory on learning outcomes in fluency/disfluency groups, where instructional materials had different levels of text legibility [Leh16]. They found a significant aptitude treatment effect among the disfluency group where they believe only learners with a high working memory capacity are capable of using those texts. Furthermore, Yu-chu yeh et. al [Yeh15] showed aptitude-treatment interactions in a system designed to teach creativity. Students with a higher knowledge management ability on the pre-test showed greater improvements in the post-test after training than those with lower levels of knowledge management. In my work, I have found evidence of aptitude treatment interactions with students of different incoming proficiencies. Students with higher incoming proficiencies are less affected between conditions than the lower proficiency students.
This presents my study: “The Impact of Looking Further Ahead: A Comparison of Two Data-driven Unsolicited Hint Types on Performance in an Intelligent Data-driven Logic Tutor.” This study focuses on evaluating the effects on learning outcomes and behaviors from a new data-driven hint type, Waypoints, in comparison to the tutor’s original data-driven Next-Step hints.

3.1 Abstract

Research has shown assistance can provide many benefits to novices lacking the mental models needed for problem solving in a new domain. However, varying approaches to assistance, such as subgoals and next-step hints, have been implemented with mixed results. Next-Step hints are common in data-driven tutors due to their straightforward generation from historical student data, as well as research showing positive impacts on student learning. However, there is a lack of research exploring the possibility of extending data-driven methods to provide higher-level assistance. Therefore, we modified our data-driven Next-Step hint generator to provide Waypoints, hints that are a few steps ahead, representing problem-solving subgoals. I hypothesized that Waypoints would benefit students with high prior knowledge, and that Next-Step hints would most benefit students with lower prior knowledge. In this study, I investigated the influence of data-driven hint type, Waypoints versus Next-Step hints, on student learning in a logic proof tutoring system, Deep Thought, in a discrete mathematics course. I found that Next-Step hints were more beneficial for the majority of students in terms of time, efficiency, and accuracy on the posttest. However, higher totals of successfully used Waypoints were correlated with improvements in efficiency and time in the posttest. These results suggest that Waypoint hints could be beneficial, but more scaffolding
may be needed to help students follow them.

3.2 Introduction

Intelligent tutoring systems (ITS) provide adaptive assistance to students and have significant positive effects on learning [Mur99; Ma14]. Multiple approaches to assistance have been explored, with some very specific assistance, like bottom-out hints [Van06], designed to ensure that students “do not flounder during problem solving” [Mer92], while other more abstract assistance, like a suggested subgoal [Cat98], is designed to allow more freedom and exploration within the domain. Providing assistance has been shown to reduce the cognitive load of learning by simplifying the task, leading to greater learning outcomes in less time [Kal11; Swe88]. However, determining what level or type of help students need is a complex task that can affect learning outcomes [Ale00; Van06; Woo99]. A major goal of providing assistance is to level the playing field of learning so that students at any incoming proficiency can master the same material in a similar amount of time. Research has shown that the level of hint and the learner’s incoming experience can affect learning outcomes in ITSs [Arr00; Kal11]. One example of this is the expertise reversal effect where methods that benefit novices, such as worked examples, become detrimental to students with higher expertise due to increasing cognitive load through redundant information [Swe08]. Prior research in instructional strategies [Cro77; Sno91] has shown the existence of aptitude-treatment interaction (ATI), where certain students are more sensitive to variations in the learning environment and may be affected differently by the treatment compared to less sensitive students who are able to perform well regardless of treatment. ATI effects have been discovered based on prior experience level, prior working memory, and incoming self-regulated learning ability [Kal01; Leh16; Fuc19; Yeh15]. Therefore, I hypothesized that different hint types could have different effects based on students’ incoming proficiency.

Similar to solving programming problems, solving logic proofs requires students to understand a system of domain principles or rules and to creatively apply them in sequence to achieve a goal. Support can be directed at any of these facets of problem solving, such as helping a student learn a rule or identify when applying a such a rule will move them towards a goal.

Data-driven methods, where actions within the tutor are designed and developed using historical data, have been used to great effect to automate and individualize computer-aided instruction [Mos16b; Sta13; Fos15; Bar08]. The tutor’s data-driven assistance matches current student work with similar prior successful and efficient examples to provide adaptive Next-Step hints. Providing the next step to derive allows students to focus their learning on discovering how to reach their new short-term subgoal, rather than textitwhat next subgoal to pursue. On the other hand, Next-Step hints may reduce student autonomy or practice in creating appropriate problem solving strategies.

Another important aspect to assistance in tutoring systems is the ease of generation. The original Hint Factory opened a new field of data-driven hint generation that was first applied in tutors for logic [Sta08; Bar08], and then for linked lists [Fos10]. More recently, the Hint Factory approach
inspired new research in generating Next-Step hints for novice programming, based on generating assistance using pieces of previous student’s solutions [Fos10; Riv17; Pri17a; Bar08]. However, there is a lack of research extending this Next-Step hint generation to provide additional forms of assistance. Therefore, I used a modified version of our Next-Step hint generator to produce Waypoint hints. This extension of Next-Hint generation to provide a higher level of hint may be used in other systems to easily generate a new hint type that could provide more adaptive assistance to address individual student needs.

I created Waypoints, that can be thought of as intermediate subgoals, by modifying our Next-Step hint generator to produce hints that mimic subgoals without the need for expert labelling. The modifications were inspired by the Approach Maps technique of graph-based mining to discover important subgoals in common student solutions [Eag14b]. The new method produces Waypoint hints that require students to perform 2-3 steps to derive them. Waypoints are intended to serve as near-term subgoals, that allow students more room for exploration and latitude in strategy construction.

My goals for this study were to 1) evaluate the effectiveness of a new hint type, Waypoint hints, 2) perform a study to compare the impacts of Waypoints with Next-Step hints on performance, and 3) determine whether prior proficiency interacted with hint type to impact tutor posttest performance. I investigated the impact of the two types of hints, Next-Step and Waypoints, on student learning via unsolicited, tutor-initiated steps inserted into the student workspace, which I refer to as “Assertions”. Assertions are designed to direct student attention to, and promote adoption of, unsolicited Next-Step and Waypoint hints.

Based on the prior research mentioned above, I hypothesized that our Next-Step hints would be most beneficial for students with lower incoming proficiency and lead to better performance on the posttest. I also hypothesized that Waypoints would be more beneficial to students with higher incoming proficiency and lead to better performance on the posttest. In other words, I predicted an aptitude-treatment interaction (ATI) effect [Cro77; Sno91] where prior student proficiency would impact which students benefit most from a treatment. I predicted an ATI effect for both Waypoint and Next-Step hints, with higher proficiency students benefitting more from Waypoints and lower proficiency students benefitting more from Next-Step hints.

In this paper, I first discuss the context of the logic tutor, Deep Thought, and the method of generation for the different hint types. I then outline my experimental setup, designed to compare these two hint types in terms of their effects on student learning outcomes. Finally, I discuss the study results and how they relate to prior literature, and provide recommendations for future data-driven hint development and research.

### 3.3 Related Works

In this section, I discuss various approaches to assistance, such as subgoals, Next-Step hints, and worked examples, within intelligent tutoring systems (referred to here as ITSs or tutors). I also
discuss cognitive theories surrounding assistance, including cognitive load and the “zone of proximal
development”, that have influenced my work.

Guided discovery, helping students discover new knowledge rather than providing direct in-
struction, is generally more beneficial than allowing students to learn unguided [Kir06; May04]. This
finding agrees with the theory of the “zone of proximal development” (ZPD), the space between
things a student can do independently and those they can only do with support [Vyg78]. Vygotsky
hypothesized the most effective learning occurs when students are assigned tasks within their ZPD,
meaning that tasks should neither be so simple that students can do them independently nor so diffi-
cult that they cannot make progress even with assistance. This dilemma of choosing an appropriate
level of assistance shows that giving or withholding information is a delicate balance with trade-offs
[Koe07]. The theory of cognitive load may explain the trade-offs of different approaches to assistance.
Providing assistance can reduce the cognitive load needed for students to learn through methods
such as simplifying the task [Kal11] or breaking the task down into easier-to-digest components,
such as subgoals [Mor15]. However, the cognitive load of a learner is affected by both the elements
of information in the task and their own ability [Swe88]. Intuitively, providing assistance that is too
hard for a particular student to understand can negatively impact learning. However, providing
assistance when it is not needed may also have a negative effect, such as the expertise reversal effect
in which providing students information they already know increases their cognitive load [Swe08].
On the other hand, it is a known problem that many students fail to request help when it is needed,
and this has been termed hint avoidance [Ale00], discussed later in this section.

3.3.1 Approaches to Assistance in Tutoring Systems

Intelligent tutoring systems (ITSs) have significant positive effects on learning outcomes [Mur99].
Many forms of contextualized assistance have been explored in ITSs, such as hints, worked examples,
and error feedback [Hum96; Fos15; Uen17; Van06]. The most minimal hint type is error-specific
feedback, which provides a hint regarding an error the student has made [Van06]. Our tutor, as
described below, includes basic error feedback when rules are not applied correctly. Many tutors
use goal-directed hint sequences to provide several hints in a row, beginning with a more general
hint then transition to more specific and directive hints [Hum96]. Our tutor has this capability, but
it was disabled for this study to determine the impact of hint type and not the amount of detail each
student might request.

3.3.1.1 Higher-level Assistance

One type of assistance higher-expertise learners benefit from is subgoals, a set of steps in the solution
process that allows users to “chunk” information for ease of learning [Cat98; Mor15]. Sweller et al.
[Swe82] found that using more abstract representations of goals in five maze-tracing experiments
resulted in “fewer errors and more rapid learning of the structure of the problem.” The authors
found that the more information solvers knew about the goal, the less they learned about problem
structure. However, studies have found that these approaches have trade-offs depending on learner
ability and problem difficulty or context [Mor15]. In regard to learner’s abilities, research within ITSs has shown that high-ability learners can benefit from lower amounts, or less guidance, while low ability learners benefit from more concrete (specific and direct) guidance [Arr00; Luc99]. These findings inspired us to explore how data-driven hint algorithms could be used to derive less direct guidance to benefit high-ability learners.

3.3.1.2 Next-Step Hints

The Hint Factory is a data-driven approach developed to generate Next-Step hints for students applying rules to solve open-ended problems in well-defined domains where there are multiple valid solutions [Sta08; Sta13]. New innovations in generating assistance from individual pieces of previous student’s solutions have helped researchers extend the ideas of the Hint Factory to generate Next-Step hints for new domains including novice programming and linked list construction [Fos10; Riv17; Pri17a; Bar08]. A standard goal-directed hint sequence within a tutoring system is Point, Teach, and Bottom-out [Hum96]. Pointing hints attempt to remind the student of relevant material. Teaching hints describe how to apply the relevant material. Bottom-out hints tell the student the next step and specifically how to implement it. The Next-Step hints derived by the Hint Factory and used in our tutor are pointing hints that suggest a statement a student could derive using a single domain rule application. I have in past Hint Factory implementations provided students 4 levels of hints that (1) suggested the next step, (2) the specific rule, (3) the prior statements needed, and finally (4) a bottom-out hint with all this information. In this study, I use only level 1 pointing hints, and disabled hint levels 2-4. Sweller et al. makes the case that providing more explicit instruction is better for novices who need to establish those individual learning blocks before they can create their own mental models [Swe08; MS11]. However, research has shown that allowing students to make successful, unaided attempts at solving a problem can provide a higher learning benefit compared to explicit instruction showing them what to focus on [Koe07]. Hint Factory Next-Step hints have been shown to be successful in supporting student learning and problem-solving, with students having access to such hints in logic being 3 times more likely to complete the tutor than those without [Sta13]. These results suggest that Next-Step hints are direct and explicit enough to support learning, but since level 1 hints do not provide the full information to achieve a next-step, students must do some unaided exploration to achieve the suggested hint statement. On the other hand, Aleven et. al notes a “one size fits all” strategy for guidance is not likely beneficial [Ale00]. Hence, I are inspired to determine whether some even less direct data-driven hints may benefit high-ability learners.

3.3.1.3 Aptitude-Treatment Effect

Aptitude-treatment interactions have been widely studied in the educational domain. Prior research in instructional strategies [Cro77; Sno91] has shown the existence of aptitude-treatment interaction (ATI), where certain students are more sensitive to variations in the learning environment and may be affected differently by the treatment compared to less sensitive students who perform regardless
of the treatment. Educational researchers have discovered ATI effects based on prior experience level, prior working memory, and incoming self-regulated learning ability [Kal01; Leh16; Fuc19; Yeh15]. For example, Lehmann et al. explored the effect of working memory on learning outcomes in fluency/disfluency groups, where instructional materials had different levels of text legibility [Leh16]. Based on these findings, I believe that there could be an aptitude-treatment effect associated with hint type. I believe that students with lower incoming proficiency may be more sensitive to hint type.

3.3.1.4 Help Avoidance

Despite this considerable research on assistance, there is pervasive problem within ITSs called help avoidance, where students do not leverage the intelligence within the system for help [Ale04]. There are many reasons for help avoidance, one of which is that certain students may lack specific metacognitive skills like knowing when to ask for help [Ale00]. As a result, some ITSs employ unsolicited hints to prevent help avoidance (i.e. providing hints when needed without request) [Van06], and I adopt this unsolicited strategy here. In this study, I compared the impacts of unsolicited Next-Step (NS) and Waypoint (WP) hints within a logic proof tutor.

3.4 The Deep Thought Logic Proof Tutor

The Deep Thought tutor is used in the context of a discrete mathematics course where students first spend 2 weeks learning about truth tables, and proving each logic rule is true in class and in online multiple-choice homework assignments. Then, students learn about formal proofs, where students iteratively apply logic rules to a set of given statements to derive a specified conclusion. A formal proof works much like any multi-step procedural problem where domain principles are applied to given and previously-derived facts to derive and justify new statements. For example, in physics, students may be given values for mass and acceleration and be asked to determine force. They would then apply the domain principle of $F = m \times a$ along with the given values of $m$ and $a$ to derive a new statement about the value of $F$. In logic, each derived statement must have a justification which consists of the domain principle and the relevant prior statements it was applied to. This corresponds to the information used to derive $F$ in the previous physics example.

Within the discrete math course, students next complete partially-worked examples in a fill-in-the-blank type interface where they are given formal logic proofs with one missing part on each step - either the derived statement, or part of the justification that consists of the rule used to derive it and the statements the rule was applied to. Many example logic proofs are worked in class, with students asked to actively solve logic proofs in small groups, and students are provided with several full worked examples in handouts. After this class work and homework, students are assumed to have reasonable familiarity with logic rule application, but need practice in determining which rules to apply in service to a problem-solving goal. Students are then assigned to complete formal logic proofs using our propositional logic tutor called Deep Thought [Mos17].
The intention of the Deep Thought tutor is to provide students with practice on solving logic proofs with a focus on problem-solving efficiency in both time and the number of steps in their solutions, i.e. shorter proofs in less time, and ideally with few mistakes in justifying or deriving new statements. To do so, the tutor must provide basic functionalities including (1) correctness feedback on each step (on both justification and derived statements), and (2) automated detection of proof completion. Like a compiler, Deep Thought provides these functions that identify errors and clearly shows when a problem is complete but do little to help students with the overall goal of reaching a problem solution through deriving and justifying a series of well-chosen statements. To bridge this gap, the Hint Factory was created to provide data-driven assistance that could point students to appropriate subgoal statements to derive [Bar08; Sta08; Sta13].

Deep Thought allows students to solve logic proofs graphically as shown in Figure 3.1.

Figure 3.1 On the left of the screen is the Deep Thought workspace. Below the workspace are the hint button and hint message box, the rules are in the middle, and to the right is the Dialogue Box where messages related to unsolicited hints as well as problem information are given.

As stated above, Deep Thought is intended to teach students to solve proofs more efficiently, in terms of time and steps taken to reach the conclusion. The tutor presents proof problems as an initial set of given statements with a conclusion to derive from them using logic rules. Each statement, given or derived, is represented by a node, with the conclusion represented with a node with a question mark ‘?’ above it, indicating that it has not yet been justified (shown to be true using logic rules applied iteratively to the givens). Each problem-solving step consists of two parts: the justification and the derived statement. The justification is the set of 1-2 existing nodes and the rule applied to them, and the derived statement is the result. Students complete the justification by clicking to
select 1-2 nodes, and clicking on a rule to apply. Students then type in the derived statement that results from applying the rule to the selected statement nodes (see Figure 3.2). Throughout the tutor, including the pre- and post-test problems, Deep Thought provides immediate error feedback for mistakes - either in justifications or derived statements. If a student clicks on the wrong rule, or their derived statement does not follow from the selected nodes and rule, Deep Thought shows a popup message and records the error. For example, if a student selects two nodes and then clicks on the Simp rule, the error prompt reads “Rule requires one premise,” then fades away. If the student enters a derived statement that is true, and the justification (consisting of the selected nodes and rule to derive it) is correct, then a new node with the derived statement appears in the workspace. To complete a problem, the student must iteratively derive and justify new statements, until the conclusion statement is derived and justified. When students have completed a problem, the conclusion's question mark is removed, and it is visually connected to the givens through a series of derived nodes and arrows indicating their justifications. Since the system automatically checks each step and detects completion in all phases of the tutor, student solutions cannot be incorrect, but some may be more expert than others. Students are considered to have learned the topic when they perform well on the posttest, especially with regard to problem solutions with fewer steps and fewer mistakes in less time.

Deep Thought includes four phases: introduction, pretest, training, and posttest. The introduction consists of three problems including two worked examples, where students click through the addition of successive nodes until a conclusion is derived, and a third problem students solve alone to learn the interface. Then, students take the pretest consisting of a solving single problem with no hints available. The pretest is used to measure students' incoming proficiency and assign them to conditions via stratified sampling. Next, students solve 18 problems in the training section. For each training problem, the dialogue box provides information on what rules to focus on while solving a problem, such as “Think about the following rules for this problem: MP, Simp, Add.” Students also receive contextual, data-driven hints during training, including both unsolicited hints generated by the system and on-demand hints upon student request, all generated using the same Hint
Factory-type approach described below. After completing training, students take a more difficult, non-isomorphic posttest, where they must solve four problems without any help or assistance. Since the posttest is not isomorphic to the pre-test, I do not expect the post-test performance to be directly comparable to the pretest performance. Rather, I use the pretest to balance incoming proficiency across groups via stratified sampling, and focus on comparing post-test performance between groups.

Expert solutions for all tutor problems range from 5-8 steps, and student solutions typically contain 5-20 steps. Longer student solutions may simply be inefficient, taking more steps than needed, or they may contain unnecessary nodes that do not lie on a direct path from the givens to the conclusion. To help students avoid deriving unnecessary statements/nodes in the training phase, the tutor colors nodes based on their necessity and frequency in our historical dataset of correct solutions by past students. Nodes that were never necessary to derive the conclusion are colored gray, while frequently-necessary nodes are colored green, and infrequently-necessary nodes are colored yellow.

3.4.1 Assistance

In training problems, students receive unsolicited hints according to their assigned study condition, but all students may also request on-demand hints, derived using the same data-driven Hint Factory method, have the same content, and are added as statement nodes in the workspace nodes, and marked with '?' to show that they have not yet been justified. Unsolicited hints, called Assertions, are provided in the workspace labeled as "Goal" and resemble the nodes students use to derive new steps. In previous work, I showed this method of providing unsolicited Assertion hints resulted in better performance than text-based messages as a method of unsolicited hint delivery [MM20b]. For the remainder of the paper, I refer to both solicited and unsolicited hints as hints. As stated above, Deep Thought only provides pointing hints to suggest statements that can be derived; neither next-step nor Waypoint hints tell students which rules to use to derive them; rather, they help students solve problems by suggesting a subgoal that helps them break down multi-step problems. To use a hint in their proof, the suggested hint statement must first be justified by applying a rule to previously-justified or given statements. A hint statement is said to be adopted, or necessary to a student’s solution, if there is a path that connects the hint statement node to the conclusion.

Figure 3.3 shows the two forms of data-driven hints: Next-Step (NS) and Waypoints (WP) and how the students would approach deriving the suggested hint statement for each type. The tutor generates hints using historical student data from four semesters, each semester with approximately 250-300 students using the tutor. Both hint algorithms produce assistance based on the most frequent and efficient paths available in the student’s current proof. NS hints suggest the best proposition that can be derived in one step from the student’s current proof. For example, one unsolicited hint could add an E node, and the student would select ¬D ∧ E and choose the

---

1Unnecessary nodes in a complete solution are easy to detect because removing them does not disconnect the conclusion from the givens, but they are difficult to detect during problem solving.
“Simplification” rule to justify/derive the hint node.

![Figure 3.3](image)

**Figure 3.3** The left side shows the justification of a Next-Step hint and the right side shows the justification of a Waypoint hint.

I use the Hint Factory [Sta08] approach to generate hints. The Hint Factory [Sta08; Sta13; Bar11] is a data-driven method to generate hints by transforming historical student problem-solving attempts into a Markov Decision Process, using observed frequencies as transition probabilities, and estimating the expected value of each previously-observed problem state based on assigning rewards to complete solutions, small negative rewards (i.e. costs) to steps to positively reward more efficient solutions, and large negative rewards to errors to de-emphasize solutions that cause many students to make mistakes. Individual student problem-solving attempts are represented by a series of states, or snapshots of the work done so far, where transitions occur between states when students add or delete problem nodes, or make an error. The Hint Factory is described in detail in Barnes and Stamper’s chapter in the 2011 Handbook on Educational Data Mining [Bar11]. All student solutions are combined into an interaction network [Eag12] that reflects all previously-observed solutions to
one specific problem. When a hint is requested by the student or tutor, the Hint Factory is used to select a target problem-solving state with the highest expected value. Note that this process can be done offline, and a simple table can be used to store problem-solving states and their corresponding hint content for real-time hint provision. Then, the latest statement derived in that state is used as the pointing hint to help students know what to try to derive next. As stated above, in this study I do not provide further information on how to derive or justify the suggested statement, meaning that all hints in this paper can be considered as partially-worked example steps.

In this study, Next-Step Hints are derived as above, with the target state is selected to be the one with the highest expected value that occurs within one rule application from the student’s current state. This corresponds to the next-step hints derived in all of our prior work using Hint Factory [Bar08; Bar11; Sta13; Sta08; Eag14b; Eag12; Pri17a; Pri17b]. Since Next-Step hints are partially-worked, they allow students to focus on how to justify them, and reflect on why they were suggested. This removes a considerable load; without a hint, students must also search among many options for the best what to derive next.

A primary motivation for this study was to determine a simple way to extend the Hint Factory to provide less direct data-driven hints without the need for expert authoring. In our prior work, we derived a new method called data-driven Approach Maps, that applies hierarchical graph mining to interaction networks to discover problem-solving states that represent critical junctures in problem-solving attempts [Eag14b], and the subgoals I observed occurred every 2-3 steps/states in our short logic proofs (which are typically 5-12 steps long). To generate Waypoints without the need to apply data-driven Approach Maps, I modified the Hint Factory to select a target statement that was 2-3 steps away from the current state. Among states that were 2 or 3 steps away, I selected the state with a higher frequency within prior correct solutions. This resulted primarily in states that need only two rule applications to derive, since the diversity of student solutions means that frequency typically decreases in interaction networks the further states are from the start. By expert review of a random sample of Waypoints, I verified that this simple algorithm results in similar hints to those generated using data-driven Approach Maps [Eag14b]. Waypoints are intended to serve as subgoals, giving students more room to explore the solution space and develop their own problem-solving strategies. Since Waypoints cannot be achieved with a single rule application, they require students to make their own problem-solving plan to derive them, considering the existing problem statements and how rules might be applied to them to derive and justify the suggested Waypoint statement.

I consider a continuum of goals for students, where Next-Step hints ideally take one step to derive, Waypoints take 2-3, and the problem conclusion takes about 5 expert steps. With longer problems or more complex problem domains like programming, I would recommend using a more complex algorithm to select Waypoints if they were shown to be effective. In logic proofs, the shortest proof is considered to be the best, so simple metrics on interaction networks can quickly discover optimal solutions and those that many students can discover.

Deep Thought includes several measures intended to prevent gaming the system, where students attempt to use system features to avoid work, or help abuse, where students request hints when
they do not need them [Bak04]. First, whenever an unjustified hint is already in the workspace, students may not receive another hint, whether it was solicited or provided automatically by the tutor. Second, no further details are provided for any hint, meaning there is no such thing as a bottom-out hint in this study. All hints provide a target statement to derive, appearing as a node with a ‘?’ in the workspace.

Within our tutor, even though students often have difficulty and hints are readily available via the hint button, most students do not request assistance. In Fall 2017, students using our tutor requested a median of zero hints per problem. Research has shown that there is a pervasive problem with help avoidance within intelligent tutoring systems [Ale04]. As a result, some ITSs employ unsolicited hints to prevent help avoidance [Van06]. In this study, to enable us to compare the impact of hint type, I periodically provided unsolicited hints to students based on the condition they were assigned, as described below.

3.5 Method

The Deep Thought tutor was used as a homework assignment for an undergraduate ‘discrete mathematics for computer scientists’ course in the Fall 2018 semester at a large research university. I analyzed 143 students’ data from two test conditions to investigate the impact of hint type on student performance and behavior. Both conditions were identical except for hint type, Next-Step or Waypoint. I used stratified sampling, using a median split on students by pretest performance, then randomly assigning to Next-Step hints (NS, \( n = 71 \)) or Waypoints (WP, \( n = 72 \)), ensuring both conditions were balanced in incoming knowledge. Before analysis, students who dropped the tutor before completion and students with technical errors in their data were removed (NS \( n = 15 \), WP \( n = 14 \)) leaving 56 students in the NS condition and 58 students in the WP condition for a total of 114 students. To study the impact of hint type, I implemented unsolicited hints so they appear randomly and with enough uniformity and frequency that even students with short proofs would receive hints. One limitation of this method of providing hints is that hints were not necessarily provided when they were most needed, which may affect learning outcomes. However, since students in tutor rarely request hints, it was necessary to provide the hints automatically and frequently to enable us to evaluate my hypotheses. For the Next-Step group, I capped the number of unsolicited hints at 1/3 of the problem length and checking every 2-3 steps to see if a hint was still extant in the workspace (e.g. it was not yet justified). If a hint still remained, the algorithm did not provide a new hint, but if there were no hints on the screen, a new one was provided. Since Waypoints take more steps to derive, they remained unjustified for longer, and thus resulted in fewer Waypoint hints by design. Note that students can delete problem nodes at any time (excluding the givens and conclusion), and this includes hint nodes, even if they are not yet justified.
3.5.1 Hypotheses

The goals of this study were to 1) evaluate the effectiveness of a new hint type, Waypoint hints, 2) compare the impacts of Waypoints and Next-step hints on performance, and 3) determine if proficiency had an effect on which hint type was more beneficial. Based on prior literature, I developed the following hypotheses:

- \( H_1 \): Next-Step hints will improve performance for students with lower incoming proficiency.
- \( H_2 \): Waypoint hints will improve performance for students with higher incoming proficiency.
- \( H_3 \): Waypoint hints will be more difficult to derive, resulting in a lower justification rate and performance during training compared to Next-Steps.

These hypotheses were based on the basic assumption that Waypoint hints are more difficult to justify and adopt, since Waypoints require students to derive more steps to justify them. On the other hand, this challenge may be precisely what high-proficiency students need for improved learning. To evaluate these hypotheses, I focused on the performance metrics discussed below.

3.5.2 Performance Evaluation Metrics

In this section I describe the metrics used to evaluate student performance. Recall that the tutor begins with an introduction with two worked examples and one practice problem followed by the pretest. I used each student’s pretest score to measure incoming knowledge/proficiency (See Equation 3.1). A student’s score is a combination of percentiles for the pretest time, number of steps, and accuracy on a single problem, ranking students based on how fast, efficient, and accurate they are compared to their peers. I chose these features because they each represent a different aspect of a student’s problem solving experience. Recall that the tutor was designed to improve time and steps to solve problems, and assumes a basic level of fluency or accuracy on rule applications. The score includes all three to ensure that my interventions do not decrease accuracy while attempting to improve (decrease) time and steps. For example, a student may take a short amount of time on a problem, but make many mistakes resulting in a lower accuracy. I use a median split on the combined pretest score to assign students into High and Low proficiency groups for some analyses.

\[
PretestScore < -(1 - TotalTime) \cdot .5 + (1 - TotalSteps \cdot .3) + Accuracy \cdot .2 \quad (3.1)
\]

During the training phase, students solve 18 training problems with access to both unsolicited and on-demand hints. The Posttest consists of four final unassisted problems. I investigated pre- to posttest changes as well as performance impacts on time spent solving a problem, total attempted steps, and accuracy. Total time is counted from the moment a problem begins until it is solved by deriving and justifying the conclusion. Total steps in a problem include any attempt at deriving a new node, which includes correct and incorrect steps. Accuracy is the percentage of correct out of the total steps, which is expected to start relatively high due to prior exposure in the class, and...
increase as students practice. Note that the tutor is not designed or assumed to promote large improvements in accuracy, since no penalties are assigned for incorrect rule applications and the tutor simply alerts students upon wrong rule applications and students may try again, even within the pre- and post-tests. Further, problems require new rules and become more difficult as the students progress. As I seek primarily to promote more efficient problem solving, I focus more on steps and time per problem while maintaining reasonable accuracy. This is because it is more difficult for students to learn to determine which steps to derive to achieve shorter, more efficient proofs, compared to learning how to apply the rules, which can be done by memorization and simple practice. Deep Thought is built primarily to allow students to practice with the strategy of problem solving, rather than fluency with rules, most of which are assumed to be learned before the tutor.

One important thing to note is that Deep Thought does not include eye-tracking, and the unsolicited hints are provided regardless of whether a student needs them or not, so I cannot determine precisely whether students followed a hint or incidentally derived the hint statement. Therefore, I have defined metrics to quantify when students justified a hint by selecting the statements and rule needed to derive it, as well as when the students adopted a hint by first justifying it and then using it directly on their path to derive the conclusion. These two hint-specific metrics are the hint Justification Rate and Adoption Rate.

The hint Justification Rate is the percentage of unsolicited hints justified (correctly identifying the rule and prior nodes needed to derive the suggested node) divided by the total number of hints given across the training problems. A hint is said to be justified when a student applies logic rules to existing logic statements to derive the hinted logic statement, and when a hint is justified, the tutor removes its '?' and connects it to its predecessor nodes with arrows labeled with the rule used to derive it. A hint justification provides evidence that a student noticed the hint and knew how to apply rules to justify it, but do not tell the full story. As in any problem-solving context, statements can be derived that are not needed in a final solution. Therefore, I also measure hint Adoption Rate, whether a hint contributes towards deriving the conclusion. A justified hint can be reached on a path from the problem's given statements. When a hint is adopted, it must first be justified and then become necessary to a student's final solution – in other words, the problem would be incomplete if the hinted statement were removed. This is shown visually when a directed path can be found from the hinted statement node to the problem conclusion. Figure 3.4 shows a completed problem with labels indicating which nodes are considered justified and which nodes were also adopted for the solution.

I also investigated impacts on help-seeking through the number of on-demand hint requests (when students click the “Get Suggestion” button). Total Requests represents the number of hint requests during the training portion. Data were analyzed to compare groups for the pretest, training, and posttest portions of the tutor. Within each hint group, I also compared performance of students with High or Low pretest scores, based on a median split on the pretest score.

To determine significant differences between hint types, I applied one way ANCOVA using the
pretest as a covariate with Benjamini-Hochberg corrections to account for multiple tests. To check that the data met assumptions for ANCOVA, I used the the Shapiro-Wilk's W test and Levene's test, as well as visually inspecting the data via Q-Q plots and histograms. Data that did not meet the assumptions were transformed using log or square-root transformations, then re-inspected. Data reported in tables For clarity, all data in tables are reported before transformation.

### 3.6 Results

Table 3.1 shows the overall hint metrics for each group during training. I expected *Total Added* \((p < 0.01)\) and *Steps Until Justified* \((p < 0.01)\) metrics to be significantly different, since each step of a problem can have a unique Next-Step (NS) but one Waypoint (WP) requires multiple steps to be derived. Based on prior literature on help avoidance and low help usage within tutors [Pri17a], I were pleasantly surprised to find students in both groups had relatively high justification and adoption rates. The Next-Step group justified a significantly \((p < 0.01)\) higher percentage of hints, as shown by the *Justification Rate*. Additionally, of the justified hints, I also saw a significantly \((p = 0.01)\) lower Adoption Rate of the WP hints in students’ final proofs. Although this is a relatively high number for both groups, the WP group’s lower justification and adoption rates are concerning. This provides evidence in support of \(H_3\) that Waypoint hints would be harder to derive; however, this evidence does not address whether this was due to the difficulty of the WP hints or students’ lack of effort to derive them. I explore the possible reasons for these differences later in this section.
Table 3.1 Hint metrics during training. For ANCOVA results controlling for the pretest score, p-values that are at least marginally significant are **bolded** and significant values also have an asterisk*.

<table>
<thead>
<tr>
<th>Metric</th>
<th>NS n = 56</th>
<th>Mean(SD)</th>
<th>WP n = 58</th>
<th>Mean(SD)</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Justification Rate</td>
<td>89%(7)</td>
<td>84%(12)</td>
<td></td>
<td></td>
<td>&lt;0.01*</td>
</tr>
<tr>
<td>Adoption Rate</td>
<td>83%(10)</td>
<td>74%(17)</td>
<td></td>
<td></td>
<td>0.01*</td>
</tr>
<tr>
<td>Steps Until Justified</td>
<td>1.1(0.1)</td>
<td>2.2(0.3)</td>
<td></td>
<td></td>
<td>&lt;0.01*</td>
</tr>
<tr>
<td>Total Added</td>
<td>49(9)</td>
<td>30(7)</td>
<td></td>
<td></td>
<td>&lt;0.01*</td>
</tr>
</tbody>
</table>

To understand the overall impact of Next-Step versus Waypoint hints, I examined the performance for both groups for the tutor pretest, training, and posttest, for all students regardless of incoming proficiency as shown in Table 3.2. There were no significant differences between the WP and NS groups on the pretest, although the pretest was slightly worse for the NS group. During training, the NS group significantly outperformed the WP group with fewer steps, less time, better accuracy and overall score (Total Time \(p < 0.01\), Total Steps \(p = 0.02\), Accuracy \(p = 0.01\)). The WP group, on average, took 20 minutes longer, took 36 more steps, and had 5% lower accuracy on the training problems. These results suggest that Next-Step hints had a stronger impact than hypothesized in \(H_1\) during training, with all Next-Step students outperforming all WP students. I examined help-seeking behaviors during training and found the NS group requested significantly more hints, although still a small number overall (approximately 1 per problem for NS versus 0.5 per problem on average for WP), so it is not likely that hint requests account for the difference in training performance.

More importantly, on the posttest, the NS group significantly outperformed the Waypoint group on total steps \(p = 0.02\) and accuracy \(p = 0.05\). The WP group had 28 more total steps and had a 5% lower accuracy, on average, on the posttest. There was a marginally significant difference between groups for the total time on the posttest \(p = 0.06\), with the WP group spending roughly 10 more minutes on the posttest. These results suggest that Next-Step hints had a stronger impact than hypothesized, showing that, overall, the NS group performed better during training and the posttest. I believe that Next-Steps allow students to focus on solving one step, which I hypothesized would reduce time spent (since students did not have to determine *what* to derive next when receiving hints, just the *how*), and total steps (since the suggested hints were efficient).

### 3.6.1 Effects on High- and Low- Pretest Groups

Recall that my hypotheses focused on the differential impact of hints based on incoming proficiency and the difficulty of applying Next-Step versus Waypoint hints:

- \(H_1\): Next-Step hints will improve performance for students with lower incoming proficiency.
Table 3.2 Performance metrics for each group on the pretest, training, and posttest; p-values that are at least marginally significant when applying ANCOVA controlled for pretest are bold and those that are significant also have an asterisk.

<table>
<thead>
<tr>
<th>Metric</th>
<th>NS (n = 56) Mean(SD)</th>
<th>WP (n = 58) Mean(SD)</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pretest</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Time</td>
<td>6.5(12)</td>
<td>5.5(9)</td>
<td>0.54</td>
</tr>
<tr>
<td>Total Steps</td>
<td>19(35)</td>
<td>15(16)</td>
<td>0.84</td>
</tr>
<tr>
<td>Accuracy</td>
<td>67%(22)</td>
<td>70%(23)</td>
<td>0.57</td>
</tr>
<tr>
<td><strong>Training</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Time (min)</td>
<td>58(24)</td>
<td>77(42)</td>
<td>&lt;0.01*</td>
</tr>
<tr>
<td>Total Steps</td>
<td>186(65)</td>
<td>222(99)</td>
<td>0.02*</td>
</tr>
<tr>
<td>Accuracy</td>
<td>74%(10)</td>
<td>69%(10)</td>
<td>0.01*</td>
</tr>
<tr>
<td>Total Requests</td>
<td>15(29)</td>
<td>7(9)</td>
<td>0.04*</td>
</tr>
<tr>
<td><strong>Posttest</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Time (min)</td>
<td>33(21)</td>
<td>42(34)</td>
<td>0.06</td>
</tr>
<tr>
<td>Total Steps</td>
<td>99(40)</td>
<td>127(87)</td>
<td>0.02*</td>
</tr>
<tr>
<td>Accuracy</td>
<td>71%(12)</td>
<td>66%(10)</td>
<td>0.05*</td>
</tr>
</tbody>
</table>

- $H_2$: Waypoint hints will improve performance for students with higher incoming proficiency.

- $H_3$: Waypoint hints will be more difficult to derive, resulting in a lower justification rate and performance during training compared to Next-Steps.

To investigate these hypotheses, I checked for differences in performance between prior proficiency groups within each group. I performed a median-split for incoming proficiency based on pretest scores and compared performance metrics across groups and proficiency (NS-High $n = 27$, WP-High $n = 30$, NS-Low $n = 29$, WP-Low $n = 30$).

First, I examined performance metrics for the High group, shown in Table 3.3. There were no significant differences between the NS and WP High groups on the pretest. For the training, the WP group took longer and made more mistakes, as indicated by the Total Time ($p < 0.01$) and Accuracy ($p = 0.04$). There was a marginally significant difference in the Total Steps between the groups ($p = 0.09$) with the WP group attempting more steps. The hint justification rate was also significantly lower for the WP group ($p < 0.01$). These results indicate that the WP-High group struggled with following hints, which may have led to them spending more time trying to figure out how to solve the problem. For the posttest, there were no significant differences between the NS-High and WP-High groups, although the WP-High group performed worse on average. This result confirms an aptitude-treatment interaction effect for high proficiency students where the treatment did not result in different results; i.e. high proficiency students were not as sensitive to the treatment choice (Next-Step or Waypoint). This means that $H_2$ was rejected; Waypoint hints did not improve performance for higher proficiency students.

Next, I examined performance metrics for the Low pretest group. There were no significant differences between the NS and WP Low groups on the pretest. For the training, the WP took longer
Table 3.3 Performance metrics between the NS and WP High proficiency groups for the pretest, training, and posttest of the tutor. ANOVA results are reported for the pretest. ANCOVA results, controlling for the pretest, are reported for the training and posttest, with p-values that are at least marginally significant in bold and significant p-values also have an asterisk *.

<table>
<thead>
<tr>
<th>Metric</th>
<th>NS</th>
<th>WP</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean(SD)</td>
<td>Mean(SD)</td>
<td></td>
</tr>
<tr>
<td><strong>Pretest</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Time (min)</td>
<td>1.7(0.64)</td>
<td>1.6(0.86)</td>
<td>0.12</td>
</tr>
<tr>
<td>Total Steps</td>
<td>5.6(1.7)</td>
<td>5.5(1.9)</td>
<td>0.68</td>
</tr>
<tr>
<td>Accuracy</td>
<td>87%(12)</td>
<td>87%(14)</td>
<td>0.91</td>
</tr>
<tr>
<td><strong>Training</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Time (min)</td>
<td>50(23)</td>
<td>64(41)</td>
<td>&lt;0.01*</td>
</tr>
<tr>
<td>Total Steps</td>
<td>164(57)</td>
<td>198(86)</td>
<td>0.09</td>
</tr>
<tr>
<td>Accuracy</td>
<td>73%(10)</td>
<td>78%(8)</td>
<td>0.04*</td>
</tr>
<tr>
<td>Total Requests</td>
<td>8(8)</td>
<td>4(5)</td>
<td>0.22</td>
</tr>
<tr>
<td>Justification Rate</td>
<td>89%(7)</td>
<td>81%(15)</td>
<td>&lt;0.01*</td>
</tr>
<tr>
<td>Adoption Rate</td>
<td>83%(9)</td>
<td>%74(17)</td>
<td>&lt;0.01*</td>
</tr>
<tr>
<td><strong>Posttest</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Time (min)</td>
<td>33(21)</td>
<td>40(34)</td>
<td>0.29</td>
</tr>
<tr>
<td>Total Steps</td>
<td>81(48)</td>
<td>98(42)</td>
<td>0.18</td>
</tr>
<tr>
<td>Accuracy</td>
<td>71%(12)</td>
<td>67%(9)</td>
<td>0.14</td>
</tr>
</tbody>
</table>

and attempted more steps, as indicated by the Total Time ($p = 0.01$) and Total Steps ($p = 0.03$). The hint justification rate was also significantly lower for the WP group ($p < 0.01$). These results follow a similar pattern as the High group, in that the WP group performed worse overall in the training and were less able to justify the hints. For the posttest, the WP group continued the pattern of taking longer ($p = 0.06$) and attempting more steps ($p = 0.06$) with marginally significant results, indicating that the (hypothesized) worse performance in the training portion may have transferred to their overall proof solving strategies on the posttest. These results confirm hypothesis $H_3$ that Waypoints are more difficult for students and have a negative impact on training performance.

I hypothesized in $H_1$ that Next-Step hints would improve (training and posttest) performance compared to Waypoint hints, for low proficiency students. The overall performance (Table 3.2) confirmed that the Next-Step hint group produced better training and posttest performance. However, Table 3.4 confirms that the benefits in the posttest are more prominently seen with the students with lower incoming proficiencies, confirming $H_1$.

I hypothesized in $H_3$ that the Waypoint hints would cause lower justification rates and worse training performance due to their increased difficulty, which is seen with both the WP-High and WP-Low groups, confirming my $H_3$ hypothesis. There was also a significant difference in the Adoption rates between the NS and WP groups for both High and Low students, with the WP adoption rates being lower. This suggests that students were not, in fact, able to independently discover the strategies that underlie the WP hints.

Although I expected a lower hint justification rate in the WP group, I thought that the increase
Table 3.4 Performance metrics between the NS and WP Low proficiency groups for the pretest, training, and posttest. ANOVA results are reported for the pretest. ANCOVA results, controlling for the pretest are reported for the training and posttest; p-values that are at least marginally significant are in bold and significant p-values also have an asterisk *.

<table>
<thead>
<tr>
<th>Metric</th>
<th>NS Mean(SD)</th>
<th>WP Mean(SD)</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pretest</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Time (min)</td>
<td>10(11)</td>
<td>11(15)</td>
<td>0.80</td>
</tr>
<tr>
<td>Total Steps</td>
<td>31(48)</td>
<td>25(19)</td>
<td>0.80</td>
</tr>
<tr>
<td>Accuracy</td>
<td>49%(13)</td>
<td>51%(16)</td>
<td>0.88</td>
</tr>
<tr>
<td>Training</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Time (min)</td>
<td>66(23)</td>
<td>89(39)</td>
<td>0.01*</td>
</tr>
<tr>
<td>Total Steps</td>
<td>206(67)</td>
<td>249(107)</td>
<td>0.03*</td>
</tr>
<tr>
<td>Accuracy</td>
<td>69%(10)</td>
<td>65%(9)</td>
<td>0.13</td>
</tr>
<tr>
<td>Total Requests</td>
<td>13(13)</td>
<td>10(10)</td>
<td>0.55</td>
</tr>
<tr>
<td>Justification Rate</td>
<td>90%(8)</td>
<td>81%(14)</td>
<td>&lt;0.01*</td>
</tr>
<tr>
<td>Adoption Rate</td>
<td>83(10)</td>
<td>74(17)</td>
<td>0.02*</td>
</tr>
<tr>
<td>Posttest</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Time (min)</td>
<td>33(20)</td>
<td>43(35)</td>
<td>0.06</td>
</tr>
<tr>
<td>Total Steps</td>
<td>99(39)</td>
<td>128(94)</td>
<td>0.06</td>
</tr>
<tr>
<td>Accuracy</td>
<td>69%(12)</td>
<td>65%(10)</td>
<td>0.17</td>
</tr>
</tbody>
</table>

in difficulty would be beneficial to high proficiency students by allowing them more exploration of the problem space. Therefore, I hypothesized in H<sub>2</sub> that higher incoming proficiency students would do better on the posttest after experiencing the WP hints in training. However, that is not the case. The WP-High group was only able to perform similarly to the NS-High group and overall performed worse, although not significantly. Therefore, H<sub>2</sub> is rejected. However, the results do seem to indicate that the high incoming-proficiency students were less affected by the treatment than the low incoming-proficiency students. As mentioned earlier, I expected that an aptitude-treatment interaction (ATI) might occur, where certain students are more sensitive to variations in the learning environment and may be affected differently by the treatment compared to less sensitive (more proficient) students who are able to perform well regardless of treatment.

3.6.2 Did Waypoints help with strategy for those who could utilize them?

Although the performance results caused us to reject H<sub>2</sub>, I wanted to investigate whether WP hints provided strategy-related benefits to those students who were able to use them. Therefore, I performed correlation analyses using the Pearson correlation coefficient between the hint Justification and Adoption rates with posttest performance metrics. Table 3.5 shows the significant correlations of hint Adoption and Justification Rates with performance metrics for NS and WP groups on the posttest, as well as correlations with the incoming proficiency groups.

For the NS group, the only significant correlation found was for the NS-High group between hint Adoption Rate and Total Steps (p = 0.06), showing a moderate, negative correlation. This indicates
that students in the NS-High group attempted fewer steps in the posttest (a better result) if they adopted more of the NS hints during training. For the WP group, there are moderate, negative correlations of Justification rate with Total Time ($p = 0.03$) and with Total Steps ($p = 0.01$), and also of Adoption Rate with Total Time ($p < 0.01$) and Total Steps ($p < 0.01$). So justifying and adopting WPs were both associated with more efficient proofs that were shorter and achieved in less time.

There was also a significant moderate, negative correlation for the WP-Low group between Total Steps ($p = 0.04$) and the hint Adoption rate. For the WP-High group, there was a similar moderate, negative correlation between Total Steps ($p = 0.02$) and hint Adoption Rate, but the WP-High group also had moderate, negative correlation between Total Time ($p = 0.02$) and hint Adoption Rate. This result aligns with my reasoning behind $H_2$, that Waypoint hints should improve efficiency- and time-related metrics on the posttest, especially for higher proficiency students. However, ultimately, the WP students performed worse. Based on these results, I conclude that more support may be needed for WPs so that students can utilize them as well as NS hints to better achieve efficiency-related benefits.

### Table 3.5

<table>
<thead>
<tr>
<th>Condition</th>
<th>Split</th>
<th>Metric-Pair</th>
<th>Corr</th>
<th>$p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>NS High</td>
<td>Adoption-Total Steps</td>
<td>-0.38</td>
<td>0.06</td>
<td></td>
</tr>
<tr>
<td>WP All</td>
<td>Justification-Total Time (min)</td>
<td>-0.30</td>
<td><em>0.03</em></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Justification-Total Steps</td>
<td>-0.32</td>
<td><em>0.01</em></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Adoption-Total Time (min)</td>
<td>-0.35</td>
<td>&lt;0.01*</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Adoption-Total Steps</td>
<td>-0.40</td>
<td>&lt;0.01*</td>
<td></td>
</tr>
</tbody>
</table>

3.6.3 What are the circumstances when Assistance was not used?

To understand if the WP hints were actually harder to derive, as hypothesized $H_3$, I investigated how many unused (unjustified) hints were attempted to be justified to better understand the circumstances surrounding the significantly lower difference in hint Justification Rate of the WP group as shown in Table 3.1 and the significantly worse performance by the WP group in the training as shown in Table 3.2. The hint Justification and Adoption rates only tell us that students were, or were not, using the hints. Therefore, I conducted analyses to see if the WP hints were truly harder to derive ($H_3$). This would be indicated by the students attempting to work towards the hint, and not succeeding, versus ignoring the hint. Because the WP hints are further away than the NS hint, students may not be able to see what they need to do next and just ignore the hint. I examined the steps taken after a hint was added (3 steps ahead for NS and 5 steps ahead for WP) to determine if the student was
Table 3.6 The total unused (unjustified) hints, percentage of hints unused out of all hints added, and the percentage of the unused hints that were attempted to be derived between the NS and WP group. For ANCOVA results controlling for the pretest score, p-values that are at least marginally significant are in bold and significant values also have an asterisk *.

<table>
<thead>
<tr>
<th>Metric</th>
<th>NS</th>
<th>WP</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Unused</td>
<td>5.4(4.4)</td>
<td>6.6(6.4)</td>
<td>0.49</td>
</tr>
<tr>
<td>% Unused/Total</td>
<td>10.4%(7.2)</td>
<td>19.4%(14.8)</td>
<td>&lt;0.01*</td>
</tr>
<tr>
<td>% Attempted/Unused</td>
<td>57.0%(37.4)</td>
<td>72.6%(28.9)</td>
<td>0.02*</td>
</tr>
</tbody>
</table>

attempting to justify the hint. If a majority of the steps examined contained variables that were also seen in the hint, it was considered attempted. Table 3.6 shows the total unused hints. Total Unused represents the total number of unused hints per person in each group. The % Unused/Total is the total number of unused hints divided by the total number of hints that were added, which provides a clearer picture of the relative percentage of hints that were left unused by each student compared to how many they were being given. The % Attempted/Unused is the percentage of the unused hints that were attempted. The WP group left a significantly larger amount of hints unused compared to the NS group (p < 0.01). Interestingly, the WP group were also attempting a larger amount of unused hints compared to the NS group (p = 0.02).

To understand when unsolicited hints were not justified, I determined the circumstances when this occurred and illustrate several situations: when students attempted to use the hint, and what the eventual outcome was—either Gave Up or Solved Without using the hint. Gave Up represents any actions that end the problem without solving it, such as restarting or skipping the problem. In this situation, students had a hint on the screen, worked a few steps, then clicked the restart or skip button without justifying the hint. When a student clicks restart or skip, this erases all current progress on the problem. I considered this to be “giving up” because the student is removing all progress made on the current problem by taking these actions, which is concerning given that a hint was on the screen. Solved Without represents when students completed a proof with an unjustified hint still on the screen. In this case, students have a hint but eventually solve the problem without using the hint. This indicates that the hint was ignored, or at the very least, was not essential to solving the proof. I am less concerned with this case because the students were able to progress. However, since the hint is the most efficient step to work towards, any student who avoided it took a less efficient route to solve the problem. Lastly, although students had the option to delete a hint, no deletions were observed possibly due to students not knowing how to delete the hint.

Table 3.7 details the two cases in which a hint was added, but the student did not justify it. For significant differences, ANCOVA was used with the pretest score as the covariate. The Total Unused, % Unused/Total and the % Attempted/Unused are defined above. I also examined how many steps the students took after a hint was given but before they gave up or solved the proof to determine how much effort was put into trying to derive the hint. Steps Before is the number of steps the student
attempted after receiving the hint until they gave up or solved the proof. This metric was added to see how long students were trying to work on the problem after the hint was given.

Table 3.7 Comparison of unused hints of each subtype by amount, percentage that were attempted, and steps before the action occurred.

<table>
<thead>
<tr>
<th></th>
<th>Gave Up</th>
<th>Solved Without</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NS (n = 56)</td>
<td>WP (n = 58)</td>
</tr>
<tr>
<td></td>
<td>Mean(SD)</td>
<td>Mean(SD)</td>
</tr>
<tr>
<td>Total Unused</td>
<td>2.9(3.5)</td>
<td>4.0(5.20)</td>
</tr>
<tr>
<td>% Unused/Total</td>
<td>46%(34)</td>
<td>47%(34)</td>
</tr>
<tr>
<td>% Attempted/Unused</td>
<td>48.3%(43.8)</td>
<td>71.5%(28.6)</td>
</tr>
<tr>
<td>Steps Before</td>
<td>1.6(1.7)</td>
<td>2.4(2.0)</td>
</tr>
</tbody>
</table>

The WP group attempted to derive a significantly higher number of the unused hints before giving up ($p = 0.02$). There were no significant differences in the Total Unused for either cases (Gave Up: $p = 0.14$ and Solved Without: $p = 0.92$), the % Attempted/Total for the Solved Without case ($p = 0.13$), or the Steps Before for either cases (Gave Up: $p = 0.13$ and Solved Without: $p = 0.79$). Therefore, both groups of students had a similar distribution of unused hints in both cases; however, the WP group attempted to derive a significantly higher percentage of the hints in the Gave Up case. This result indicates that the WP students had attempted to make progress towards the hints, were unable to justify them, and then gave up. This is more concerning than giving up on a problem in which they had not attempted to derive the hint, and indicates the WP hints may have been too hard to derive. The purpose of this analysis was to investigate $H_3$ and determine the circumstances surrounding why the WP group had a significantly lower hint Justification Rate than the NS group. The results provide evidence in support of $H_3$ that the Waypoint hints were harder for students to derive.

### 3.7 Discussion

This work aims to explore the extension of a Next-Step hint generator to easily create subgoal-inspired assistance. The Next-Step group saw overall the best performance for both the training and posttest, including the students with lower incoming proficiencies providing supporting evidence for $H_1$. My results indicated that the Waypoint group performed overall worse in both training and posttest causing us to reject $H_2$. Results also showed that the lower proficiency students, specifically, were less able to utilize this form of assistance; however, students who were able to utilize Waypoints did see benefits in terms of time and efficiency on the posttest. Furthermore, I explored the circumstances surrounding when hints were not utilized and found that students in the Waypoint group attempted a larger percentage of the hints before giving up, providing evidence in support of $H_3$ that Waypoints would be harder to derive. In this section, I discuss the trade-offs of the two hint
3.7.1 Waypoint hints

WPs were intended for students to learn strategies for solving proofs by breaking the problem into smaller subgoals and providing students with more independent problem solving experience than NS. However, the majority of WP students appeared to have struggled with WP hints instead, a trade-off of the assistance dilemma [Koe07]. The WP group performed worse overall in both training and posttest portions of the tutor (see Table 3.2). Another interesting result, shown in Section 5.1, is that the WP Low-pretest group has a significantly lower Justification Rate and marginally significant decrease in posttest performance metrics. This aligns with literature showing that lower proficiency learners are less able to use abstract guidance [Kir06; Swe08]. Therefore, the WP hints might not provide enough guidance for students. Research has shown that complex assistance can hinder learning by taxing cognitive load [Swe88; Swe11], which can happen when learners try to process new information and incorporate complex assistance at the same time and “thus forcing learners to use random search procedures” [Kal07]. This is a limitation of my study as Waypoints may produce better results with more scaffolding.

The Justification Rate being significantly lower for the WP group indicates that the lower performance may be due to an inability to properly use the assistance (see Table 3.1). This is partially supported by Table 3.6 and Table 3.7, which shows that the WP group had a higher percentage of attempts to justify a hint without succeeding, compared to the NS group. The Adoption Rate being significantly lower for the WP group indicates that, even when students in the WP group were able to justify the hints, they were less able to adopt them to connect the WP hints to the conclusion. Due to the design of the hint being a few steps away, students could end up on a solution path different from the path initially given by the WP. Consequently, students who were unable to justify the WP or adopt it into the solution were not following the most efficient path, hindering their ability to learn from the strategies behind the WP hints.

One positive result with Waypoints is shown in Table 3.5, with respect to the significant negative correlations of Justification and Adoption rate with total time and total steps on the posttest. Students who were able to justify and adopt the WP saw benefits of shorter time and steps on the posttest. This correlation aligns with my original intention of using WP to support strategy development by helping students become more efficient in their problem solving process. Therefore, it is possible that students with more experience and domain knowledge may better utilize Waypoints and receive strategy-related benefits. Based on these results, WPs can be improved by providing more information (perhaps automatically provided once I detect that a student is unsuccessfully attempting to justify the hint) or incorporating ideas from recent research with promising methods of scaffolding goal-based hints [Mar19].
3.7.2 Next-Step hints

The total time, total steps, and accuracy were significantly different, or trending towards significance, between groups as shown in Table 3.2 for the training and posttest. Since the groups had similar pretest scores, these results show that both the NS and WP groups came into the tutor performing similarly, but by the posttest the two groups had diverged; the NS group had higher accuracy and fewer total steps. Furthermore, the NS group were able to increase their accuracy between the non-isomorphic pre- and posttest compared to the WP group who did not show such improvements. This was perhaps due to the increased practice in applying rules to justify both unsolicited and on-demand hints - since the NS group received and justified significantly more hints in both of these categories.

The differences in time, steps, and accuracy between the groups show that NSs were more beneficial for students. As shown in Table 3.1, there is a significant difference in the higher Justification Rate for NSs. I believe these results may be due to the alignment of NSs with novice’s bottom-up problem solving approaches that focus on what to do in the short term [And84; Guz95; Swe08]. NSs may also have potentially resulted in an overall lower cognitive load [Kal11], though this supposition is only based on their design and not data from students. As a justification, students considering NSs only needed to think about which nodes and rules could be used to derive the NS. In contrast, WP students needed to think about which nodes to use, which rules to apply, and what intermediate steps they would have to achieve before deriving the WP.

Interestingly, the NS group requested more on-demand hints (see 3.2). This suggests that the NS group may have found the assistance more helpful and became more comfortable requesting help. Prior research has shown students are more likely to request help when they received help that they perceive to be more suitable for their needs [Pri17a].

Although WPs were designed to promote more independent, strategic problem solving, it is possible that NSs also helped students learn problem solving strategies. Based on the hint generator design, NS students following the hints were seeing the most efficient next step based on the current proof state. Problems with frequent Next-Step hints could be acting as partially-worked examples, which are known to increase efficient problem solving strategies [Swe85; McL14]. Previous research on hint usage during problem solving in programming suggests that hints can, sometimes, save students time but reduce learning [Mor15]. In my research, NS hints seem to save students time and increase performance on the post-test. This suggests that NS hints may help students learn to solve the problems more efficiently (more quickly and with fewer steps).

3.8 Contribution

This paper contributes a study showing an extension of the Hint Factory to create higher-level hints, and the effects of two types of hints on students’ efficiency and accuracy in solving logic proofs. Next-Step hints (NSs) helped students become quicker, more accurate, and more efficient in their proofs. However, the more distant goals of Waypoints (WPs) seemed to be harder for the
students, which not only affected the training problems where the assistance occurred, but resulted in lower accuracy and reduced efficiency in the posttest. Despite the WP group spending more time on problem solving during training, their performance did not benefit as much as the NS group. Furthermore, learners with lower incoming proficiency were least able to utilize WPs, while NSs provided benefits to both higher and lower proficiency groups. Although NS performed better overall, students who were able to incorporate WP, especially those in the WP-High group, saw benefits in terms of time and efficiency on the posttest. Another interesting outcome was that the NS group had higher justification rates and requested more help, which agrees with previous research showing that hint quality affects help-seeking behaviors. In the future, WPs could be augmented to reduce cognitive load without eliminating the multi-step aspect by eliminating other elements of the task, such as highlighting needed nodes or offering multiple hint levels. Finally, I hope to transfer these findings to other open-ended problem domains like programming in order to offer additional instructional supports and hints to novice students.
CHAPTER

4

STUDY 2 - AUTONOMY

This chapter focuses on Study 2: “Does autonomy help? The impact of unsolicited hints and choice on help avoidance and learning.” This study compares test conditions with varying levels of student autonomy regarding tutor-initiated, proactive assistance.

4.1 Abstract

Some research has shown that unsolicited hints can improve student learning while other research suggests that on-demand hints are more beneficial. In this study, I compare three types of student autonomy regarding hints: 1) Choice, with periodic popups asking whether the student would like a hint, 2) Control, with on-demand hints, and 3) Assertions, with periodic unsolicited hints. I found that the Control and Assertion groups performed similarly, and significantly better on the post-test than Choice. Further, the Assertions group had the fewest steps where help was needed but was not received, effectively solving the help avoidance problem. The Choice group developed strong preferences for accepting or refusing hints, with those refusing hints performing the worst on the post-test. Overall, our results suggest that unsolicited hints can effectively ensure that more help is delivered when it is needed, reducing autonomy without reducing learning. Further, autonomy was detrimental to students in the Choice group who refused hints. Our results suggest that adaptive policies may increase learning by providing unsolicited help when it is needed, and allowing autonomy otherwise.
4.2 Introduction

Intelligent Tutoring Systems (ITSs) have the unique ability to provide adaptive assistance to support student learning. Although research has shown that allowing students to have autonomy while learning a new domain can benefit learning [Bur79; Sch89; Shu88; Sch87; Kat07], studies have shown that students many not have the required skills to self-regulate their learning to seek help appropriately [Ale06; Pri17a; Zho16]. Research has explored ways of supporting self-regulated learning [Peñ11; Aze04; Ale06]. Additionally, research has found that providing students with meta-cognitive training increased students’ learning [Aze04]. One review compiled research showing that increasing metacognition skills increase opportunities to accomplish educational goals and introduced a paradigm to inspire the design of tutoring systems [Peñ11]. However, ITS students often attempt to ‘game the system’, to receive help when it was not needed in an attempt to avoid work [Bak04; Bak07].

One concern with unsolicited assistance is whether this support limits student autonomy in deciding whether they need help. Research has shown that students often cannot make effective decisions regarding when they need a hint [Zho16]. One specific example of students lacking help-seeking abilities can be seen in work that has found that students often partake in help avoidance, where students do not use assistance available within a tutoring system [Ale04; RAN14]. To address help avoidance, some ITSs employ proactive assistance to prevent help avoidance [Van06]. While one paper found that on-demand assistance, i.e. full autonomy, produced better learning outcomes [Raz10], other studies have shown that providing tutor-initiated, unsolicited hints at the appropriate time, i.e. with no student autonomy about hints, can augment students’ learning experience and improve performance [Bun04; Pud98; Bar06]. Having the tutor proactively intervene while students are solving a problem has the potential to counteract help avoidance and provide needed help to students. This may be especially true for students with lower self-regulating skills who may also have lower incoming proficiencies. Therefore, addressing help-avoidance may particularly benefit such students.

The goal of this work is to investigate whether unsolicited Assertion hints, provided using a random-yet-reasonable periodic policy, can solve the help avoidance problem. I compare three conditions: 1) Choice, where students were periodically asked whether they would like a hint, 2) Control, with on-demand hints, and 3) Assertions, where unsolicited hints were periodically added to the student’s workspace without any element of student choice. Overall, I hypothesize that the benefit of receiving help when it is needed outweighs the negative impact of removing student autonomy about when and whether to receive a hint. Specifically, I hypothesize that:

• $H_1$. Assertions can increase the chance that students will receive help when it is needed, while not harming performance.

• $H_2$. The Choice group will demonstrate more help avoidance than Control and Assertions, and this will negatively impact their performance in the posttest.
• $H_3$ The Control group will take longer in the training than the Assertions group and have more instances of needing help but not receiving it, but have similar performance in the posttest.

### 4.3 Background

Student autonomy over the learning domain along with appropriate guidance is argued to be crucial to learning [Bur79; Sch89; Shu88; Sch87]. Indeed, much of the prior work has shown that it is desirable for students to experience a sense of control (i.e. autonomy) over their learning, which could enhance their motivation and engagement [Cor96; Kin89] and improve their learning experience [Sch98; Yeh01]. On the other hand, students often do not have good self-regulated learning skills as evidenced by research to better support them [Peñ11; Aze04]. Further, students frequently demonstrate poor help-seeking behavior, failing to ask for help when it is needed (help avoidance) or requesting help when the student could solve a problem without it (help abuse) [Ale06; Pri17a].

Despite the known benefits of assistance and researchers’ attempts to provide students with supportive technologies, students often display help avoidance patterns, where students do not leverage the intelligence within the system for help [Ale04]. There are many reasons for help avoidance, one of which is that certain students may lack specific meta-cognitive or self-regulation skills like knowing when to ask for help [Ale00]. A study by Zhou et al. found that students were more likely to make effective pedagogical decisions at the problem-level rather than the step-level, showing that students were less able to effectively decide when they needed a hint on a particular problem-solving step [Zho16]. In one study, researchers found that a large number of students using Andes, the physics tutor, would guess instead asking for hints [RAN14]. A study comparing unsolicited hints or meta-hints (asking the student to ask for a hint) suggested that unsolicited hints were more effective than on-demand hints [Ran11].

Some ITSs prevent help avoidance by proactively providing unsolicited hints when they are needed [Van06]. While Razzaq et al. [Raz10] found that students learned more reliably with hints on-demand than unsolicited hints, other studies have shown that providing hints at the appropriate time can augment students’ learning experience [Bun04; Puu98] and improve their performance [Bar06]. Furthermore, Arroyo et al. [Arr01] observed higher learning gains for low performing students when unsolicited hints were provided. Another study by Murray et al. found that proactive hints avoided the negative effects of frustration and saved students time by preventing unproductive struggle [Mur06].

One important factor of effective assistance is designing it to minimize cognitive load. Research by VanLehn et al. focused on tutoring additional concepts in Andes, a physics tutor, by adding additional support that was minimally invasive to the student’s already-complex learning environment [Van02]. Cognitive load is affected by both the difficulty of the task, and their own proficiency with the topic, so adding support into a system should ensure that the addition doesn’t impose additional cognitive load-bearing activities to the student [Swe88]. In our prior work, I developed a new inter-
face for providing unsolicited hints, called Assertions, designed to minimize the extraneous tasks students would need to do to follow the hints. A study by Maniktala et al. found that the Assertion interface design reduced help avoidance and fostered productive persistence among students with low prior knowledge when compared with text message-based hints, both generated using the same technique.[MM20b]

This study seeks to determine how hints all using the same easy-to-use interface impact hint avoidance and performance when they are on-demand, unsolicited, or provided after offering a choice.

4.4 Deep Thought, a logic proof tutor

Our propositional logic tutor [Mos17] allows students to solve proofs using a graphical interface (see Figure 4.1). Students construct logic proofs in the workspace. The tutor presents proof problems as a set of given logic statements, shown at the top of the workspace, and a conclusion to be derived, marked with a question mark at the bottom of the workspace. Each logic statement, is represented by a node on the screen. Students solve problems by iteratively deriving new logic statement nodes until they derive and justify the conclusion. To create a new statement node, students first ‘justify’ it by selecting 1-2 existing nodes and a rule to apply to them, and ‘derive’ it by typing in the resulting statement. Note that justifying a statement means referring to the previously-justified nodes and domain rule used to derive it. Throughout the tutor, including the pre- and post-test problems, our logic proof tutor provides immediate error feedback for rule application mistakes. If a student clicks on the wrong rule, or their derived statement does not follow from the selected nodes and rule, the tutor provides immediate error feedback and records the error. For example, if a student selects two nodes and then clicks on the Simp rule, the error prompt reads “Rule requires one premise,” then fades away. If the student enters a derived statement that is true, and the justification (consisting of the selected nodes and rule to derive it) is correct, then a new node with the derived statement appears in the workspace. To complete a problem, the student must iteratively derive and justify new statements, until the conclusion statement is derived and justified. When students have completed a problem, the conclusion’s question mark is removed, and it is visually connected to the givens through a series of derived nodes and arrows indicating their justifications. Since the tutor includes a truth verification system, a rule application checker, and tracks the status of the conclusion node, it is straightforward to provide error feedback and detect problem completion. A student’s final solution cannot be wrong. The system logs every click-based action in every portion of the tutor.

Students request hints using the “Get Suggestion” button below the workspace. The tutor logs every click-based action, including: the time between each click, the logic rule applied and prior nodes to justify new nodes, and on-demand hint requests. The tutor is divided into introduction, pretest, training, and posttest. The introduction includes two worked examples where students click through the derivation and justification of all the nodes, followed by one practice problem to
learn the interface. Next, a student takes the pretest problem, which I use to compare the student’s incoming proficiency for stratified sampling. The pretest score is a combination of normalized metrics for the pretest time, number of steps, and accuracy, ranking students based on how fast, efficient, and accurate they are compared to their peers. Next, the tutor guides students through 5 training levels with increasing difficulty. Students solve 15 problems in this training section, where students receive can receive assistance and can also skip or restart problems. Finally, students take a more difficult non-isomorphic posttest, where all students must solve the same set of 4 problems without any tutor assistance, and the posttest score uses the same metrics as the pretest.

Figure 4.1 The logic tutor, showing the workspace on the left with Givens (1,2,3,4) at the top and the Conclusion (Q → R) at the bottom. Rules are in the middle, and the Info box is to the right, where problem information and notices to attempt to justify Assertions appear.

4.4.0.1 Assistance

The tutor uses a data-driven approach to generate assistance from historical student data consisting of prior student solution paths from the problem start to the goal through a series of problem-solving states. This algorithm results in assistance based on the most frequent and efficient paths available based on the student’s current attempt.

Assistance provided by the tutor in the form of hints can either be initiated by the student, in which case they are called on-demand hints, or they can be initiated by the tutor, in which case they are called unsolicited hints. In this work, I used our recently-designed Assertions interface to place all next-step hints, which consist of a suggested next statement to derive, as nodes marked with a question mark and a ‘Goal’ label (to denote that they have not been justified) in the workspace.
as shown in Figure 4.2. To use a hint in their proof, students must first correctly justify the hint by indicating which prior nodes and rule can be used to derive the node’s statement. In training problems, all students may request hints in addition to those provided or offered by the tutor according to their study condition. Pretest and posttest problems disallow any hints. Hints do not tell students which rules or prior nodes can be used to justify the suggested statement. Rather, our hints are designed to help students solve problems by suggesting a subgoal statement that helps them break down multi-step problems into smaller chunks. All hints are next-step, which suggest the next, best statement that can be derived in one rule-application step from the student’s current state. For example, one tutor hint could add an $F$ node as a subgoal, and the student would select $I \land F$ and choose the “Simplification” rule to justify it (See Figure 4.2).

**Figure 4.2** A hint statement being justified. (1) Shows a hint appearing in the workspace. (2) Two statements are selected (highlighted in blue) and the rule “Modus Ponens” applied. (3) The hint has been justified and is connected to the student proof.

### 4.5 Methods

The tutor was used as a mandatory, online homework assignment by students in an undergraduate discrete mathematics for computer scientists course at a large public university in the United States in the Spring 2019 semester. For this study, 94 students’ data from three conditions were compared to investigate the impact of student-choice on performance and behavior. While all conditions allowed on-demand hints, they differed slightly in unsolicited help. The three conditions included: 1) **Control**, 2) **Choice**, and 3) **Assertions** groups.

The Assertions group received periodic unsolicited hints on approximately 40-50% of the steps to produce assistance similar to a partially worked example, or turn-taking tutor where the tutor and the student co-construct a solution to the problem. The Choice group was asked “Would you like a suggestion?” after completing approximately every third step, with a max of 3 questions per problem. The student had to choose either “Yes” or “No” to proceed. I chose this amount so that it would
be frequent enough to be comparable to the A group, but not frequent enough to be distracting. The Control group was prompted via a message in the box below the workspace (See Figure 4.1) to request a hint if they spent longer than the 75th percentile of historical step time data on that step for that problem. The message prompt was “Feeling stuck or tired? Click the “Get Suggestion” button!” In this group, students had the autonomy to refuse or accept these hint prompts, and hints were only given when students answered yes.

The Assertions group provides students with the least amount of autonomy regarding when to receive a hint, by adding unsolicited Assertion hints to the workspace. Students may ignore these hints, but since they are the most efficient next step, students avoiding them will have less efficient solutions. The Choice group is the middle ground for hint autonomy because students can choose not to receive a hint by clicking “No” on the pop-up. However, due to the need to respond to the question and make a help-seeking decision, this group is considered to have a medium level of hint autonomy. The Control group is considered the most autonomous because they control the entire interaction surrounding hints – they only receive hints when they press the Get Suggestion button.

Stratified sampling was used, splitting students by performance on the pretest, then randomly assigning them to Assertions (n = 38), Choice (n = 27), and Control (n = 29) to ensure all conditions were balanced in incoming knowledge. The Assertions group was designed to have a slightly larger size, to ensure sufficient data collection for the study, and since I felt that this condition would be more beneficial to students than the Choice or Control conditions.

### 4.5.1 Performance Metrics

The tutor begins with an introductory tutorial consisting of two worked examples and one practice problem followed by the pretest. I used each student’s pretest score to measure incoming knowledge (See Equation 4.1). A student’s score is a combination of normalized metrics for the pretest time, number of steps, and accuracy on a single problem, which ranks a student based on how fast, efficient, and accurate they are compared to their current peers (descriptions of each metric discussed below). Note that the tutor is not designed or assumed to promote large improvements in accuracy, since no penalties are assigned for incorrect rule applications and the tutor simply alerts students upon wrong rule applications and students may try again, even within the pre- and post-tests. Further, problems require new rules and become more difficult as the students progress. As we seek primarily to promote more efficient problem solving, we focus more on steps and time per problem while maintaining reasonable accuracy. This is because it is more difficult for students to learn to determine which steps to derive to achieve shorter, more efficient proofs, compared to learning how to apply the rules, which can be done by memorization and simple practice. Deep Thought is built primarily to allow students to practice with the strategy of problem solving, rather than fluency with rules, most of which are assumed to be learned before the tutor. Furthermore, the time and steps are correlated, i.e. more steps generally result in a higher time. However, including both of these features provides a better estimate of the student’s performance, e.g. a student with a higher amount of steps can potentially complete problems in a shorter amount of time than a student with fewer steps.
Therefore, both time and step features are essential to capture the true performance of each student.

\[
\text{Pretest Score} = (1-\text{TotalTime}) \times 0.5 + (1-\text{TotalSteps}) \times 0.3 + \text{Accuracy} \times 0.2
\]  

(4.1)

The posttest is the average performance on four unassisted problems at the end of the tutor to measure post-training performance. I investigated pre- to posttest changes as well as performance impacts on time spent solving a problem, total attempted steps, and accuracy. Total time is counted from the moment a problem is on the screen until it is solved by deriving and justifying the conclusion. Total steps in a problem include any attempt a student makes at deriving a new step, which includes both correct and incorrect steps (node derivations). I also calculate the hint justification rate, which represents the percentage of hints received that students connected to their current solution through justification. I also investigated impacts on help-seeking through the total number of Hint Requests over the training portion of the tutor. Hints Received are defined as the total hints a student received during the tutor, whether through unsolicited hints or hint requests. I also have a metric specifically for the Choice group to represent how many times they were asked whether they would want a hint (i.e. # Asked).

An important goal of this study was to investigate whether periodic unsolicited hints could address help avoidance by increasing the number of times students who needed help received it. I further felt that, since our hints are partially-worked steps and not full bottom-out hints, and students could easily ignore them, unsolicited hints would not harm students who did not need them. Therefore, I determined when a hint was needed vs. not needed via a help-need model described in \[\text{MM20a}\]. The model uses (1) the quality of the current step based on a combined productivity measure of the optimality of their current state (how close it is to the solution based on the Hint Factory \[\text{Sta08}\]), and the time taken to derive it, and (2) a prediction of whether help is needed in the next step (e.g. if the next step is not predicted to be productive, then help is needed). I use the current state's help-need measure to check for help avoidance, since the quality of the current state lets us determine whether a hint would have been appropriate. I use predicted help-need to check for help abuse, to see whether students requested hints when the help-need model predicts that they could have been productive without help. I note that our help-need predictor is not ground truth, but a data-driven student model based on historical student data and the domain-based principle that shorter solutions are better. While this is not ground truth, our cited work shows that the Help-Need predictor is correlated with post-test performance. Since learning and post-test performance are the tutor goals, a predictor correlated with these seems is a reasonable and fair comparison across study conditions.

To understand the Choice group's help-seeking behavior, I analyzed the patterns of decisions (yes/no choices) using a procedure used in other educational domains to measure the consistency and pattern of student choices. Specifically, I followed the procedure in Snow et. al that uses Random Walks, Shannon Entropy \[\text{Sha51}\], and Hurst Exponent \[\text{Hur51}\] analyses to quantify students decisions in terms of consistency and deterministic behaviors \[\text{Sno16}\].

Data were analyzed to compare groups for the pretest, training, and posttest portions of the
tutor. Within each group, I also compared the performance of students with high or low initial proficiency, based on a median split on pretest score. ANOVA with Tukey's post hoc tests were used to examine the significance of differences in the means of the populations between pretest groups. For training and posttest metrics, I applied one way ANCOVA using the pretest as a covariate. I used Benjamini-Hochberg corrections to account for multiple tests. To check that the data met assumptions for ANOVA and ANCOVA, I used the the Shapiro-Wilk's W test and Levene's test, as well as visually inspecting the data via Q-Q plots and histograms. Data that did not meet the assumptions were transformed using log or square-root transformations, then re-inspected. Data reported in tables are before transformation for clarity.

4.6 Results & Discussion

This section discusses the comparison between the Assertion, Choice, and Control groups, and the differences in performance between students with the High and Low incoming proficiency. Lastly, the Choice group is examined in finer detail to evaluate how student's choices affected their overall performance.

4.6.1 Hint Usage and Help Need

To understand each group's exposure and utilization of hints, I examined the overall hint-related metrics (# Hints Requested, # Hints Received, # Hint Requests, # Asked, and hint Justification rate) and applied ANOVA for between group comparisons. Table 4.1 shows the mean, standard deviation and Tukey HSD's results between the Assertions, Choice, and Control groups for the hint metrics. Note that total hints includes both on-demand hints and unsolicited hints (for the Assertions group). Using ANOVA, I found a significant difference in the mean # Hints Received ($F(2, 91) = 25.576, p < 0.01$) between the groups. Tukey Contrasts analysis showed significant differences among each comparison (Control-Choice ($p < 0.01$); Choice-Assertions ($p < 0.01$); and Control-Assertions ($p < 0.01$)). I expected these differences because the Assertions group was given frequent, unsolicited hints, the Choice group was asked whether they wanted a hint at a slightly lower frequency, and the Control group received hints only upon request. Since all three groups could request on-demand hints in addition to any the tutor might provide or offer, I compared Total Hints Requested, but there were no significant differences between the 3 groups on this metric ($F(2, 91) = 0.1816, p = 0.83$). Note that the Total Hints are not the same as the Hints Requested because Hints Requested only counts how many requests a student made, and some students may have requested for new hints when the most optimal hint was already present in the workspace.

It is natural to assume that students would be more likely to use the hints they request, and at least one study has confirmed this intuition [Raz10]. The Control group justified 85% of the hints they requested, on average, and this was expected since they actively requested hints. The mean Hint Justification rates were 84% for Assertions and 80% for the Choice group. ANOVA results revealed a significant difference between groups for the Hint Justification Rate ($F(2, 91) = 6.0633, p < 0.01$).
Table 4.1 Mean and Standard Deviation (SD) of the Hint Usage Metrics over the Training portion of the tutor for the Control, Choice, and Assertions groups.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Control (n = 29)</th>
<th>Choice (n = 27)</th>
<th>Assertions (n = 38)</th>
</tr>
</thead>
<tbody>
<tr>
<td># Asked</td>
<td>34 (10)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td># Hints Received</td>
<td>19 (16)*</td>
<td>35 (25)*</td>
<td>51 (12)*</td>
</tr>
<tr>
<td># Hint Requests</td>
<td>21 (21)</td>
<td>26 (27)</td>
<td>25 (23)</td>
</tr>
<tr>
<td>Hint Justification Rate</td>
<td>85% (25)</td>
<td>80% (20)*</td>
<td>84% (6.5)*</td>
</tr>
</tbody>
</table>

Tukey Contrasts analysis showed significant differences among Control-Choice (p = 0.03), and Control-Assertions (p < 0.01). However, there was not a significant difference between Choice and Assertions group (p = 0.79). This is somewhat surprising because I expected the Choice group to have a higher Justification rate than the Assertions group, since, like the Control, they chose to get a hint, while the Assertions group received the least autonomous treatment. These results suggest that unsolicited Assertions were just as well received as hints offered as a choice.

To examine how each of these methods of providing hints affected hint usage, I looked at student’s help need, hint abuse, unnecessary hints, and times in which they received an appropriate level of help (i.e. received a hint when needed and did not receive a hint when not needed). These measures were defined to address all three hypotheses concerning hint usage and based on our model of help need described in Section 4.5.1. Our definition of help need measures when a student needed a hint, but did not receive one (either unsolicited or requested). % Help Needed is the percentage of total steps our Help-Need model identified as unproductive, where a student could have benefited from a hint, and a hint was not received % Hint Abuse is the percent of total steps where our model predicted no Help-Need but a student requested a hint, representing bad help-seeking decisions. % Unnecessary Hint is the percent of total steps where students received a hint on a step where I predicted no Help-Need. Unnecessary hints includes both hint abuse requests and the number of times Assertions were given but not needed, as a measure of how frequently students in each condition received unneeded help. I also included Hint Abuse because I wanted to ensure none of the conditions were promoting gaming the system. % Appropriate Hint is the percent of steps where Help-Need model aligned with the student need (e.g. a student received a hint when they were predicted to need one or a student did not receive a hint and the model labelled the step as no help-need).

I determined when a hint was needed or not via the help-need model described in [Ano19] and discussed in Section 4 (Methods). Table 4.2 shows the differences in these metrics between the groups. Using ANCOVA, controlling for the pretest, I found a significant difference between the groups for % Unnecessary Hints (F(2, 91) = 38.35, p < 0.01) and % Help Needed (F(2, 91) = 10.11, p < 0.01). For % Unnecessary Hints, Tukey Contrasts analysis revealed significant differences between all 3 groups: Choice-Control (p = 0.01), Choice-Assertions (p < 0.01), and Control-Assertions (p < 0.01). For % Help Needed using the same procedure, I found significant differences between
Choice-Assertions ($p = 0.01$) and Control-Assertions ($p < 0.01$); however, there was no significant difference between Control-Choice ($p = 0.45$). There were no significant differences for Hint Abuse ($F(2, 91) = 0.04, p < 0.96$) or the Appropriate Hint metrics ($F(2, 91) = 0.57, p < 0.56$).

The Control group had the lowest percentage of steps with Unnecessary hints, which was expected because they had full autonomy and requested fewer hints than the other 2 groups. The Control group also had the highest percentage of steps where Help-Need was detected, meaning that these students spent more time in steps being unproductive (as defined in the model). The Choice group fell in the middle for both Percent Help Needed and Percent Unnecessary Hints.

Hypothesis $H_2$, stated that the Choice group would have more help avoidance than the other two groups. The Control group showed similar help avoidance to the Choice group by not requesting hints when needed. However, the Choice group had a significantly higher Help Avoidance than the Assertion group, which provides partial evidence in support of $H_2$.

Additionally, the Control group having a significantly higher percentage of steps labelled Help Needed partially supports $H_3$, in which I hypothesized that the Control group would not request hints often enough. The results also indicate that the Assertions group decreased steps where students needed help but were not receiving it but increased instances where the student did not need a hint but received it, confirming $H_1$. My goal was to reduce students being stuck in steps without receiving help because low performing students are often the ones in need and could benefit the most [Arr01], which was achieved even though the frequency of unsolicited hints was not based on an intelligent policy. Incorporating an intelligent policy to determine when to give a hint should result in an even smaller percentage of help need and reduce instances of unnecessary hints. To test whether the larger percentages of Unnecessary Hints would be worse for posttest performance, a simple linear regression was calculated to predict the posttest score based on the % Unnecessary Hints and was not significant ($F(1, 91) = 0.33, p = 0.57$). Therefore, I do not believe these Unnecessary Hints had a significant impact on performance. Another simple linear regression was calculated to predict the posttest score based on the % Help Needed, and a significant regression was found ($F(1, 91) = 8.49, p < 0.01$) providing support that addressing help need is important.

### Table 4.2

<table>
<thead>
<tr>
<th>Metric</th>
<th>Control $n = 29$</th>
<th>Choice $n = 27$</th>
<th>Assertions $n = 38$</th>
</tr>
</thead>
<tbody>
<tr>
<td>% Help needed</td>
<td>20(12)</td>
<td>16(11)</td>
<td>10(8)*</td>
</tr>
<tr>
<td>% Hint Unnecessary</td>
<td>4(5)*</td>
<td>7(5)*</td>
<td>15(4)*</td>
</tr>
<tr>
<td>% Help abuse</td>
<td>7(6)</td>
<td>9(9)</td>
<td>7(7)</td>
</tr>
<tr>
<td>% Appropriate Hint</td>
<td>72(11)</td>
<td>71(12)</td>
<td>73(7)</td>
</tr>
</tbody>
</table>
4.6.2 Evaluating Students’ Performance Across the Tutor

To examine the effects on performance each group had, the pretest and posttest performance metrics for the 3 groups were analyzed (see Table 4.3). These analyses were conducted in order to address the second portion of all three hypotheses concerning how hint usage might have an affect on performance. ANOVA was performed on pretest metrics to determine if there was a similar distribution of proficiency between the groups. There were no significant differences between the groups on Total Time ($F(2, 91) = 0.28, p = 0.76$) or Total Steps ($F(2, 91) = 1.01, p = 0.37$) in the pretest metrics; however, the Choice group performed the fastest with the fewest steps. There was a marginally significant difference between the groups for accuracy ($F(2, 91) = 2.38, p = 0.09$), but this is not a meaningful difference due to the few number of steps in the pretest and the Choice’s group lower average number of steps – meaning that each mistake would have contributed more to the percentage. Therefore, I concluded that each group had a distribution of students’ with similar incoming proficiency. For the training and posttest performance metrics, ANCOVA was used controlling for pretest metrics. There were no significant differences between any performance metric in the training portion of the tutor (Total Time ($F(2, 90) = 2.07, p = 0.13$); Total Steps ($F(2, 90) = 1.84, p = 0.16$); Accuracy ($F(2, 90) = 1.34, p = 0.27$)). The posttest metrics show a significant difference in the Total Time ($F(2, 90) = 5.24, p < 0.01$) between the groups. Tukey Contrast analysis revealed that there was a significant difference between the Assertion and Choice group ($p < 0.01$); however, there was not a significant difference between the Choice and Control ($p = 0.29$) or the Assertion and Control ($p = 0.19$). There was no significant difference between the Total Steps($F(2, 90) = 2.09, p = 0.13$) or the Accuracy ($F(2, 90) = 0.05, p = 0.95$) between the groups. These results provide support for $H_3$ that the students in the Assertions group would perform similarly to the Control group; however, the Control group did not perform worse in the training as expected. These results along with the results in 4.2 confirm $H_1$. Assertions reduce help need without harming performance. While the Choice group was the fastest in the pretest, they were the slowest in the posttest. On the other hand, while the Assertions group was the slowest in the pretest, they were the fastest in the posttest. Students in the Choice group attempted more steps than both of the other two conditions, although not significantly. These results provide evidence in support of $H_2$; however, these results do not address whether the worse performance was due High or Low groups in particular.

To see if incoming proficiency had an effect on which condition was most helpful, I examined the differences between the High and Low proficiency students based on a median split of the pretest score and using ANCOVA with pretest metrics as the covariate. There were no significant differences in the means of the pretest, training, or posttest metrics between any of the groups. However, the Choice group performed worse on the posttest than the both the Control and Assertions group in both time and steps for the High and Low proficiency groups, although the differences are not significant. Both the Choice-High and Choice-Low performed similar to each other, in terms of Total Time (Choice-High Mean(SD) : 44(37); Choice-Low Mean(SD) : 41(28)), Total Steps (Choice-High Mean(SD) : 130(33); Choice-Low Mean(SD) : 124(73)), and Accuracy (Choice-High Mean(SD) : 69%(12); Choice-Low Mean(SD) : 68%(10)). This result was peculiar as I expected that
Table 4.3 Pretest, Training and Posttest performance metrics for the Assertion, Choice, and Control groups. ANOVA for the pretest and ANCOVA using pretest metrics as the covariate was used to determine significance between group means. P-values are reported in text and bold in table.

<table>
<thead>
<tr>
<th>Test</th>
<th>Metric</th>
<th>Control n=29</th>
<th>Choice n=27</th>
<th>Assertion n=38</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pretest</td>
<td>Total Time (min)</td>
<td>5.8(7)</td>
<td>4.0(2)</td>
<td>6.5(6)</td>
</tr>
<tr>
<td></td>
<td>Total Steps</td>
<td>15(30)</td>
<td>9(7)</td>
<td>11(13)</td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>40(14)</td>
<td>35(14)</td>
<td>43(14)</td>
</tr>
<tr>
<td>Training</td>
<td>Total Time (min)</td>
<td>137(50)</td>
<td>114(49)</td>
<td>122(62)</td>
</tr>
<tr>
<td></td>
<td>Total Steps</td>
<td>374(126)</td>
<td>348(124)</td>
<td>323(118)</td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>63%(12)</td>
<td>66%(11)</td>
<td>66%(10)</td>
</tr>
<tr>
<td>Posttest</td>
<td>Total Time (min)</td>
<td>37(29)</td>
<td>43(34)*</td>
<td>34(20)*</td>
</tr>
<tr>
<td></td>
<td>Total Steps</td>
<td>104(56)</td>
<td>129(75)</td>
<td>102(47)</td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>69%(12)</td>
<td>69%(11)</td>
<td>69%(11)</td>
</tr>
</tbody>
</table>

proficiency would be playing a larger role in the results of the Choice group. This result provides evidence in support of $H_2$. One limitation to this analysis is by splitting the 3 main groups into subgroups, the sample size decreases substantially.

4.6.3 Did the Choice group make good help-seeking decisions?

To try to understand why the Choice group made their decisions, I evaluated their choices from multiple approaches. First, I observed that students seemed to be polarized to either accept or reject the choice to receive a hint, having a bimodal distribution when looking at looking at the percentage of times a student chose ‘Yes’ compared to their total choices (i.e. percentageYes). Therefore, I looked at the decisions Choice group students made when asked whether they would like a hint. By choosing yes or no, students are deciding whether they want a hint at that given time — but I do not know why each student made these decisions. I hypothesized that there may be a personal preference for these decisions. So, I followed a procedure by Snow et. al that uses Random Walks, Shannon Entropy [Sha51], and Hurst Exponent [Hur51] analyses to quantify students decisions in terms of consistency and deterministic behaviors [Sno16]. Similar research has also used these procedures to measure students' behaviors in tutoring systems[SP15; Sno15]. Random walks create a visualization of student's decisions with each trajectory beginning at the origin (0,0) and moving along the x- and y-axis depending on student's decisions. I generated my Random Walks by plotting each student's decision (yes +1 x-axis and no + 1 y-axis). Figure 4.3 below shows the Random Walks for each student colored by their Pre- and Posttest groups.

In Figure 4.3, the plots show the Low and High pretest groups plotted with the lines colored based on their posttest performance. This visualization demonstrates the often polarized choices the students were making, especially the High pretest students. One interesting result not obvious
Figure 4.3 Random walks of the Low and High Choice groups are labelled. In each plot, lines are colored red for Low-posttest and blue for High-posttest. Low/High posttest splits were determined via median split of the pretest score among all students in the study.

from the plot is that the red line in the Low pretest plot is actually 4 students plots. These 4 students all started off with a Low pretest, answered yes to whether they would like a hint each time, and ended up in the Low posttest group.

To quantify regularity of these decisions, Entropy and Hurst are among two of the most commonly used [Sno16; Sno15]. Entropy scores are created by trying to represent bits of information presented within a time series. For example, flipping a fair coin has a .5 probability. If I flip the coin twice and get heads both times, the Entropy would be 0 (i.e. uniform bits of information). Whereas, if I flip the coin and get one heads and one tails, the Entropy would be 1.0. Note, if you had more than two choices, the Shannon Entropy value could be greater than 1. Low entropy values indicate that the behavior pattern is an ordered, controlled process, whereas high entropy suggests a random process. Shannon entropy was used on the bins of student’s yes/no decisions. The entropy of the student decisions was relatively low \( n = 27, \text{Mean}(SD) : 0.49(0.34) \) indicating that student's made consistent decisions.

Although Entropy analyses can provide a measure of how ordered students' choices are, this measure does not capture how each choice within the pattern relates to the other choices within the time series. The Hurst exponent has the ability to capture the predictability of the time series. The Hurst exponent divides actions into equal time windows and considers how each choice within the pattern relates to the other choices before and after it. Hurst values between 0.5 and 1 \( (0.5 < H <= 1) \) indicate deterministic behaviors; \( H=0.5 \) indicates random behaviors, and lower Hurst values with \( 0 <= H < 0.5 \) indicate anti-persistent behavior trends. The Hurst exponent was calculated for each student’s random walk trajectory. The Hurst Exponent results exhibited highly deterministic
behaviors \((n = 27, Mean(SD) : 0.76(0.02))\). This suggests that students do display strong personal preferences, and that they maintain these preferences throughout the tutor.

To understand why the Choice group performed worse despite similar training performances to the Assertions and Control groups and with little difference between High and Low proficiency subgroups, I further investigated how student choices may have affected performance in the Choice group.

Using Hierarchical clustering with the Ward’s method on percentageYes, I saw that the best fitting clusters were formed with clusters of 2 compared to higher numbers of clusters based on both intra- and inter-cluster similarity (Silhouette index(0.54 and Calinski-Harabasz index(44.02) as well as Davies-Bouldin Index which uses only intra-cluster similarity (0.66). These 2 clusters form the Yes group \((n = 15, Mean(SD): 85\% (14))\) and the No group \((n = 12, Mean(SD): 15\% (14))\).

Table 4.5 shows the performance and hint metrics for the pretest, training, and posttest for the Yes and No groups. There is no significant difference in the means of the pretest results for Total Time \((F(1, 24) = 2.34, p = 0.14)\), Total Steps \((F(1, 24) = 2.7, p = 0.11)\), and Accuracy \((F(1, 24) = 0.52, p = 0.48)\) between the two groups.

Using ANCOVA, controlling for pretest metrics, I compared the Yes and No groups performance. For the training portion, there are significant differences between the hint Justification rate \((F(1, 24) = 7.86, p = 0.01)\) and the Hints Requested \((F(1, 24) = 33.0216, p < 0.01)\), but no significant differences in the means of the Total Time \((F(1, 24) = 0.17, p = 0.68)\), Total Steps \((F(1, 24) = 0.31, p = 0.58)\), and Accuracy \((F(1, 24) = 0.01, p = 0.91)\). For the hint Justification rate (justifying hints that are already on the screen) reported in Table 4.1, the Choice surprisingly has one of the lower hint Justification rate among the groups, as I would have expected it to be similar to the Control group. Looking at Table 4.4, I can see that the hint Justification rate for the No group is much lower than the Yes group, most likely contributing to the overall lower Justification rate for the Choice group. The No group requested and accepted far fewer hints. One possible reason for the No group having a low hint Justification rate is that they either saw hints that they did not find helpful or decided that they did not want to use the hints, resulting in their individual hint Justification being low, especially if they later decided not to request hints lowering the total number of hints they received (e.g. 1 unused hint out of 5 total hints has a larger impact on the hint Justification rate compared to 1 unused hint out of 20 total hints).

For the posttest, there are no significant differences in the means of Total Time \((F(1, 24) = 0.52, p = 0.47)\) or Total Steps \((F(1, 24) = 0.41, p = 0.52)\), or \((F(1, 24) = 0.15, p = 0.69)\). However, the average Total Time and Total Steps are higher for the No groups.

Next, I investigated the distribution of High and Low proficiency students in the two clusters. Table 4.5 shows the posttest performance score for the High and Low Proficiency students in the Choice group, also split by the Yes and No students. There are no significant differences between the groups, most likely due to the low sample sizes. However, an interesting result is that students in the Choice-High-Yes group and students in the Choice-Low-No group performed the worse out of all 4 Choice subgroups on the posttest. In Table 4.4, I can see that the Choice-No group performed worse
Table 4.4 The Posttest Performance Metrics for the Yes and No groups within the Choice condition.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Pretest</th>
<th>Training</th>
<th>Posttest</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Yes (n=15)</td>
<td>No (n=12)</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Total Time (min)</strong></td>
<td>5.9(2.5)</td>
<td>5.1(1.9)</td>
<td>56(23)</td>
</tr>
<tr>
<td><strong>Total Steps</strong></td>
<td>20(18)</td>
<td>16(14)</td>
<td>189(66)</td>
</tr>
<tr>
<td><strong>Accuracy</strong></td>
<td>34(12)</td>
<td>38(15)</td>
<td>66%(12)</td>
</tr>
<tr>
<td><strong>Hint Acceptance</strong></td>
<td>-</td>
<td>-</td>
<td>90%(6)*</td>
</tr>
<tr>
<td><strong>Hints Requested</strong></td>
<td>-</td>
<td>-</td>
<td>37(32)*</td>
</tr>
</tbody>
</table>

The Posttest Performance Score for the High- Low- Proficiency groups within the Yes and No groups within the Choice condition.

Table 4.5

<table>
<thead>
<tr>
<th>Group</th>
<th>Choice_Low</th>
<th>Choice_High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>0.63 (n = 8)</td>
<td>0.56 (n = 7)</td>
</tr>
<tr>
<td>No</td>
<td>0.60 (n = 6)</td>
<td>0.64 (n = 6)</td>
</tr>
</tbody>
</table>

I have two possible hypotheses for why this occurred with the Choice group.

The first hypothesis is that the questions asking whether or not they would like a hint could have been frustrating or distracting. This distraction could have caused them to lose focus; however, I would have expected the total time in the training to be significantly different in that case. Students in the Choice group may have also been distracted by the question of “Would you like a suggestion?” possibly believing that the tutor is asking because they are not performing well. This question may have inadvertently created an additional task increasing the cognitive load to these learners having to evaluate their own performance to make the decision of whether they wanted a hint [Swe88].

Another theory is that the students are not good at making self-regulating decisions and could have been making bad choices. In Table 4.5, the Choice-Low-No group performed worse than the Choice-Low-Yes group. This provides evidence to suggest that the students were making bad self-regulatory choices potentially due to not requesting hints when needed. Furthermore, this outcome provides evidence to support part of $H_2$. In Table 4.5, the Choice-High-Yes group performs compared to the Yes group, and the Yes group's results now look more similar to the posttest results, although the Total Steps is still worse. An important thing to notice is that there is approximately an even split of High and Low students in the Yes and No groups, this will be discussed further in the next section. By looking at Table 4.5 I can see that the Yes and No group are split almost evenly with High and Low proficiency students, meaning that incoming proficiency did not determine whether students said gravitated towards choosing Yes or No.
worst on the posttest score. This provides further evidence to the theory that bad self-regulation habits are causing poor performance. Both refusing help when it is needed [Ale04] and requesting help when assistance is not necessary [Bak04] can produce suboptimal learning outcomes.

A counterargument to this theory is that I would have expected the Control group to have performed similarly to the No-Low group considering the Control only received hints when they requested them and received fewer hints in total. Further, the Yes-High group should have also been experiencing a similar environment as the Assertions group (i.e. receiving frequent in-workspace hints). Therefore, I believe that due to the combination of the questions being potentially distracting and the effectiveness of a hint being dependent on the student’s proficiency, I observed the poor performances of the Choice group.

Lastly, one of my research questions was to determine whether students were better at self-regulating than a random proactive policy. The Assertions group was the slowest in the pretest, but they were the fastest in the posttest, shown in Table 4.3. The overall hint Justification rate was also high, shown in Table 4.1. Along with the results confirming $H_2$ in the Table 4.2 and Table 4.3, these results suggest that the Assertions group with unsolicited, tutor-initiated hints did no harm to students in terms of learning outcomes compared to the Control group and produced better learning outcomes than the Choice group. Therefore, I believe these results suggest that proactively adding hints at the very least did no harm, but compared to students with poor self-regulating abilities, the Assertions group performed better. These results suggest that a machine-learning based proactive hint policy has the potential to produce even better learning outcomes.

## 4.7 Contribution

In conclusion, this work contributes a study investigating the affects of three groups with varying levels of autonomy of assistance on learning outcomes, metrics to evaluate hint usage and hint avoidance, and an application of consistency and deterministic analysis to evaluate help-seeking behaviors. The three groups from most autonomous to least: 1) Control, where students could request on-demand hints, 2) Choice, where students were periodically asked whether they would like a hint, and 3) Assertions, where hints were periodically added to the student’s workspace without any element of student choice. This study sought to determine whether students’ autonomy over when and how the interface provides hints affects hint utilization and, in turn, overall success. My results show that the Assertion and Control group produce similar learning outcomes; however, the Choice group performed worse on the posttest. Overall, my results suggest that unsolicited hints can effectively ensure that more help is delivered when it is needed, reducing autonomy without reducing learning. These results demonstrate that with an effective, machine-learned proactive hint policy, better learning outcomes are possible.
In this chapter, I discuss Study 3: “An Exploration of Reinforcement Learning to Create Effective Proactive Hint Policies.” This study compares two Reinforcement Learning step-policies deployed in the Fall of 2019 and Spring of 2020 that decide when to give a hint.

In Study 1: Hint Content and Study 2: Autonomy, I investigated what type of data-driven hints produced the best learning outcomes and whether proactively adding hints, called Assertions, into the student workspace impacted student learning, in terms of autonomy regarding assistance or invasiveness of the assistance. These two studies were used to investigate students’ reactions to proactive, tutor-initiated hints and determine what hint type would be most beneficial. In Study 1, I found that the Next-Step hints resulted in the best learning outcomes - reducing time spent and total steps and increasing accuracy of the students. However, I noted that Waypoint hints could help higher proficiency students improve efficiency with additional scaffolding to help them learn to achieve the Waypoint hints. Due to the results of Study 1, I concluded that without more research surrounding Waypoint hints, Next-step hints would be the best choice for a proactive hint policy. In Study 2, my results show that the Assertion and Control group produced similar learning outcomes; however, the Choice group performed worse on the posttest. Overall, my results suggest that unsolicited hints can effectively ensure that more help is delivered when it is needed, reducing autonomy without reducing learning. These results suggest that with an effective, machine-learned proactive hint policy, better learning outcomes are possible.
5.1 Background

This dilemma of choosing an appropriate level of assistance shows that giving or withholding information is a delicate balance with tradeoffs [Koe07]. Furthermore, providing assistance when it is not needed can result in worse performance for some students as shown in the expertise reversal effect, in which providing students information they already know increases their cognitive load [Swe08]. However, students often do not have good self-regulated learning skills as evidenced by research to better support them [Peñ11; Aze04]. Further, students frequently demonstrate poor help-seeking behavior, failing to ask for help when it is needed (help avoidance) or requesting help when the student could solve a problem without it (help abuse) [Ale06; Pri17a]. Proactive hints have been suggested as a potential approach to combat these issues and provide hints to students when it is determined to be helpful [Van06].

Razzaq and Heffernan completed a randomized controlled experiment to compare on-demand hints to proactive hints in the tutoring system, ASSISTment System [Raz10]. They found students who asked for a higher number of hints learned significantly more with hints-on-demand, but for students who asked for a low number of hints there was no significant difference between the two conditions. They suggest that the students who requested a higher number of hints may have good help-seeking behavior; thus, not needing proactive help and possibly causing proactive help to be distracting for them. On the other hand, students who asked for a low number of hints may have been exhibiting help avoidance. The authors conclude that these students may have benefitted from being shown a hint when they needed one. The hints in this system were also shown after students made a mistake, when the student is possibly frustrated, and for students who had been using the system without proactive hints throughout the semester, which could have made these automatic hints an unwelcome surprise. When students enter Deep Thought, this is the first time they are using the system. Furthermore, hints are provided before students would make a mistake by providing hints at the beginning of a step when the system determines that they need one. Instead of remediating mistakes, the goal is to provide help on steps that would produce beneficial outcomes in terms of time spent and total steps. By providing hints in this way, I am hoping to guide students to optimal solutions and reduce their overall frustration by simplifying the learning process.

Policies have been created to explore the feasibility of proactive hints and when to give them [Uen17] and proactive agents are noted as one of the next steps to close the learning gain gap between human tutoring and tutoring systems [Bra11]. In the research by Ueno et al., researchers used a probabilistic model, i.e., item response theory (IRT), that determines when to give a hint based on the predicted success of the learner [Uen17]. However, there is limited research on creating effective proactive hint policies.

In my research, I have chosen to use offline, off-policy Reinforcement Learning (RL). In general, there are two main categories of RL: online and offline. Online learning uses real-time training via interacting with the environment, whereas, offline learning is based on historical data. Online policies are generally used when the state space is clear and interacting in the environment is not
computationally expensive; however, in the educational domain, the state space is quite complex and undefined - with the associated rewards, learning outcomes, only available later in the process. Furthermore, an RL policy can be induced either on-policy, from data that was collected using the same policy, or off-policy, where the data collected may be unrelated or using a different policy. Both online and offline RL approaches have been used for pedagogical policy induction; however, prior research has mostly used an offline approach [Chi11; Igl09; Man14; Raf16; She16; Zho17]. In the work of Chi et. al, researchers apply RL to Cordillera, a Natural Language Tutoring System teaching students introductory college physics. Their results show that by using a rather small set of data for training, the RL-induced strategies improved the effectiveness of Cordillera and improved students’ learning gains. Mandel et al. used offline policy evaluation to create policies for an educational game and deployed the best policies to 2000 real students. Their results show that the learned adaptive policy resulted in higher improvements compared to random and expert baselines. Due to the availability of real student data using Assertions and the computational costs, I chose to take an offline, off-policy approach.

Another challenge to reinforcement learning is creating rewards that accurately capture the task you want the model to complete. Most applications of reinforcement learning use immediate rewards, where the result of the action is immediately known after an action. However, with educational data, the rewards are not only delayed but also noisy. For ITSs, the most appropriate rewards are based on the student’s learning by the end of the tutor due to the complex nature of learning. Therefore, assessing a student’s learning from one step to the next may not be reflective of the student’s overall learning by the end. This complexity presents a problem in reinforcement learning as assigning the credit of the result of the end learning to levels, problems, or steps can be difficult.

Additionally, data collected cannot be guaranteed to be perfect or an entirely accurate and complete reflection of the student’s learning. One potential solution is to infer the immediate rewards from the delayed rewards by intelligently assigning credit backwards from the delayed reward. One process has been created to address this issue by distributing delayed rewards using a Gaussian Process(GP)-based immediate reward approximation algorithm [Azi].

Aptitude-treatment interactions have been widely studied in the educational domain. Prior research in instructional strategies [Cro77; Sno91] has shown the existence of aptitude-treatment interaction (ATI), where higher performing students are less affected by a treatment than lower performing students. Educational researchers have discovered ATI effects based on prior experience level, prior working memory, and incoming self-regulated learning ability [Kal01; Leh16; Fuc19; Yeh15]. For example, Lehmann et al. explored the effect of working memory on learning outcomes in fluency/disfluency groups, where instructional materials had different levels of text legibility [Leh16]. Based on these findings, there could be an aptitude-treatment effect associated with how RL step-level hint policies affect students of differing levels of prior proficiency, affecting lower performing students the most. Therefore, I investigated the potential differing effects between high and low performing students (based on a median pretest split described in Section 5.7).
This work seeks to compare two offline, off-policy RL step-level hint policies. One policy uses Deep Q-Networks to approximate the Q-function for whether or not to give an Assertion, i.e. a proactive partially worked step [Mni13; Mni15]. This method has already been proven effective with PS/WE policies within Deep Thought [Zho19]. The second RL policy uses a Long Short-Term Memory (LSTM) [Hoc97] architecture that considers the previous three steps and is induced with training data using inferred rewards described above [Azi].

5.2 The Deep Thought Logic Proof Tutor

The Deep Thought tutor is used in the context of a discrete mathematics course where students first spend 2 weeks learning about truth tables, and proving each logic rule is true in class and in online multiple-choice homework assignments. Then, students learn about formal proofs, where students iteratively apply logic rules to a set of given statements to derive a specified conclusion. A formal proof works much like any multi-step procedural problem where domain principles are applied to given and previously-derived facts to derive and justify new statements. For example, in physics, students may be given values for mass and acceleration and be asked to determine force. They would then apply the domain principle of \(F = ma\) along with the given values of \(m\) and \(a\) to derive a new statement about the value of \(F\). In logic, each derived statement must have a justification which consists of the domain principle and the relevant prior statements it was applied to. This corresponds to the information used to derive \(F\) in the previous physics example.

Within the discrete math course, students next complete partially-worked examples in a fill-in-the-blank type interface where they are given formal logic proofs with one missing part on each step - either the derived statement, or part of the justification that consists of the rule used to derive it and the statements the rule was applied to. Many example logic proofs are worked in class, with students asked to actively solve logic proofs in small groups, and students are provided with several full worked examples in handouts. After this class work and homework, students are assumed to have reasonable familiarity with logic rule application, but need practice in determining which rules to apply in service to a problem-solving goal. Students are then assigned to complete formal logic proofs using our propositional logic tutor called Deep Thought [Mos17].

The intention of the Deep Thought tutor is to provide students with practice on solving logic proofs with a focus on problem-solving efficiency in both time and the number of steps in their solutions, i.e. shorter proofs in less time, and ideally with few mistakes in justifying or deriving new statements. To do so, the tutor must provide basic functionalities including (1) correctness feedback on each step (on both justification and derived statements), and (2) automated detection of proof completion. Like a compiler, Deep Thought provides these functions that identify errors and clearly shows when a problem is complete but do little to help students with the overall goal of reaching a problem solution through deriving and justifying a series of well-chosen statements. To bridge this gap, the Hint Factory was created to provide data-driven assistance that could point students to appropriate subgoal statements to derive [Bar08; Sta08; Sta13].
Deep Thought allows students to solve logic proofs graphically as shown in Figure 5.1.

![Deep Thought Workspace Diagram](image)

**Figure 5.1** On the left of the screen is the Deep Thought workspace. Below the workspace are are the hint button and hint message box, the rules are in the middle, and to the right is the Dialogue Box where messages related to Assertions as well as problem information are given.

As stated above, Deep Thought is intended to teach students to solve proofs more efficiently, in terms of time and steps taken to reach the conclusion. The tutor presents proof problems as an initial set of given statements with a conclusion to derive from them using logic rules. Each statement, given or derived, is represented by a node, with the conclusion represented with a node with a question mark ‘?’ above it, indicating that it has not yet been justified (shown to be true using logic rules applied iteratively to the givens). Each problem-solving step consists of two parts: the justification and the derived statement. The justification is the set of 1-2 existing nodes and the rule applied to them, and the derived statement is the result. Students complete the justification by clicking to select 1-2 nodes, and clicking on a rule to apply. Students then type in the derived statement that results from applying the rule to the selected statement nodes (see Figure 5.2). Throughout the tutor, including the pre- and post-test problems, Deep Thought provides immediate error feedback for mistakes - either in justifications or derived statements. If a student clicks on the wrong rule, or their derived statement does not follow from the selected nodes and rule, Deep Thought shows a popup message and records the error. For example, if a student selects two nodes and then clicks on the *Simp* rule, the error prompt reads “Rule requires one premise,” then fades away. If the student enters a derived statement that is true, and the justification (consisting of the selected nodes and rule to derive it) is correct, then a new node with the derived statement appears in the workspace. To complete a problem, the student must iteratively derive and justify new statements,
until the conclusion statement is derived and justified. When students have completed a problem, the conclusion's question mark is removed, and it is visually connected to the givens through a series of derived nodes and arrows indicating their justifications. Since the system automatically checks each step and detects completion in all phases of the tutor, student solutions cannot be incorrect, but some may be more expert than others. Students are considered to have learned the topic when they perform well on the posttest, especially with regard to problem solutions with fewer steps and fewer mistakes in less time.

Deep Thought includes four phases: introduction, pretest, training, and posttest. The introduction consists of three problems including two worked examples, where students click through the addition of successive nodes until a conclusion is derived, and a third problem students solve alone to learn the interface. Then, students take the pretest consisting of a solving single problem with no hints available. The pretest is used to measure students’ incoming proficiency and assign them to conditions via stratified sampling. Next, students solve 18 problems in the training section. For each training problem, the dialogue box provides information on what rules to focus on while solving a problem, such as “Think about the following rules for this problem: MP, Simp, Add.” Students also receive contextual, data-driven hints during training, including both unsolicited hints generated by the system and on-demand hints upon student request, all generated using the same Hint Factory-type approach described below. After completing training, students take a more difficult, non-isomorphic posttest, where they must solve four problems without any help or assistance. Since the posttest is not isomorphic to the pre-test, I do not expect the post-test performance to be directly comparable to the pretest performance. Rather, I use the pretest to balance incoming proficiency across groups via stratified sampling, and focus on comparing post-test performance between groups.

Expert solutions for all tutor problems range from 5-8 steps, and student solutions typically contain 5-20 steps. Longer student solutions may simply be inefficient, taking more steps than needed, or they may contain unnecessary nodes that do not lie on a direct path from the givens to
the conclusion. To help students avoid deriving unnecessary statements/nodes in the training phase, the tutor colors nodes based on their necessity and frequency in our historical dataset of correct solutions by past students. Nodes that were never necessary to derive the conclusion are colored gray, while frequently-necessary nodes are colored green, and infrequently-necessary nodes are colored yellow.

5.2.1 Assistance

In training problems, students in the Reinforcement Learning policies receive unsolicited hints, called Assertions, but all students may also request on-demand hints, derived using the same data-driven Hint Factory method, have the same content, and are given in the form of a message (MSG) in the box beside the request hint button, and marked with ‘?’ to show that they have not yet been justified. Assertions are provided in the workspace labeled as "Goal" and resemble the nodes students use to derive new steps. In previous work, I showed this method of providing unsolicited Assertion hints resulted in better performance than text-based messages as a method of unsolicited hint delivery [MM20b]. When requesting MSG hints, students are shown an increasingly specific hint to the next step — these are capped to 4 per problem to prevent gaming of the tutor. This cap prevents students from seeing more than one bottom-out hint per problem. To use a hint in their proof, the suggested hint statement must first be justified by applying a rule to previously-justified or given statements. A hint statement is said to be adopted, or necessary to a student’s solution, if there is a path that connects the hint statement node to the conclusion.

Assistance provided by the tutor in the form of hints can either be initiated by the student, in which case they are called on-demand hints, or they can be initiated by the tutor, in which case they are called unsolicited hints. In this work, I used our recently-designed Assertions interface to place all next-step hints, which consist of a suggested next statement to derive, as nodes marked with a question mark and a ‘Goal’ label (to denote that they have not been justified) in the workspace as shown in Figure 5.3. To use a hint in their proof, students must first correctly justify the hint by indicating which prior nodes and rule can be used to derive the node’s statement. In training problems, all students may request hints in addition to those provided or offered by the tutor according to their study condition. Pretest and posttest problems disallow any hints. Hints do not tell students which rules or prior nodes can be used to justify the suggested statement. Rather, our hints are designed to help students solve problems by suggesting a subgoal statement that helps them break down multi-step problems into smaller chunks. All hints are next-step, which suggest the next, best statement that can be derived in one rule-application step from the student’s current state. For example, one tutor hint could add an $F$ node as a subgoal, and the student would select $I \land F$ and choose the “Simplification” rule to justify it (See Figure 5.3).

Deep Thought uses the Hint Factory [Sta08] approach to generate hints. The Hint Factory [Sta08; Sta13; Bar11] is a data-driven method to generate hints by transforming historical student problem-

---

1 Unnecessary nodes in a complete solution are easy to detect because removing them does not disconnect the conclusion from the givens, but they are difficult to detect during problem solving.
solving attempts into a Markov Decision Process, using observed frequencies as transition probabilities, and estimating the expected value of each previously-observed problem state based on assigning rewards to complete solutions, small negative rewards (i.e. costs) to steps to positively reward more efficient solutions, and large negative rewards to errors to de-emphasize solutions that cause many students to make mistakes. Individual student problem-solving attempts are represented by a series of states, or snapshots of the work done so far, where transitions occur between states when students add or delete problem nodes, or make an error. The Hint Factory is described in detail in Barnes and Stamper’s chapter in the 2011 Handbook on Educational Data Mining [Bar11]. All student solutions are combined into an interaction network [Eag12] that reflects all previously-observed solutions to one specific problem. When a hint is requested by the student or tutor, the Hint Factory is used to select a target problem-solving state with the highest expected value. Note that this process can be done offline, and a simple table can be used to store problem-solving states and their corresponding hint content for real-time hint provision. Then, the latest statement derived in that state is used as the pointing hint to help students know what to try to derive next. As stated above, in this study I do not provide further information on how to derive or justify the suggested statement, meaning that these hints can be considered as partially-worked example steps. In this study, our Assertions and MSG hints are derived as above, with the target state selected to be the one with the highest expected value that occurs within one rule application from the student’s current state. This corresponds to the next-step hints derived in all of our prior work using Hint Factory [Bar08; Bar11; Sta13; Sta08; Eag14b; Eag12; Pri17a; Pri17b]. Since Next-Step hints are partially-worked, they allow students to focus on how to justify them, and reflect on why they were suggested. This removes a considerable load; without a hint, students must also search among many options for the best what to derive next.

Deep Thought includes several measures intended to prevent gaming the system, where students attempt to use system features to avoid work, or help abuse, where students request hints when they do not need them [Bak04]. First, whenever an unjustified hint is already in the workspace, students

---

**Figure 5.3** A hint statement being justified. (1) Shows a hint appearing in the workspace. (2) Two statements are selected (highlighted in blue) and the rule “Modus Ponens” applied. (3) The hint has been justified and is connected to the student proof.
may not receive another hint, whether it was solicited or provided automatically by the tutor. All hints provide a target statement to derive, appearing as a node with a ‘?’ in the workspace.

Within our tutor, even though students often have difficulty and hints are readily available via the hint button, most students do not request assistance. In Fall 2017, students using our tutor requested a median of zero hints per problem. Research has shown that there is a pervasive problem with help avoidance within intelligent tutoring systems [Ale04]. As a result, some ITSs employ unsolicited hints to prevent help avoidance [Van06]. In this study, to enable us to compare the impact of hint type, I periodically provided unsolicited hints to students based on the condition they were assigned, as described below.

5.3 Method

The Deep Thought tutor was used as a homework assignment for an undergraduate ‘discrete mathematics for computer scientists’ course in the Fall 2019 and Spring 2020 semesters. I analyzed 108 students’ data from three test conditions to investigate the impact of the Reinforcement learning step-level hint policies on student performance and behavior. The three conditions are: Control ($n = 37$, independent problem solving with on-demand MSG hints), DQN ($n = 36$, step-level hint policy), and LSTM ($n = 35$, step-level hint policy). All of the conditions were identical except the DQN and LSTM groups provided Assertions, proactive partially worked steps, as unsolicited hints to the students. I used stratified sampling based on pretest performance and randomly assigned students to each condition in each semester, ensuring all conditions were balanced in incoming knowledge.

The Control and LSTM hint policy were deployed and collected in the Spring of 2020 during the COVID-19 pandemic while students were adjusting to new circumstances. I note that this could have an effect on the resulting data. To control for the potential differences in student performance, I selected a matched sample from the Fall 2019 DQN hint policy students. To select the matched sample, I used fuzzy clustering [Bez84] with the centers of each cluster set to students in the LSTM group. I used this group instead of the Control for a more fair comparison between the Reinforcement Learning policies. For the clustering features, I used the pretest Total Time, Total Steps, Accuracy, and the time they started the tutor relative to the due date. I chose students based on assigned memberships to each cluster above 75% resulting in 36 students being chosen out of 70.

The following subsections discuss the methods used in creating the DQN and LSTM policies including the data used for training each step-level hint policy, the reward generation methods, the policy induction methods.

5.3.1 DQN Data Used for Training

In this study, I applied offline, off-policy reinforcement learning using a deep Q-network to learn a policy for whether students should receive step-level assertions during problem solving based on two semesters of prior data. The prior data were collected in Study 1 in Fall 2018, with 56 students
as reported in Chapter 3, and Study 2 in Spring 2019, with 38 students. Overall, the data contain 94 students who completed the tutor and were randomly given proactive Next-Step Assertions. Although more data would be better for inducing a policy, these data were determined to be sufficient and included a high number of student-action pairs, with a total of 15,384 student step-action pairs. Each step included all of the data, including features like time taken on that step, on-demand Hint Requests, and average accuracy, from our step-student model.

5.3.2 DQN Reward Generation

In order to build an RL policy, I first need to generate rewards for each potential step-action pair. Generally, for educational systems, the reward for a particular step is not immediately known, as what the student has learned from a single step cannot be measured until some point in the future, which may be the full problem being completed or possibly a full level. Therefore, policies within the educational domain use delayed rewards to represent learning, and then must distribute the rewards across all previous steps so they sum to the final delayed reward. For this dataset, I generated rewards using custom R scripts and used a standard method of evenly distributing the overall problem reward to each step in a problem, as described in more detail below.

The reward, Equation (1), for each problem is a combination of delayed rewards at the following levels: problem, level, and posttest. Each of these delayed rewards is based on a per-student improvement calculation (Improvement Score), Equations (3). The features avgAccuracy and avgStepTime in Equation (3) were based on raw data. For ImpAccuracy and ImpStepTime, were scaled to be within the same range.

The reward also has additional modifications based on a class-relative ranking (Relative Score) of how well the student is performing in comparison to others in terms of a score based on Accuracy, TotalTime, and TotalSteps for that problem - Equation (2). Accuracy, TotalTime, and TotalSteps have been normalized and standardized, and TotalTime and TotalSteps have been flipped so that high total time or steps causes a decrease in the score via subtracting the normalized value from 1. Scores above 20 were the highest performing students, meaning that their performance was high and the improvements between problems and levels were low. Due to most of our policies rarely influencing high performing students, I decided to augment the rewards for the higher performing students so that the variance may help the model learn a better policy for them. Lastly, rewards for each student-action pair were created from the above by distributing them via the majority action for the problem.

Below are the equations for calculating the rewards. Equation (1) represents the final reward generated for each problem. Equation (2) is the score based on the class-relative ranking. The sets of equations under Equations (3), (4), and (5) are the delayed rewards for the problem, level, and posttest per-student improvement calculation. The method of distributing the final problem rewards is shown in Equation (6) along with the resulting final step reward for each student-action pair.
\[
\text{Reward} = (\text{Improvement Score}_{\text{Posttest}} \times 0.5 + \text{Improvement Score}_{\text{Level}} \times 0.3 + \text{Improvement Score}_{\text{Problem}} \times 0.2) \times \beta,
\]
if \(\text{Rel. Score} > 20\) : \(\beta = 2\),
if \(0 \leq \text{Rel. Score} \leq 20\) : \(\beta = 1.5\),
and if \(\text{Rel. Score} < 0\) : \(\beta = 1\)

(5.1)

\[
\text{Rel. Score}_n = \text{Total Time}_n \times 5 + \text{Total Steps}_n \times 3 + \text{Accuracy}_n \times 2,
\]
where \(n\) is the current problem

(5.2)

\[
\text{Improvement Score}_n = \text{Imp Accuracy}_n + \text{Imp Step Time}_n
\]

where \(n\) and \(m\) represents the following for each scenario:

- Posttest: \(n = \text{Posttest}, m = \text{Pretest}\)
- Training Level: \(n = \text{current level}, m = \text{previous level}\)
- Problem: \(n = \text{current problem}, m = \text{previous solved problem}\)

(5.3)

\[
\text{percentage Of Assertion Steps} = \frac{\# \text{Assertions}}{\# \text{Steps}}
\]
if Assertion was given,

\[
\text{Step Reward} = \text{Reward}_{\text{Problem}} \times \text{percentage Of Assertion Steps}
\]
if Assertion was not given,

\[
\text{Step Reward} = \text{Reward}_{\text{Problem}} \times (1 - \text{percentage Of Assertion Steps})
\]

(5.4)

### 5.3.3 DQN Policy Induction

Next, I used a state-of-the-art algorithm, Deep Q-Networks [Mni13; Mni15] using offline, off-policy learning to induce the policy. This method has already been proven effective with PS/WE policies within Deep Thought [Zho19]. A multi-layer perceptron neural network was used to approximate the Q-function. The inputs to the neural network were the last observation of the Step Student Model and the outputs were the Q values for each possible step level action (give Assertion (partially worked step) or let student work the full step on their own).

The network consists of 3 (512 unit, 1024 unit, 128 unit) hidden layers with the rectified linear unit (ReLU) activation function and a linear output layer (see Figure 5.4). I created this network by starting with the previously successful model and tuning the hyperparameters to find the best result for this dataset. I used batches of 500 samples and 20,000 iterations to train the model. Instead
of individually providing samples to update the weights, providing larger batch-sizes reduces the likelihood of overfitting. Furthermore, 20,000 iterations were chosen after initially training for 50,000 iterations and seeing a decrease in performance, possibly due to overfitting. Before deciding on this DQN model, I tried other methods such as decision forests and switching out a layer for a dropout layer to reduce the chances of overfitting. However, the DQN method described outperformed the others in terms of Expected Cumulative Reward (ECR) and ECR convergence.

Figure 5.4 Neural Network architecture diagram for the DQN model.

5.4 LSTM Data Used for Training

For the LSTM, I continued my application of offline, off-policy reinforcement learning to learn a policy for whether students should receive step-level Assertions during problem solving based on five semesters of prior data. Data were used from the random hint policies in Study 1 in Fall 2018, with 56 students as reported in Chapter 3, and Study 2 in Spring 2019, with 38 students. Furthermore, data were used from the DQN step-level Assertion policy in Fall 2019 with 71 students and from another step-level Assertion policy with 66 students. Lastly, data was incorporated from the 29 students in the Spring 2019 Control in Study 2 by considering student’s Hint Requests as Assertions given (requested hints in the Spring 2019 version of the tutor gave students Assertions instead of MSG-based hints). Overall, 260 students’ step-level data with a total of 32,462 student step-action pairs were used (over double the amount of data compared to the DQN policy). Each step included all of the data, including features like time taken on that step, on-demand Hint Requests, and average accuracy, from our step-student model.

An important consideration in using this variety of data sets in our training is the effect it has on the learned policy. There was a range of frequency and timing of when Assertions were given in each
of the policies (random, student, and intelligent policies). The data from the Spring 2019 control had the fewest amount of hints given and were based on student's decisions of when they needed hints, which is known to be sub-optimal [Zho16; Ale00]. However, this data set provided a wider range of frequency that I hoped would help our learned policy make an effective policy with an average of 19 hints received. The two random hint policies and the intelligent policies (DQN and the other Fall 2019 policy) provided hints frequently (on average 35-55 hints), with the DQN providing hints most frequently. With a variety of frequency and timing of Assertions, I hoped capturing the most effective way would be easier to determine and that the larger amount of student data would be beneficial; however, I acknowledge that this could potentially negatively influence the LSTM learning by containing too much noise from the multitude of policies or including a policy from which student requested hints were considered Assertions.

5.5 LSTM Reward Generation

Based on preliminary results of the Fall DQN policy, I modified the reward generation algorithm in Section 5.3.2. The results showed that the policy was suboptimal for the high-proficiency - with high proficiency students performing worse than the initial lower proficiency students by the Posttest. Therefore, the following equation was used to calculate the new rewards:

$$\text{Reward} = (\text{Improv. Score}_{\text{Posttest}} \times 0.7 + \text{Rel. Score}_{\text{Posttest}} \times 0.3) \times 0.5 + \text{Improv. Score}_{\text{Level}} \times 0.3 + \text{Improv. Score}_{\text{Problem}} \times 0.2$$

(5.5)

Furthermore, I updated the method of distributing the delayed rewards. Instead, I used a method to intelligently distribute the delayed rewards using the method to infer immediate rewards from delayed rewards described in the paper by Azizsoltani et. al [Azi]. In this method, delayed rewards are distributed using a Gaussian Process (GP)-based immediate reward approximation algorithm. Due to how crucial an accurate reward function is to generating an effective policy, approximating the immediate rewards can drastically change the ultimate policy that is learned by making the rewards more consistent with what step actually influenced the eventual delayed reward. Furthermore, this method was shown to produce better learning outcomes on the posttest compared to studies using only delayed rewards.

5.6 LSTM Policy Induction

In attempt to improve on the DQN’s results in the Fall, Long Short-Term Memory (LSTM) [Hoc97] layer was used instead of the Fully-Connected Networks used for the DQN. LSTM’s have been used with better or similar success than memory-less models within the educational domain [Zha18; Mao18]. Therefore, I believed switching to this network could improve upon our results because
using a time-series approach allows the model to see a bigger picture of how the student is currently performing while problem solving. Therefore, I used Long Short Term Memory (LSTM) to estimate the action-value function $Q$. Our LSTM architecture consists of one layer of 400 LSTM units, and a linear layer as output (see Figure 5.5. Additionally, for a given time $t$, I explored three step input settings: 1) $k = 1$ that use only the last state $s_t$; 2) $k = 2$ that uses to use the last two states: $s_{t-1}$ and $s_t$; and 3) $k = 3$ for using $s_{t-2}$, $s_{t-1}$ and $s_t$. L2 regularization using the Adam Optimizer method [Kin14] was employed to get a model that generalizes better to unseen data and avoid overfitting. I initially trained the models for 50,000 iterations to see where the model's converge, then retrained to using iterations closer to their convergence to avoid overfitting. The final model was training over 25,000 iterations. Multiple batch sizes were explored from 50 to 500 and the optimal batch size found, in terms of convergence, was 300.

![Figure 5.5](image)

**Figure 5.5** Neural Network architecture diagram for the LSTM model.

Although this architecture is smaller, in terms of both layers and units, compared to the Fall DQN architecture, the model is not less complex. The reason for this is that for each LSTM unit, there are 4 times as many trainable parameters. To determine if the LSTM architecture performed better than the DQN, in terms of ECR, I reran the same DQN architecture on the new training data. I found that the new LSTM architecture converged to a higher ECR compared to the DQN's ECR at convergence. This result led us to choosing the LSTM architecture over the DQN approach.

### 5.7 Analysis and Performance Metrics

During the training phase, students solve 18 training problems with access to both unsolicited and on-demand hints. The Posttest consists of four final unassisted problems. I investigated pre- to posttest changes as well as performance impacts on time spent solving a problem, total attempted
steps, and accuracy. **Total time** is counted from the moment a problem begins until it is solved by deriving and justifying the conclusion. **Total steps** in a problem include any attempt at deriving a new node, which includes correct and incorrect steps. **% Adopted Steps** is the percentage of steps that were used to derive the conclusion, as opposed to derived but not contributing to the final solution (the concept of adoption is further discussed later in this section). **Accuracy** is the percentage of correct out of the total steps, which is expected to start relatively high due to prior exposure in the class, and increase as students practice. Note that the tutor is not designed or assumed to promote large improvements in accuracy, since no penalties are assigned for incorrect rule applications and the tutor simply alerts students upon wrong rule applications and students may try again, even within the pre- and post-tests. Further, problems require new rules and become more difficult as the students progress. As I seek primarily to promote more efficient problem solving, I focus more on steps and time per problem while maintaining reasonable accuracy. This is because it is more difficult for students to learn to determine which steps to derive to achieve shorter, more efficient proofs, compared to learning how to apply the rules, which can be done by memorization and simple practice. Deep Thought is built primarily to allow students to practice with the strategy of problem solving, rather than fluency with rules, most of which are assumed to be learned before the tutor.

To analyze the RL step-level hint policies’ behavior, multiple Assertion metrics are used. **Total Assertions** are the total number of Assertions that were added to the workspace. **% Steps Hinted** is the percentage of derived steps that received an Assertion. **% Steps Helped** is the percentage of derived steps that received an Assertion and the student justified. One important thing to note is that Deep Thought does not include eye-tracking, and the Assertions are provided regardless of whether a student needs them or not, so I cannot determine precisely whether students followed a hint or incidentally derived the hint statement. Therefore, I have defined metrics to quantify when students *justified* an Assertion by selecting the statements and rule needed to derive it, and when a hint is justified, the tutor removes its ‘?’ and connects it to its predecessor nodes with arrows labeled with the rule used to derive it. A hint justification provides evidence that a student noticed the hint and knew how to apply rules to justify it, but do not tell the full story. As in any problem-solving context, statements can be derived that are not needed in a final solution. Therefore, I also measure the hint Adoption Rate, whether a hint contributes towards deriving the conclusion. A justified hint can be reached on a path from the problem's given statements. When a hint is *adopted*, it must first be justified and then become necessary to a student's final solution – in other words, the problem would be incomplete if the hinted statement were removed. This is shown visually when a directed path can
be found from the hinted statement node to the problem conclusion. Figure 5.6 shows a completed problem with labels indicating which nodes are considered *justified* and which nodes were also *adopted* for the solution. Based on these definitions, **Total Justified** and Total Adopted represent the total number of Assertions that were justified and adopted, respectively.

**Figure 5.6** A completed problem with nodes that were used to derive the conclusion (justified and adopted) and one node that was not used to derive the conclusion (justified but not adopted).

I also investigated impacts on help-seeking through the number of on-demand Hint Requests (when students click the “Get Suggestion” button). **Hint Requests** represents the number of Hint Requests during the training portion. All hints requested are presented as messages in the box beside the “Get Suggestion” button. **Total MSG Hints Received** is the total number of new message-based(MSG) hints that were given to students. This number differs from the Hint Requests because students may request a hint when they already have one and another one is not available. **Total Hints Received** is the total of both MSG hints and Assertions received. Data were analyzed to compare groups for the pretest, training, and posttest portions of the tutor. Within each hint group, I also compared performance of students with High or Low pretest scores, based on a median split on the pretest Relative Score.

To determine significant differences between hint types, I applied one way ANCOVA using the pretest as a covariate with Benjamini-Hochberg corrections to account for multiple tests. Initially, to account for the difference in number of hints given between the groups due to the Assertions, I added the number of hints received as a covariate. However, this variable was not significant and, therefore, removed from the model.
To check that the data met assumptions for ANCOVA, I used the Shapiro-Wilk's W test and Levene's test, as well as visually inspecting the data via Q-Q plots and histograms. Data that did not meet the assumptions were transformed using log or square-root transformations, then re-inspected. For data that still did not meet assumptions after transforming, Kruskal-Wallis test with Dunn post-hoc test and Bonferroni corrections were used. For clarity, all data in tables are reported before transformation.

The research questions I sought to answer with this study are:

1. Can Reinforcement Learning step-level policies benefit student’s more than self-regulated problem solving?
2. Will an LSTM architecture using 3 steps be able to better learn when to provide a hint then a single step DQN architecture and result in better overall performance on the posttest?

My hypotheses were as follows:

1. $H_1$: The RL hint policy groups will perform better, in terms of total time and steps, in the training compared to the Control.
2. $H_2$: The RL hint policy groups will result in overall better performance on the posttest, especially for lower performing students which are generally more affected by treatments than higher performing students.
3. $H_3$: The LSTM group will perform better on the training and posttest than the DQN group.

5.8 Results & Discussion

To compare the differences in frequency of proactive hints between the DQN and LSTM hint policies, I examined several hint-related metrics (see Table 5.1). There was a significant difference in the Total Assertions added ($F(2, 68) = 15.67, p < 0.01$), Total Justified ($F(2, 68) = 29.52, p < 0.01$), and Total Adopted ($F(2, 68) = 9.55, p < 0.01$) between the two groups with the LSTM group giving much fewer proactive hints. I examined the portion of those hints that were utilized and found no significant difference in the percentage Justified between the groups ($F(2, 68) = .99, p = 0.37$), indicating that students were justifying a similar portion of the hints they were given. However, there was a significant difference in the percentage Adopted between the groups ($F(2, 68) = 3.41, p = 0.03$) with the LSTM group adopting more hints into their solution, indicating that the hints were more useful to the LSTM group. To give a better picture of how many steps in the training portion of the tutor were assisted, the % Steps Hinted and % Steps Helped metric are listed and significant differences were found (% Steps Hinted ($F(2, 68) = 19.33, p < 0.01$), % Steps Helped ($F(2, 68) = 17.61, p < 0.01$)), which is expected given the difference in total Assertions given. These results partially confirm $H_3$, as the LSTM did provide fewer hints. However, I discuss whether these hints were provided when students were struggling below and in Section 5.14.
To examine the effects each policy had on performance, I analyzed the pretest and posttest performance metrics for the 3 groups (see Table 5.2). These analyses were conducted in order to address my performance-centric hypotheses (H1, H2, and H3). ANOVA was performed on pretest metrics to determine if there was a similar distribution of proficiency between the groups. There were no significant differences between the groups in the pretest metrics (on Total Time (F(2, 105) = 1.04, p = 0.35), Total Steps (F(2, 105) = 0.20, p = 0.81), Accuracy (F(2, 105) = 0.57, p = 0.56), and % Adopted Steps (F(2, 105) = 1.52, p = 0.22)). Therefore, I concluded that each group had a distribution of students’ with similar incoming proficiency. For the training and posttest performance metrics, ANCOVA was used, controlling for pretest metrics. There were significant differences between multiple performance metrics in the training portion of the tutor (Total Time (F(3, 104) = 6.33, p < 0.01); Total Steps (F(3, 104) = 7.15, p < 0.01); % Adopted Steps (F(3, 104) = 6.14, p < 0.01)); and Accuracy (F(3, 104) = 9.24, p < 0.01). For Total Time, Tukey post-hoc analysis revealed that there was a marginally significant difference between the Control and LSTM group (p = 0.07), indicating that the LSTM group went through the training faster than the Control group. For Total Steps, Tukey post-hoc analysis revealed that there was a significant difference between the Control and DQN group (p < 0.01), indicating that the DQN group was able to solve the problems in fewer steps. Although the Total Time between the Control and DQN group and the Total Steps between the Control and LSTM group were not significantly different, the time and steps were faster and fewer, respectively. These results show that the DQN and LSTM group performed faster with fewer steps than the Control in the training. These results provide evidence for H1 which posited that the RL hint policy groups would outperform the Control in training, and thus, I partially accept H1 . The reason for the partial acceptance is that neither the DQN nor LSTM group outperformed the Control group in all performance features, but the results indicate that the policies did slightly improve the students’ training performance. These results also provide evidence in rejecting H3 due to the LSTM group performing similarly to the DQN group in the training; however, posttest performance still needs to be considered.
Additionally, I looked at MSG hint-related features and for these analyses I used Kruskal Wallis with post hoc Dunn test to determine significant differences between the groups because the data could not be transformed to meet ANOVA assumptions that the data be normal. Square root and log transformations still produced significant differences in normality of the data. There was a significant difference for both Hint Requests ($H = 6.32, p = 0.04$) and MSG Hints Received ($H = 14.22, p < 0.01$) between the groups. For Hint Requests, the Dunn test post-hoc analysis revealed that there were significant differences between the Control and DQN ($p = 0.03$) and the DQN and LSTM ($p < 0.01$). The Control and LSTM had a similar amount of Hint Requests ($p = 0.27$). Consequently, a similar amount of MSG hints were received, with the Control and LSTM being similar ($p = 0.49$) and the DQN group totals being significantly different than the LSTM group ($p < 0.01$) and Control ($p < 0.01$). This result isn’t surprising given the high number of proactive hints given in the DQN group (30% of steps were provided hints), most likely resulting in the lower number of hints requested.

For the non-isomorphic posttest metrics, there were no significant differences between the groups for % Adopted Steps ($F(5, 102) = 1.30, p = 0.26$). However, there was a significant interaction between the pretest and group discovered for % Adopted Steps ($p = 0.04$), this result is discussed later in this section. A significant difference in the Accuracy between the groups was found ($F(3, 104) = x, p < 0.01$), with Tukey post-hoc analysis showing a significant difference between the Control and LSTM group ($p = 0.02$) and the LSTM and DQN group ($p < 0.01$), with the LSTM having an overall worse accuracy on the posttest. Significant differences between the groups were found for Total Time ($F(5, 102) = 2.64, p = 0.02$) and Total Steps ($F(5, 102) = 2.73, p = 0.02$), and a significant and marginally significant interaction, respectively, was discovered between the pretest metrics and the group for those metrics Total Time ($p = 0.01$) and Total Steps ($p = 0.08$). For Total Time, Tukey post-hoc analysis revealed that there was a significant difference between the Control and DQN group ($p = 0.05$) and a marginally significant difference between the LSTM and DQN group ($p = 0.06$). For Total Steps, Tukey post-hoc analysis revealed that there was a significant difference between the Control and DQN group ($p = 0.04$). I also examined the Relative Score defined in Section 5.3.2. For the Relative Score, a significant difference was found between the groups ($F(5, 102) = 2.83, p = 0.02$) and a significant interaction with the covariate was discovered ($p = 0.02$). Tukey post-hoc analysis revealed that there was a marginally significant difference between the Control and DQN group ($p = 0.06$) and the LSTM and DQN group ($p = 0.09$). For all posttest performance metrics with significant interactions, the regression plots with posttest metrics plotted against pretest metrics showed a similar trend. Figure 5.7 demonstrates this trend using the Relative Score metric. For both the Control and LSTM, a slight positive slope (less time/fewer steps on pretest = less time/fewer steps on posttest); however, the DQN group’s regression slope was slightly negative (less time/fewer steps on pretest = more time/steps on posttest). These results indicate that the DQN group benefited the lower performing students more than the initially high performing students and may have harmed the higher performing students. One important note is that the regression plot for the DQN group was flatter compared to the LSTM and Control group indicating that the initially low and high performing students performed more similarly to each other compared to the Control and LSTM
group. Both the Control and the LSTM group performed similarly on all of the posttest metrics.

To compare the groups overall improvements on the non-isomorphic posttest, I examined the Improvement Score (results reported are normalized (0,100)). I found a significant difference between the groups for the Improvement Score ($F(3, 104) = 53.61, p < 0.01$) and no significant interactions. Tukey post-hoc analysis showed that there was a significant difference between the DQN and LSTM group ($p = 0.01$); however, no significant difference was found between the DQN and Control ($p = 0.67$) or the LSTM and Control ($p = 0.10$). This indicates that the DQN policy leads to higher overall improvement than the LSTM policy, based on the defined Improvement Score.

For the isomorphic posttest, there were some significant or marginally significant differences found for Total Time ($F(3, 104) = 2.52, p = 0.06$), Total Steps ($F(3, 104) = 2.86, p = 0.04$), Accuracy ($F(3, 104) = 2.93, p = 0.04$), and % Adopted Steps ($F(3, 104) = x, p = 0.06$)). Although Tukey post-hoc analysis showed no significant differences between any of the groups for any of the isomorphic posttest performance metrics. However, the LSTM group seemed to perform better overall with the shortest time, fewest steps, and highest % Adopted steps compared to the Control and more similar performance compared to the DQN group.

One of my intended goals in providing proactive assistance was to reduce the time spent learning without reducing student’s performance. Even though neither policy performed better in the posttest, my results in the training indicated that DQN and LSTM group performed better in the training with regard to time, than the Control.

An important observation from the training is that the percentage of Adopted Steps is higher, most likely in part due to the percentage of the Adopted Assertions being higher. These results combined with the frequency of hints between the groups (see Table 5.1) showing the LSTM group received fewer Assertions indicates that the LSTM group may have been provided more useful hints, resulting in the lowest training time and higher percentage of Adopted Steps. Considering that the LSTM group performed worse overall in the posttest suggests that these hints may have been helping students through harder steps in each problem, reducing their struggle to an extent that inhibited their learning. This result provides some evidence that the LSTM would better provide hints when students were struggling. However, due to the overall worse performance on the posttest in comparison to the DQN and the similar performance in the training, I reject $H_3$ which posited that the LSTM would perform better in both the training and posttest in comparison to the DQN.

The DQN group had the fewest steps in the training and the highest amount of Assertions received, but ended up with a similar total training time as the Control and LSTM (although the DQN group was slightly faster than the Control in training). Due to the percentage of Adopted Assertions being lower and the total training time not being significantly different from the Control, these hinted steps were not as useful to their final solution, but provided them with practice. Combined with the results of the LSTM group, the DQN students may have been receiving hints on less difficult steps allowing them more room to struggle, but providing them shortcuts for easier steps — resulting in the overall better performance on the posttest than the LSTM group.

Furthermore, my results in the non-isomorphic posttest showed the LSTM performed similarly
to the Control. For the DQN group, the results indicated that lower performing students performed better than similar lower performing students in the Control and LSTM groups, and higher performing students may have performed worse than similar higher performing students in the Control and LSTM groups. This is possibly due to the higher number of Assertions received, which may have been harmful to higher performing students. This aligns with research showing that providing too much information to higher performing students can hinder learning [Sal10].

These results provide some evidence in support of $H_2$ given the DQN group’s performance. The hypothesis posited that the RL hint policy groups would perform better on the posttest compared to the Control, especially for low proficiency students. Although, the LSTM group performed worse on the posttest, the results of the DQN group’s performance indicate that the lower proficiency students may have benefited, in terms of performance on the posttest. To further investigate whether either policy had a significant effect on students with differing levels of incoming proficiency and whether high performing students were negatively impacted, I performed further analysis in Section 5.9.

Figure 5.7: Regression plot of the Relative Score between the Control, DQN, and LSTM groups with the Posttest Relative Score plotted against the Pretest Relative Score.
Table 5.2 Mean and Standard Deviation (SD) for the Performance Metrics in the Pretest, Training, and non-isometric and isometric Posttest. Significance tests’ p-values that are at least marginally significant are bolded and significant values also have an asterisk(*)..

<table>
<thead>
<tr>
<th>Section</th>
<th>Metric</th>
<th>Control</th>
<th>DQN</th>
<th>LSTM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(n = 37)</td>
<td>(n = 36)</td>
<td>(n = 35)</td>
</tr>
<tr>
<td>Pretest</td>
<td>Total Time (min)</td>
<td>53(42)</td>
<td>63(37)</td>
<td>54(38)</td>
</tr>
<tr>
<td></td>
<td>Total Steps</td>
<td>79(67)</td>
<td>82(59)</td>
<td>78(58)</td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>46%(20)</td>
<td>42%(16)</td>
<td>41%(18)</td>
</tr>
<tr>
<td></td>
<td>% Adopted Steps</td>
<td>72%(21)</td>
<td>63%(26)</td>
<td>69%(22)</td>
</tr>
<tr>
<td>Training</td>
<td>Total Time (min)</td>
<td>84(55)</td>
<td>71(33)</td>
<td>65(32)</td>
</tr>
<tr>
<td></td>
<td>Total Steps</td>
<td>193(94)*</td>
<td>156(54)*</td>
<td>171(61)</td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>68%(12)</td>
<td>70%(13)</td>
<td>67%(15)</td>
</tr>
<tr>
<td></td>
<td>% Adopted Steps</td>
<td>79%(10)*</td>
<td>80%(10)*</td>
<td>85%(8)*</td>
</tr>
<tr>
<td></td>
<td>Total Hint Requests</td>
<td>16(21)*</td>
<td>8(12)*</td>
<td>20(22)*</td>
</tr>
<tr>
<td></td>
<td>Total MSG Hints Received</td>
<td>14(17)*</td>
<td>4(6)*</td>
<td>13(14)*</td>
</tr>
<tr>
<td></td>
<td>Total Hints Received</td>
<td>14(10)*</td>
<td>58(17)*</td>
<td>48(24)*</td>
</tr>
<tr>
<td>Posttest (non-iso)</td>
<td>Total Time (min)</td>
<td>47(30)</td>
<td>51(30)</td>
<td>56(37)</td>
</tr>
<tr>
<td></td>
<td>Total Steps</td>
<td>113(74)</td>
<td>107(48)</td>
<td>134(65)</td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>70%(11)*</td>
<td>70%(12)*</td>
<td>63%(13)*</td>
</tr>
<tr>
<td></td>
<td>% Adopted Steps</td>
<td>82%(15)</td>
<td>76%(17)</td>
<td>79%(15)</td>
</tr>
<tr>
<td></td>
<td>Improv. Score</td>
<td>53(27)</td>
<td>58(18)*</td>
<td>51(23)*</td>
</tr>
<tr>
<td></td>
<td>Rel. Score</td>
<td>73(17)</td>
<td>72(15)</td>
<td>66(19)</td>
</tr>
<tr>
<td>Posttest (iso)</td>
<td>Total Time (min)</td>
<td>8(10)</td>
<td>7(15)</td>
<td>6(5)</td>
</tr>
<tr>
<td></td>
<td>Total Steps</td>
<td>18(11)</td>
<td>16(15)</td>
<td>16(8)</td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>81%(16)</td>
<td>82%(13)</td>
<td>77%(16)</td>
</tr>
<tr>
<td></td>
<td>% Adopted Steps</td>
<td>88%(17)</td>
<td>83%(26)</td>
<td>94%(9)</td>
</tr>
</tbody>
</table>
5.9 Effects on Students with High and Low Prior Proficiency

To explore the effects on students with different levels of prior proficiency, I split the students into High and Low based on their Relative Score in the pretest. Then, I looked at the Improvement and Relative Score on the non-isomorphic posttest between the groups. Table 5.3 shows the Improvement Score and Relative Score for High and Low prior proficiency students between the groups. As expected, I did not find any significant differences in the posttest metrics for students with High prior proficiency as high performing students are often less affected by treatment [Cro77; Sno91]. For students in the Low group, I found significant differences in the Improvement ($F(3, 50) = 3.34, p = 0.03$) and Relative Score ($F(3, 50) = 1.37, p = 0.05$). For the Improvement Score, Tukey post-hoc analysis revealed that there was a significant difference between the LSTM and DQN group ($p = 0.05$) and a marginally significant difference between the Control and LSTM group ($p = 0.08$). There was no significant difference between the Control and LSTM group ($p = 0.93$). For the Relative Score, a significant difference was found between the DQN and LSTM group ($p = 0.05$) and no significant difference found between the DQN and Control ($p = 0.47$) or the LSTM and Control ($p = 0.25$). These results indicate that the lower performing students were most affected by the hint policies and that the DQN group was better at assisting these students compared to the LSTM policy. These results provide additional reasons to reject $H_3$, because the DQN group outperformed the LSTM group. As hypothesized in $H_2$, the lower proficiency students were more affected by the study; however, the LSTM group performed the worst and the DQN group has comparable performance on the posttest to the Control. Therefore, $H_2$ is not supported. Although the DQN showed promising results for the low performing students, there was not a significant difference, so I cannot claim that the DQN outperformed the Control on the posttest. However, it is important to note that the DQN group slightly outperformed the Control in training. Given the low sample size of the proficiency splits in these results and an comparable Relative Score on the posttest for the Low proficiency students compared to the High proficiency students, the DQN’s results are encouraging. Additionally, these results suggest that neither policy negatively affected High proficiency students, as previously stated as a concern. However, the LSTM group seemed to have a negative impact on Low performing students, as evidenced by the posttest Improvement and Relative Score.
Table 5.3 Mean and Standard Deviation(SD) for Improvement and Relative Score on the non-isomorphic Posttest Metrics between the groups for High and Low prior proficiency students. Significance tests’ p-values that are at least marginally significant are **bolded** and significant values also have an asterisk(*).

<table>
<thead>
<tr>
<th>Pretest Split</th>
<th>Metric</th>
<th>Control Mean(SD)</th>
<th>DQN Mean(SD)</th>
<th>LSTM Mean(SD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>Improv. Score</td>
<td>42(19)</td>
<td>47(17)</td>
<td>44(21)</td>
</tr>
<tr>
<td>Low</td>
<td>Improv. Score</td>
<td>67(17)</td>
<td>66(14)*</td>
<td>56(13)*</td>
</tr>
<tr>
<td></td>
<td>Rel. Score</td>
<td>77(12)</td>
<td>72(14)</td>
<td>72(12)</td>
</tr>
<tr>
<td>Low</td>
<td>Rel. Score</td>
<td>67(21)</td>
<td>72(16)*</td>
<td>60(22)*</td>
</tr>
</tbody>
</table>

5.10 Investigating Policy Carry-out in the Historical Random Assertion data

To investigate the effectiveness of the policies further, I used the Random Assertion data that was in both of the training data sets from the Fall 2018 and Spring 2019 (n = 94 students) and ran the DQN and LSTM policies over the data. Running these policies produces the action that either policy would have taken seeing the same data. Then, using the actions that the DQN and LSTM policy would have taken plus the action that was taken via the Random policy, I compared when the two decisions to determine when they matched (an Assertion was predicted to be given and an Assertion was given, and an Assertion was not given and the student worked a step). When both the Random and the DQN or LSTM matched on a decision, I say that the decision was “carried-out”. Next, I took the total actions that matched divided by the total actions taken to create the percentage of actions that were carried out (**Carry-out**). For the DQN policy, the mean(SD) Carry-out was 62.8%(8.2). For the LSTM policy, the mean(SD) Carry-out was 48.3%(8.1). Note, due to the Random and DQN policies having similar amounts of Assertions given, the LSTM having a lower Carry-out makes sense.

Using this metric of Carry-out, I looked for correlations of Carry-out with the student’s performance on the posttest. However, no significant correlations were discovered for any performance metric. Due to students taking approximately 150-200 steps during the training, the likelihood of the actions matching is relatively low considering the Random policy made random decisions. With mean and standard deviations of the Carry-out between the DQN and LSTM policies being relatively low, it is likely that not enough decisions matched or that the matched decisions were not as important as the unmatched decisions. For example, the DQN policy gave more Assertions when students logged back into a new session in the tutor, which most likely would not have matched the Random policies decisions. If those decisions were more important to student performance, then the effect would not be seen in the Random policy where similar decisions would not have occurred.
5.11 Exploring the Behavior of the Proactive Hint Policies

This section discusses the behavior of each proactive step-level hint policy. For both policies, students were clustered based on their pretest performance and how many Assertions and Hint Requests they had. Clustering was performed using K-means clustering, selecting the optimal cluster number based on the average silhouette width. Then, these clusters were split and performance on the posttest were compared. Lastly, I compare the differences and commonalities between the DQN and LSTM policies by approximating what features were most influential in each policy's decisions.

5.12 Behavior of DQN Policy

For the DQN group, the optimal number of clusters was found to be 4. Figure 5.8 shows the plot of the average silhouette width for the range of clusters explored.

![Optimal number of clusters](image)

Figure 5.8 Optimal number of clusters determined by test 1-5 clusters and comparing the average silhouette width. The number of clusters that was optimal for the students in the DQN group was 4.

To illustrate the quality of fit of the clusters, I used a method of plotting [Mae19] that creates a bivariate plot visualizing the clustering of the data using principal components (see Figure 5.9. As shown, the two components explain 76.63% of the point variability.

To examine what types of students the clusters captured, I plotted each student’s normalized features that were used for clustering. The data were normalized within the DQN group. Figure 5.9 shows the individual student series colored based on each cluster. I have labelled the clusters based on the prior proficiency (PP) features from the pretest (Pre-Total Time and Pre-Accuracy) and the hint features (Total Assertions and Hint Requests). The blue series labelled “High PP - Mid Assist” represents students with the highest pretest metrics that seem to have received a mid amount of Assertions compared to the other students and very few Hint Requests. The “High PP - Low Assist” are students with slightly lower pretest metrics than the blue group; however, much
lower levels of assistance (both Assertions and Hint Requests). The “Mid PP - High Assist” group is most prominently defined by the high levels of Assertions and Hint Requests and a range of prior proficiency mostly falling towards the mid range. The last cluster is the “Low PP - Mid Assist” group with the lowest prior proficiency metrics and a mid amount of assistance (both Assertions and Hint Requests) compared to the other clusters.

To determine whether any of these clusters prevailed, I looked into the Improvement and Relative Score on the posttest for each cluster (see Table 5.4). The Improvement Score shows the students Low PP - Mid Assist group with the highest Improvement Score, as expected as those students have the most room to improve upon. Interestingly, the Mid PP - High Assist group made almost identical improvements and ended up with the highest Relative Score on the posttest out of all of the clusters. Looking at the amount of assistance each group received, the higher the assistance indicated the higher performance on the posttest and the incoming proficiency did not have as much of a factor on the final performance.
5.13 Behavior of LSTM Policy

For the LSTM group, the optimal number of clusters was found to be 3. Figure 5.11 shows the plot of the average silhouette width for the range of clusters explored.

To illustrate the quality of fit of the clusters, I used the same method described above in Section 5.12 for the LSTM clusters [Mae19] (see Figure 5.12. As shown, the two components explain 74.29% of the point variability.

To examine what types of students the clusters captured, I plotted each student’s normalized features that were used for clustering. Figure 5.13 shows the individual student series colored based on each cluster. I have labelled the clusters based on the prior proficiency (PP) features from the pretest (Pre-Total Time and Pre-Accuracy) and the hint features (Total Assertions and Hint Requests). The blue series labelled “High PP - Mid Assist” represents students with the highest pretest metrics that seem to have received a mid amount of Assertions and very few Hint Requests compared to the other students. The “Mid PP - High Assist” group is also most prominently defined by the high levels of Assertions and Hint Requests and a range of prior proficiency mostly falling towards the mid range very similar to the green DQN cluster. The last cluster is the “Low PP - Mid Assist” group with the lowest prior proficiency metrics and a mid amount of assistance (both Assertions and Hint Requests) compared to the other clusters. These clusters follow similar patterns as the DQN clusters; however it is important to note that the normalized values are different for each plot (i.e. a value of

Figure 5.10 Each student in the DQN group is plotted based on their Pretest Total Time, Pretest Accuracy, Total Assertions Received, and total Hint Requests. Lines are colored according to their cluster assignment.
Figure 5.11 Optimal number of clusters determined by test 1-5 clusters and comparing the average silhouette width. The number of clusters that was optimal for the students in the LSTM group was 3.

Figure 5.12 Bivariate plot visualizing the clusters based on two principal components for the LSTM group.

0.8 for Total Assertions in the LSTM group is not the same as a 0.8 value in the DQN group). Table 5.2 shows that the LSTM group received less hints (on average 48 total compared to the DQN group with on average 58 total).

To determine whether any of these clusters prevailed and to see if these clusters showed similar impacts as the closely related DQN clusters, I looked into the Improvement and Relative Score on the posttest for each cluster (see Table 5.5). For the LSTM clusters, the amount of assistance did not have the same impact as seen for the DQN clusters in Table 5.4. Instead, prior proficiency had the largest impact on their Improvement Relative Score, as is normally the case. The Low PP - Mid Assist students had the lowest Relative Score and highest Improvement score, because they start off lower and have more room to improve compared to higher proficiency students. Whereas, the High PP - Mid Assist students have less room to improve, resulting in the lowest Improvement Score and highest Relative Score. These results indicate that the LSTM was not able to provide effective help for lower performing students.
Figure 5.13 Each student in the LSTM group is plotted based on their Pretest Total Time, Pretest Accuracy, Total Assertions Received, and total Hint Requests. Lines are colored according to their cluster assignment.

Table 5.5 Improvement and Relative Score on the non-isomorphic posttest for each cluster within the LSTM group.

<table>
<thead>
<tr>
<th>LSTM</th>
<th>Improv. Score</th>
<th>Rel. Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster</td>
<td>n</td>
<td>Mean(SD)</td>
</tr>
<tr>
<td>High PP - Mid Assist</td>
<td>14</td>
<td>46(20)</td>
</tr>
<tr>
<td>Mid PP - High Assist</td>
<td>11</td>
<td>49(18)</td>
</tr>
<tr>
<td>Low PP - Mid Assist</td>
<td>10</td>
<td>59(14)</td>
</tr>
</tbody>
</table>

5.14 Discussion of Commonalities and Differences in Proactive Hint Policy Behavior

Although both the DQN and LSTM groups had similar types of students in each cluster, however, the performance of those clusters were different, with the DQN group showing that more assistance in the training indicated higher performance on the posttest and the LSTM group showing that prior proficiency played a large factor on their posttest performance. Both groups had clusters of students with high levels of proactive assistance and high levels of help-seeking behaviors. For the DQN policy, those students performed the best on the posttest within the DQN group. For the LSTM policy, those students performed average on the posttest within the LSTM group. To understand why the effect of assistance was different between the LSTM and DQN group, I looked at the distribution
of hints (Assertions and MSG hints) across the training levels of the tutor. Figure 5.14 shows the mean hints per level for Total Hints (Assertions and MSG combined) and total Assertions and total MSG hints. The DQN group's plot shows a relatively stable amount of hints given throughout the training levels, with a slight dip in the 4th level for all hint metrics. The LSTM group shows a peak in training level 2, the hardest level (based on years of historical student data indicating this level takes significantly longer than the other levels). Therefore, the most assistance from the LSTM policy was provided during the hardest levels, suggesting that the policy was providing hints when students were struggling on the more difficult problems. This result, in combination with how fast the LSTM group completed the training and the fewer amount of Assertions given compared to the DQN policy (shown in Table 5.2), provides support for that these hints were helping the students avoid struggle. Even though the LSTM policy may have been providing hints when the students were struggling in the difficult problems, the lower performance on the posttest indicates that the assistance may have taken away crucial learning opportunities. Considerations from both the assistance dilemma [Koe07] and the Zone of Proximal Development (ZPD) [Vyg78] focus on a need to provide assistance just the right amount of assistance. Considering the DQN group was provided with more assistance and still ended up doing better on the posttest, the amount of assistance does not seem to be the issue. However, another important consideration with both the assistance dilemma and the ZPD is providing assistance in such a way as to not interfere with natural active learning. Therefore, by focusing hints on the hardest steps, the LSTM policy may have interfered with the students learning and negatively impacted their performance.

To investigate what each policy's decisions may have been focused on, I generated an approximation of each policy's top 5 most important features using a decision tree based on the training data from each policy. Using decision trees to approximate Deep Learning algorithms, such as DQNs and LSTMs, is a common method to gain insight into a complex model's decisions [Fro17; Bas17]. The feature importance table (see Table 5.6) shows an approximation of important features generated by decision tree learning on the actions taken by the models on student data. The purpose of this analysis is to provide insight into how the model was making decisions on when to provide hints and what type of features were most heavily associated with that decision. The top five most important features of the decision tree provide insight into how each model may have been making decisions. The DQN uses features that are centered around the student's current circumstance: pNewSession - did they just log back in, sAssertionJustified - did they just justify an Assertion, pRuleDescription - how often are they looking at how to use a rule within this problem. Whereas the LSTM model focuses on the overall progress for the student: tProactiveHintCount - how many Assertions have they received up to this point, actionCount - how many times have they interacted with the screen, tAccuracy - what is their overall Accuracy, and tTime - how long have they been working in the tutor. By focusing on features based on the student's current circumstance, the DQN model performed better. New research has shown that including features that focus on a student's current circumstance improves student modelling and can allow the model to incorporate when a student may partially forget a skill [Ned15; Nag19; Qiu11]. Since students may log out and log in to the tutor as
Figure 5.14 The mean Total Hints received, Total Assertions given, and Total Msg Hints for the Control, DQN, and LSTM groups.

Table 5.6 Feature importance table for DQN and LSTM Policies Approximated by Decision Tree Learning

<table>
<thead>
<tr>
<th>DQN Features</th>
<th>Rank</th>
<th>LSTM Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>pNew Session</td>
<td>1</td>
<td>tProactiveHintCount</td>
</tr>
<tr>
<td>sAssertion Justified</td>
<td>2</td>
<td>actionCount</td>
</tr>
<tr>
<td>pRuleDescription</td>
<td>3</td>
<td>currentProblem</td>
</tr>
<tr>
<td>prevLevelAccuracy</td>
<td>4</td>
<td>tAccuracy</td>
</tr>
<tr>
<td>tTime</td>
<td>5</td>
<td>tTime</td>
</tr>
</tbody>
</table>

many times as they want, it can be expected that a student who logs out and logs back in a few days later will have lost some sharpness to their skills compared to a student working through the tutor all at once. The DQN model also considered the student’s own help-seeking behavior as seen with the pRuleDescription features, which indicates the students are trying to refresh themselves on how to use a specific rule. By focusing on the student’s current behavior instead of an overall view of the behavior (as indicated by the LSTM’s important features), the DQN model may capture when students may benefit from assistance better compared to the LSTM model.

One reason for this difference in what each policy considers important features may be the training data that was used to induce each policy. The DQN policy was trained on data that used a random, but reasonable policy to provide Assertions to students. Whereas, the LSTM policy was trained on data with multiple policies: the same random, but reasonable policy, a student-based policy which considered Hint Requests as Assertions, and two learned step-level hint policies (one being the DQN). The inclusion of these data was intended to give a wider range of situations where
Assertions were given due to each policy having a differing range of Assertion frequency. However, this may have negatively influenced the model training by producing too much noise or causing bias from the learned policies. Furthermore, the training data also contained new rewards based on inferring the rewards [Azi] rather than a simple calculation to distribute them and removing the tiered structure that boosted the reward for higher performing students. This may have also influenced the training and eventual performance.

As mentioned in Section 5.6, in deciding to use the LSTM architecture, I trained a new policy on this same training data using the same DQN architecture (described in Section 5.3.3 as deployed in the Fall 2019 and saw the LSTM architecture outperform it in terms of ECR. Therefore, to further explore whether the LSTM architecture was the reason for the difference in decisions, I used the decision tree method of approximating why the policy made decisions. The new DQN trained on the LSTM’s training data (with a lower converged ECR) focused on a similar set of important features as the Fall 2019 DQN: 1) currentProblem, 2) pNewSession, 3) actionCount 4) sActionCount - how many interactions have occurred on this step 5) sRuleDescription - how many times a student viewed a rule's description during this step. Therefore, with both DQNs focusing on features based on the student’s current circumstance, the LSTM architecture may be a more likely reason for the worse performance. Furthermore, unlike the LSTM which provided few Assertions (Mean(SD): 35(15)), the new DQN provided a high amount of Assertions (Mean(SD): 80(34)), which is more similar with the deployed DQN’s high amount of Assertions (Mean(SD): 54(13)) and would be consistent with the pattern seen in Section 5.12 where higher amounts of Assertions were associated with better performance.

Other educational research has found that despite theoretical results showing that the ECR outperforms baseline models, the same results are not always realized [Azi]. In their study, two reinforcement induced policies on whether to give problem solving or worked examples showed that the models outperformed the random baseline. However, the empirical results of the study show that the models performed similar to the random policy. I recommend evaluating policies not only based on traditional methods, such as ECR, but consider using interpretable models or ways of approximating models to evaluate how a model is making its decision.

\section{Discussion of Overall Results}

Based on the results from Table 5.2, Table 5.3, Figure 5.14 and Table 5.6, the LSTM policy seemed to provide hints on the most difficult steps of the tutor, reducing the time taken to solve the training problems. However, by focusing the hints on the most difficult steps, crucial learning opportunities may have been taken away resulting in the overall worse performance in the posttest. On the other hand, Table 5.2, Figure 5.7, and Table 5.3, which show the performance-related metrics for the tutor, show the DQN group performed better on the training than the control, taking a similar but slightly faster time to solve the training with significantly fewer steps and outperformed the LSTM policy on the posttest performance measures. Furthermore, Table 5.4 shows regardless of incoming
proficiency, the assistance provided was able to bring lower performing students up to a similar level by the posttest as students with higher incoming proficiency. This effect was not seen in either the LSTM or the Control groups. Looking at Table 5.6, the DQN’s decisions were most influenced by the student's current behavior and not as much their overall performance up to that point and Table 5.14 shows a relatively comparable number of hints provided in each training level. This distinction allowed the DQN to provide a higher amount of Assertions without hindering the students’ learning. In comparison to the Control, the DQN group were able to complete the training slightly faster with fewer steps without hindering their posttest performance. Therefore, I believe that with more focus on features surrounding the current progression of the student considering whether they are reentering the tutor, as focused on in [Ned15; Nag19], the need for assistance can be better determined and overall performance improved. In research by Nagatani et al., they incorporated “forget”, a metric to consider lag time between students interactions with the tutor, into a knowledge tracing model and showed that it increased the predictive performance of the model.

5.16 Limitations

This section discusses limitations of this work and the implications of these limitations my conclusions.

As discussed in Sections 5.3.3 and 5.6, both policies were initially trained using 50,000 iterations and reduced to a number near where previous models converged. Although unlikely, the models may have not fully converged due to reducing the number of iterations. However, for both policies the ECR convergence was stable for approximately 5,000 iterations before training was terminated.

Due to the differing rewards and training data between the DQN and LSTM policies, a direct comparison of the architectures would be misleading. I cannot say that the DQN or the LSTM is more suited for approaching this type of problem, based on this research. However, the results surrounding how the policies behaved, in terms of when Assertions were given, provides insight into how number of hints and timing affects student performance. Furthermore, using the old DQN architecture on the new training data, I found the new DQN’s (not deployed in a formal study) important features and amount of Assertions given more similar to the old DQN, which suggests that the LSTM architecture might be the reason for the differing behavior. However, the new DQN has not been deployed and the actual effect on the student’s performance is unknown.

Another limitation of this work is that these policies all use next-step hints and delivered via Assertions (in-workspace hints). Other types of hints or hints delivered in another fashion may produce different results.

5.17 Contribution

This chapter contributes a study comparing the effects on performance of two Reinforcement Learning step-level hint policies to a control with hints on-demand only. The results show that the
RL hint policies were able to improve some of the performance metrics in the training. However, the LSTM policy, which focused its decisions on the student’s overall progression, providing hints during the hardest part of the training, hindered learning and resulted in the worst performance out of all the groups. The DQN policy focused on features surrounding the student’s current performance, including when students were reentering the tutor. The DQN policy performed better on the training and comparably to the Control, with the lowest performing students benefiting the most. Even though the LSTM was theoretically evaluated to perform better than the Fall DQN architecture, this was not realized in the empirical results. Therefore, using interpretable methods to induce policies or in addition to evaluate policy decision is recommended. This study contributes insight into what types of features are beneficial to providing effective step-level hints. Furthermore, these results indicate that with more attention to creating policies that focus on providing assistance based on the student’s current situation with less focus on their overall progress, students with the lowest incoming proficiency can achieve similar results on the posttest compared to the initially high proficiency students.

This work contributes:

- Methods for generating step-level rewards and policy induction in a tutoring system with an open-ended problem space
- Insight into what cases a proactive hint policy may succeed or fail at providing better learning outcomes for students with different educational needs
- What features are most important in making effective decisions on when to give hints
- Practical insight that informs future proactive hint policy studies and future system design in Deep Thought
6.1 Contributions

My work offers the following contributions:

- Insight into how existing data-driven methods for next-step hints can be used to generate higher-levels of hints (RQ1)
  
  Design suggestions for scaffolding higher-level hints for future studies

- Greater understanding of how different hint types affect learning outcomes, especially between high and low proficiency students (RQ1)

- Insight into how different levels of autonomy regarding assistance affects learning outcomes (RQ2)

- Insight into reward generation and policy induction methods for creating a proactive hint policy (RQ3)
BIBLIOGRAPHY


[Azi] Azizsoltani, H. et al. “Unobserved is not equal to non-existent: using Gaussian processes to infer immediate rewards across contexts”.


[Swe06] Sweller, J. “The worked example effect and human cognition.” Learning and instruction (2006).


