Extending the Hint Factory for the Assistance Dilemma: A Novel, Data-driven Help-Need Predictor for Proactive Problem-solving Help

Mehak Maniktala
North Carolina State University
mmanikt@ncsu.edu

Christa Cody
North Carolina State University
cncody@ncsu.edu

Amy Isvik
North Carolina State University
aaisvik@ncsu.edu

Nicholas Lytle
North Carolina State University
nalytle@ncsu.edu

Min Chi
North Carolina State University
mchi@ncsu.edu

Tiffany Barnes
North Carolina State University
tmbarnes@ncsu.edu

Determining *when* and *whether* to provide personalized support is a well-known challenge called the assistance dilemma. A core problem in solving the assistance dilemma is the need to discover *when* students are unproductive so that the tutor can intervene. Such a task is particularly challenging for open-ended domains, even those that are well-structured with defined principles and goals. We present a set of data-driven methods to classify, predict, and prevent unproductive problem-solving steps in the well-structured open-ended domain of logic. This approach leverages and extends the Hint Factory, a set of methods that leverages prior student solution attempts to build data-driven intelligent tutors. We present a **HelpNeed** classification that uses prior student data to determine *when* students are likely to be unproductive and need help learning optimal problem-solving strategies. We present a controlled study to determine the impact of an Adaptive pedagogical policy that provides *proactive* hints at the start of each step based on the outcomes of our HelpNeed predictor: productive vs. unproductive. Our results show that the students in the Adaptive condition exhibited better training behaviors, with lower help avoidance, and higher help appropriateness (a higher chance of receiving help when it was likely to be needed), as measured using the HelpNeed classifier, when compared to the Control. Furthermore, the results show that the students who received Adaptive hints based on HelpNeed predictions during training significantly outperform their Control peers on the posttest, with the former producing shorter, more optimal solutions in less time. We conclude with suggestions on how these HelpNeed methods could be applied in other well-structured open-ended domains.

Keywords: assistance dilemma, problem solving, unproductivity, efficiency, unsolicited hints, adaptive support, data-driven tutoring, propositional logic, help appropriateness

1. INTRODUCTION

Intelligent Tutoring Systems (ITSs) provide individuals with adaptive feedback and hints, improving learning (VanLehn, 2011). Studies suggest that hints, when provided appropriately, can augment students' learning experience (Bunt et al., 2004; Puustinen, 1998) and improve their performance (Bartholomé et al., 2006). However, researchers often find that students display poor help-seeking behavior (Aleven et al., 2006; Price et al., 2017): some abuse hints to expedite problem completion, and some avoid seeking help when they are in need (Aleven and Koedinger, 2000; Price et al., 2017). To ensure learning despite student help avoidance, several ITSs provide unsolicited assistance (Arroyo et al., 2001; Murray and VanLehn, 2006; Kardan and Conati, 2015). However, determining *when* and *whether* to provide proactive assistance, i.e., unsolicited help in anticipation of future struggle, is particularly challenging in open-ended domains where there are many possible correct solutions. More generally, it belongs to a well-recognized challenge in the domain of ITSs, called the assistance dilemma.

The assistance dilemma is a trade-off between giving and withholding information to achieve optimal learning (Koedinger and Aleven, 2007). A core problem of the assistance dilemma is the need to discover *when* and *whether* students are unproductive so that the tutor can intervene. Several studies have explored ways to determine the timing of assistance, as well as scaffolding in open-ended domains to improve learning (Fossati et al., 2015; Ueno and Miyazawa, 2017; Borek et al., 2009; Kardan and Conati, 2015) and prevent student failure in exams (Merceron and Yacef, 2005). While some researchers have explored the generalizability of such approaches (Fratamico et al., 2017; Bunt and Conati, 2003), determining when to provide assistance is still a challenging task for most open-ended domains, particularly because of differences in domains and learners (Klahr, 2009; McLaren et al., 2014).

Open-ended domains can be ill-structured, where problems do not have a clear goal, set of operations, end states, or constraints; or they can be well-structured, where problems have a clear goal, end states, or constraints. In this study, we seek to determine *when* to provide unsolicited hints in a logic tutor to solve the assistance dilemma. While logic problems are well-structured in that they contain all the information needed to solve the problem and there are well-defined algorithms that students can use to solve them, they are open-ended in the sense that they have many possible solutions that can all be correct. Our approach builds upon the Hint Factory, a method for hint generation (Barnes et al., 2011). We extend the Hint Factory algorithm to define step-level productivity and define *HelpNeed*, to classify steps as unproductive or needing help. Based on this HelpNeed definition, we explore several machine learning methods to build a HelpNeed predictor and design an Adaptive pedagogical policy to proactively provide hints when the next step is predicted to need help. We conduct a controlled study to investigate the impact of Adaptive hints on student posttest performance, as well as the frequency of possible appropriate and inappropriate help during training when compared to students in the Control group with access to on-demand hints but no Adaptive hints.

Our results showed that students in the Adaptive condition have significantly higher posttest performance than those in the Control condition, as measured by solution optimality and time, and the former also exhibit better hint usage behaviors while training on the tutor in that they had fewer HelpNeed steps, a lower possible help avoidance, and higher possible help appropriateness (a higher chance of receiving help when it was likely to be needed) when compared to the Control. Our main contributions can be summarized as:

- This work presents a novel, data-driven HelpNeed classification model to determine un-

productivity for steps and a predictor for when unproductive steps might occur in well-structured, open-ended, multi-step problem-solving domains.

- We develop a new Adaptive pedagogical hint policy that uses a HelpNeed predictor to provide proactive hints when they are likely to be needed and investigate its impact on performance and appropriate help in a controlled study.

In the remainder of the paper, we first discuss the related work that establishes the need for new ways to solve the assistance dilemma in well-structured open-ended problem solving. Next, we introduce the Deep Thought logic tutor, the context for this work. Then, we present our novel extensions to prior work on data-driven intelligence for tutoring to build HelpNeed classification models. We then compare machine learning methods to derive a HelpNeed predictor tuned to proactively identify steps where students are likely to need help. We present an Adaptive hint policy using the HelpNeed predictor and describe our controlled study and its impact on posttest performance and whether it helped solve the assistance dilemma during training.

2. RELATED WORK

2.1. HELP SEEKING

Aleven et al. (2006) defined non-optimal help-seeking behaviors, including *help avoidance*, where students can benefit from seeking help but choose not to, and *help abuse*, where students excessively use help in situations where they could solve problems without assistance. While some studies aim to solve the problem of help avoidance by regulating students' help-seeking behavior (Roll et al., 2011; Aleven et al., 2006), some employ unsolicited hints as we propose here (Arroyo et al., 2001; Murray and VanLehn, 2006; Rus et al., 2017). Arroyo et al. (2001) observed that unsolicited hints could improve learning gains for students with low prior knowledge. Murray and VanLehn (2006) found that proactive help reduced frustration and saved students' time when they were struggling.

2.2. ASSISTANCE DILEMMA

The *assistance dilemma* has been a well-known challenge in providing unsolicited hints. Providing assistance may reduce frustration and save students' time but may lead to shallow learning or a lack of motivation to learn by oneself. On the other hand, withholding information can encourage students to learn by themselves but may lead to frustration and wasted time (Koedinger and Aleven, 2007; McLaren et al., 2014). Researchers have investigated data-driven approaches to address this dilemma and create adaptive assistance both for *on-demand* hints provided on request (Anohina, 2007; Price et al., 2017; Wood and Wood, 1999) and *proactive* unsolicited hints (Kardan and Conati, 2015; Ueno and Miyazawa, 2017). Koedinger and Aleven (2007) worked towards addressing this dilemma in a cognitive tutor by initially withholding information about problem solutions and steps, and then interactively adding information, only as needed, through yes/no feedback, explanatory hints, and dynamic problem selection. Several studies have investigated approaches to resolve the assistance dilemma by modeling student behavior (Murray et al., 2004; Ueno and Miyazawa, 2017; Conati et al., 2002; Kock and Paramythis, 2010). One group created a tutoring policy within the DT Tutor that applied decision theory to make its choice about whether to give a hint (Murray et al., 2004). In a study conducted by Ueno and Miyazawa

(2017) in a programming tutor, they modeled student performance using Item Response Theory to determine optimal hint scaffolding for providing assistance proactively upon mistakes. The authors discovered that hint scaffolding with predicted probabilities for the learners' success to be 0.5 provides the best learning performance. We recently added unsolicited hints to our tutor, improving post-test performance for students with low prior knowledge (Maniktala et al., 2020). However, the posttest performance of students with high prior knowledge was negatively, but not significantly, correlated to how often they used the unsolicited hints, suggesting the potential for negative effects from receiving too many hints. Therefore, in this work we seek to develop adaptive support that can determine *when* to proactively provide hints based on students' aptitude or performance, as in other ITSs (Bunt and Conati, 2003; Kardan and Conati, 2015).

The assistance dilemma is particularly challenging to address in open-ended domains that deal with ill-structured problems (McLaren et al., 2014; Borek et al., 2009) as well as well-structured problems (Ueno and Miyazawa, 2017). McLaren et al. (2008) conducted a study to explore the assistance dilemma in an ill-structured chemistry tutor with inquiry learning using three levels of problem-level assistance: high (worked examples), mid-level (less assistance than worked examples), and low (untutored problem solving). They found mid-level assistance to lead to better learning than either low- or high-level assistance. In another study, Borek et al. (2009) investigated the optimal amount of step-level assistance in a discovery-oriented virtual chemistry laboratory (VLAB) and found that students learned the conceptual tasks better with a mid-level assistance approach of hints and feedback. Further, their results suggested that assistance should be given only when students are far off track.

Kardan and Conati (2015) used clustering and mining association rules to model when to provide unsolicited adaptive support (hint timing, content, and interface) in an ill-structured exploratory interactive simulation environment for teaching constraint satisfaction problems. While the students who received adaptive support did not have improved task performance, they learned significantly more than those who did not receive adaptive support. Fratamico et al. (2017) applied Kardan and Conati's 2015 framework to an electronic circuits simulator and found that it successfully classified students into groups of high and low learners. Mostafavi et al. (2015) similarly applied machine learning to form data-driven proficiency profiles used in problem selection and showed that it reduced the time taken in a previous version of the Deep Thought logic tutor. Conati et al. (2002) presented several evaluations of a Bayesian Network used to model student behavior in a well-structured physics tutor that, in part, is used to determine when unsolicited mini-lessons should be provided to students. They also described how these modeling tasks involve a high level of uncertainty when students are allowed to follow various lines of reasoning and are not required to explicitly show their reasoning. As mentioned above, a study conducted by Ueno and Miyazawa (2017) in a well-structured programming tutor used Item Response Theory to determine optimal scaffolding for unsolicited hints and discovered that providing scaffolding associated with 50% probability for students' success led to the best student outcomes.

2.3. UNPRODUCTIVE BEHAVIOR

Several studies have used the term "unproductive" to refer to undesirable behavior during training (Kai et al., 2018; Botelho et al., 2019; Park and Matsuda, 2018). For example, Beck and Gong (2013) define unproductive persistence or "wheel-spinning" based on whether or not a student achieved mastery (three correct problems) in ten problem attempts. Their definition of

unproductivity has been used in recent studies to predict when an intervention can help students by distinguishing between productive and unproductive behavior using decision trees (Kai et al., 2018) and recurrent neural networks (Botelho et al., 2019). However, this definition of problem-level and problem-completeness based productivity is not suitable for our objective of guiding students toward efficient problem-solving strategies at a more granular step-level, specifically in domains where problems can have several solution paths.

In a study on an open-ended and ill-structured inquiry-learning program, McLaren et al. (2014) defined unproductive events as student actions that are unlikely to advance understanding. Similarly, we identify unproductive problem-solving steps that are not likely to advance student's problem-solving strategies. However, our definition is based on time and solution length/optimality rather than domain-specific productivity models.

In many multi-step open-ended but well-structured problem-solving domains, shorter solutions are considered to be more optimal, and solving problems in less time reflects both learning and fluency (Mayer, 1992; Smith, 2012; Yacef, 2005; Cen et al., 2007). We use these basic assumptions about time and solution length to design a data-driven approach to model productivity on problem-solving steps, extending the Hint Factory, a data-driven approach for hint generation. This approach uses prior students' data to assign scores to problem-solving states (Stamper et al., 2008; Barnes et al., 2011). A core insight of this paper is that we can similarly use student data to score productive problem-solving steps without the need to model the domain.

Some studies have observed that students game educational systems to elicit unsolicited hints to complete problems faster (Baker et al., 2009; Baker et al., 2006; Murray and VanLehn, 2005). Therefore, we limit the amount of information provided in our unsolicited hints, presenting them as partially-worked steps.

3. TUTOR CONTEXT

The research in this paper is conducted using Deep Thought, a data-driven intelligent logic tutor where students practice constructing formal propositional logic proofs in discrete math courses, with 200-350 students per semester, since 2008 (Stamper et al., 2008; Barnes and Stamper, 2010; Cody and Mostafavi, 2017; Mostafavi and Barnes, 2017). Each fall (f) or spring (s) semester, we use stratified sampling on a pretest to assign students to conditions. Historical data exist for 2017, 2018, and 2019. This study uses data from f17, s18, f18, and s19; the HelpNeed predictor uses f18 and s19; and the controlled study with the proactive hint policy was in f19. Finally, the results of the controlled study are analyzed comparing data from f18R, s19R, along with f19, where R represents a subset of the population that received *randomly* administered frequent unsolicited hints[1](Maniktala et al., 2020). Each section provides more details on the specific datasets used.

Figure 1 shows the Deep Thought interface for a single problem, where students construct their proofs in the *workspace*. Each proof problem starts with a set of given logic statements with a conclusion to derive. Each statement is represented by a *node*, which students can click to select and apply a logic rule to derive a new node. Each node derivation consists of two parts: the *justification* and the derived statement. The justification is a set of 1-2 existing nodes and the rule applied to them, and the derived statement is the result. The tutor checks each step's justification and derived statement to provide immediate feedback on the correctness of their

---

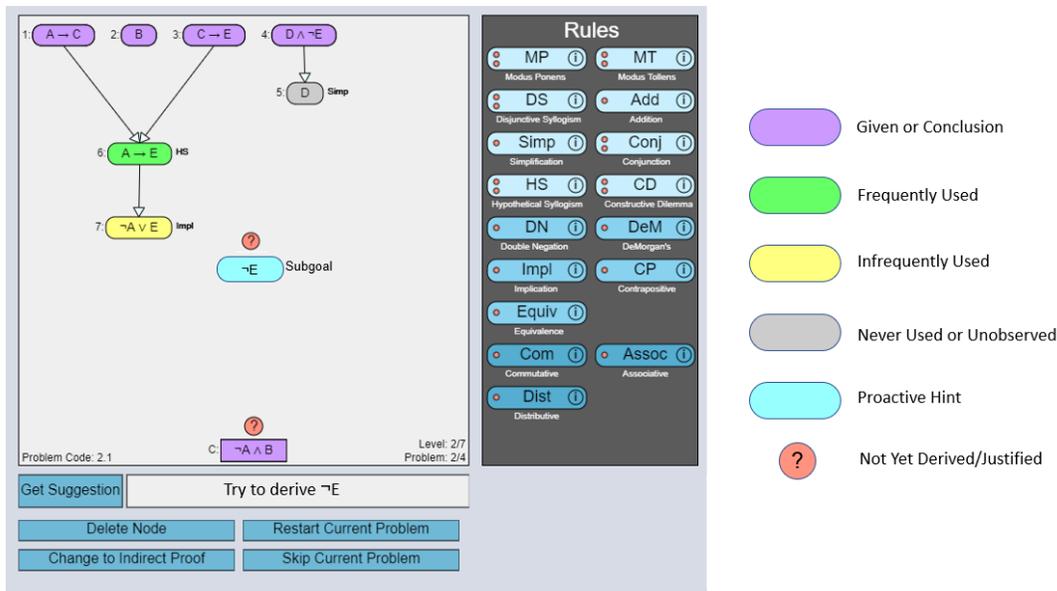[1]With a constraint that three unsolicited hints cannot be given in a row

Figure 1: Deep Thought interface: *Workspace* (left) with the provided givens (top) and the conclusion (bottom), domain rules (middle), the *'Get Suggestion' hint button*, and *hint message box* (bottom-left). The color-coding is explained on the right (provided to students in an infobox).

logic rule applications. In case of rule application errors, students receive a notification of the mistake via popup, but the attempted incorrect node is deleted from the workspace. When a statement is correctly justified and derived, a correct step is recorded, and its node appears. The nodes are colored according to their *use-frequency*[2] in historical student solutions. A frequently-used node is colored green, an infrequently-used node is colored yellow, and a node that has never been used or has not been observed in the historical data is colored gray (see Figure 1). Hints are available during Deep Thought's training section using the Hint Factory (Stamper et al., 2008). Students can receive *on-demand* hints below the workspace by clicking the "Get Suggestion" button, and they can also receive tutor-initiated *proactive* hints that appear within the workspace (see Figure 1).

A snapshot of the workspace at any given time is called the problem-solving *state*, and transitions between states are called *steps*. When a problem is complete, students have connected the given statements to the conclusion.

4. HELPNEED CLASSIFICATION AND PREDICTOR

In this section, we present our methods to determine when students need help learning efficient problem-solving strategies. We seek to identify unproductive steps where students need help so that we can predict them and provide proactive hints with the aim to prevent them. To do this, we first define our HelpNeed classification based on prior literature on problem solving and later present our HelpNeed predictor to detect the need for help at the start of each step.

---

[2]A node is said to be *used* if its removal from the solution would make the problem incomplete.

4.1. HELPNEED CLASSIFICATION

We designed the HelpNeed classification based on the literature on what behaviors may demonstrate a detectable need for help during problem-solving. Based on the literature, learning is reflected in both correctness and/or time (duration) in problem solving (Corbett and Anderson, 1994; Kai et al., 2018; Beck and Gong, 2013). However, since multi-step problem-solving involves several steps in a row, we cannot directly use correctness as a criterion. Therefore, we create new methods to detect efficiency, which is a proxy for eventual correctness and optimality in multi-step problem solving, as discussed in detail in section 4.1.2.

Table 1 shows our HelpNeed classification, which is based on a step's duration and efficiency. Duration is the time taken by a student to carry out a step, described in detail below in 4.1.1. Efficiency is our unique extension of the Hint Factory that uses prior data to assign numeric values that reflect a step's quality and how well it promotes progress to a good solution, described in 4.1.2.

The HelpNeed classification scheme is designed to identify steps that reflect suboptimal strategies, i.e., steps that unnecessarily increase the solution length and/or problem-solving time. We first define steps that do not demonstrate HelpNeed. For our first two categories, efficient steps, which lead to shorter, more optimal solutions, are considered productive irrespective of duration, with short times representing (1) *Expert-like* behavior and longer times classified as (2) *Strategic*.

For our third category, we classify a single quick but inefficient step, i.e., a plausible guess, as an (3) *Opportunistic* step. Research suggests that we should provide students some opportunities to guess throughout the learning process in semi open-ended domains such as math (Capraro et al., 2012; Polya, 2004). Polya (2004) describes guess-and-check to be a strategy when students apply *plausible reasoning*, where the goal is to distinguish a more reasonable guess from a less reasonable one. In a beneficial use of guess-and-check, the student *checks* whether they are moving closer to the goal and adjusts their strategy in the next step. Therefore, we classify a single quick but inefficient Opportunistic step as one where help is *not needed*. To transfer the idea of Opportunistic steps to other domains, it will be important to determine what combination

Table 1: Defining HelpNeed using step Efficiency and Duration

Classification	Behavior	Description
No HelpNeed	Expert-like	A quick efficient step; demonstrating mastery
	Strategic	A long efficient step; taking longer on an expert-like step
	Opportunistic	A single, quick inefficient step
HelpNeed	Far Off	Maximum number of inefficient steps in a sequence and/or multi-step duration for inefficient steps before intervention is desired In our tutor: consecutive quick but inefficient steps
	Futile	A long inefficient step; taking too long on a step that does not help make progress

of efficiency and duration constitutes a plausible guess, or Opportunistic step, where help is not needed, in contrast to the remaining categories below.

Our fourth (4) *Far Off* category is meant to represent one in a series of steps that demonstrate a lack of strategy, or being far off-track, and needing help. A study by Borek et al. (2009) in an ill-structured open-ended domain suggests that students who are far off track in problem-solving can benefit from a hint. Likewise, in the well-structured, open-ended domain of linked lists, Fossati et al. (2015) intervened when students made steps that prevented problem completion. Research suggests that repeated guessing when faced with difficulty can be a significant stumbling block in developing effective learning strategies, and feedback at this stage can help students (Kinnebrew et al., 2014).

Based on the meaning of Far Off in these studies, we have defined Far Off steps as those that demonstrate that the student is not reflecting on whether their consecutive steps are leading toward an efficient solution. In our tutor, evidence of being Far Off can be seen when a student quickly performs sequential inefficient steps. Therefore, we define Far Off steps as the second and later steps in a series of quick, inefficient steps. In a sequence of three quick inefficient steps, the first would be Opportunistic - with no need for help, and the second and third would be classified as Far Off - with help needed. In this tutor, where the mean student solution length is 10 steps, we determined that 2 steps are about 20% of the problem, and this is long enough to intervene. The definition of Far Off steps could be adjusted for other domains to align with being far off track, using a combination of duration and efficiency, for example by setting a maximum time or number of inefficient steps that can elapse without an efficient step being taken.

Next, researchers use the term "wheel-spinning" or unproductive struggle to refer to the lack of mastery in a timely manner (Beck and Gong, 2013; Kai et al., 2018). In our final category, a student who has spent significant time but has derived a step that is inefficient may be engaging in unproductive struggle, and we classify this as a (5) *Futile* step where a student needs help. In the next two sections, we describe the step duration and efficiency components of HelpNeed classification.

4.1.1. Step Duration

A step duration is said to be *Long* if it is carried out in a time greater than 75th percentile of step time for that problem and *Quick* otherwise. These values are computed using historical per-problem data averaged across solutions by students ($N = 437$) to account for the longer step times in more difficult problems. For example, a step is Long in a difficult problem of our logic tutor when the step time is greater than 5.48 min, whereas a step is Long in one easy problem of our tutor if the step time is greater than 2.95 min.

4.1.2. Step Efficiency

We define *step efficiency* as a data-driven measure of how much a student's most recent step contributes to an efficient (short) solution. We explore four new step efficiency metrics defined using the combination of the *quality* of each state (local or global) and the *progress* made in a step (absolute or relative). We first present the Hint Factory, and then how we extended it to define step efficiency.

4.1.2.1. THE HINT FACTORY

The Hint Factory is a method for generating hints in well-structured open-ended domains (Barnes et al., 2008). In this approach, historical student solutions are used to form Markov decision processes (MDPs) from interaction networks (Eagle et al., 2015), where vertices are observed student problem-solving states (snapshots of their on-going or completed proof), and edges are problem-solving steps, i.e., a transition between states. The Hint Factory uses value iteration, a classic reinforcement learning technique, given in Equation 1, to assign an *expected value* $V(s)$ to each state s, where $R(s)$ is the state's reward, γ is the discount factor, and $P_a(s'|s)$ is the proportion of the observed solutions in state s that lead to state s' using the action a. An action in this equation is what causes a transition between states; therefore, action a represents a problem-solving step that can be carried out either by correct rule applications or by node deletions. For example, in Figure 3, the arrows represent actions (steps). More specifically, an action a to derive node $A \to E$ from the top start state is a tuple consisting of the rule, the node list the rule applies to, and the resulting logic statement (HS, $A \to C, C \to E, A \to E$). Note, while the Bellman Equation 1 is used for stochastic environments in reinforcement learning, these actions (steps) are carried out deterministically in our tutor. The original Hint Factory used this equation not to account for the uncertainty in carrying out steps but rather to take into account the probabilities of transition between each pair of states. This ensures that the value of each state is dependent on the probabilities of transition at each successive step. In the Hint Factory, a large reward is set for the problem-completion or *goal states* (100), penalties for incorrect states (10), and a cost for taking each action (1) (Barnes et al., 2011). A non-zero cost on actions causes the MDP to penalize longer solutions.

$$V(s) := R(s) + \gamma \max_a \sum_{s'} P_a(s'|s)V(s') \tag{1}$$

### 4.1.2.2. STATE QUALITY - EXTENDING THE HINT FACTORY

In this section, we leverage the Hint Factory approach to generate two quality metrics that determine the expected values for each observed problem-solving state. The first metric of state quality is our prior work on the Hint Factory, which we label as *local quality value* (LQV) with local rewards (LR). Local quality provides insights about how far a state is from the closest goal state, weighted by the probabilities of transitions, but it cannot provide information about whether the state is on an efficient path to a solution.

$$GQV(s) := GR(s) + \gamma \sum_{s'} P_a(s'|s)GQV(s') \tag{2}$$

Global Quality. We devised a novel, data-driven global quality value function, GQV in Equation 2, to give higher values to states on efficient solution paths. Equation 2 sums $GQV(s')$ over all states s' reachable from s, weighted by $P_a(s'|s)$, taking into account all future actions from a current state, rather than just the one with the max expected value. The global rewards GR are identical to LR for errors and actions but are different for goals, giving shorter, more efficient solutions higher rewards. The global reward $GR(g)$ for each goal state g on a problem is $GR(g) = 100 - p * \delta(g)$ where $\delta(g)$ is the difference between the solution length of g and that of the shortest solution, and p is a penalty for longer solutions. We set $p = (100 - 80)/\delta_{median})$ where δ_{median} is the difference between the median and shortest solution lengths for each problem because median student solution lengths are assigned a global reward of 80. Our

intention in setting this value of 80 is so that the student's performance with a median solution length represents a low B grade. Note, the relationship between the action a and follow-up state s' is the same for both local and global state values, because performing the action a when in state s deterministically leads to the next state s' based on our definition of actions. And, finally, the proof of convergence for the modified value iteration Equation 2 is given in Appendix A.

Figure 2: Three Solutions with Varying number of steps for a logic problem in Deep Thought
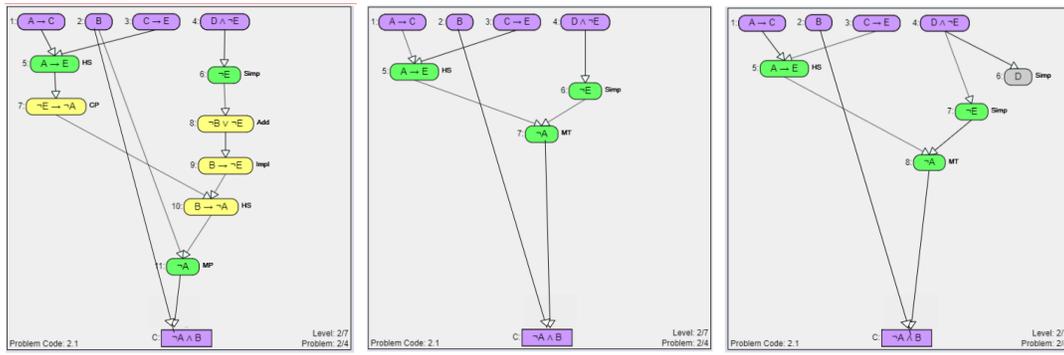
(a) Trajectory T_{long}: Eight steps and all used nodes

(b) Trajectory T_{short}: Four steps and all used nodes

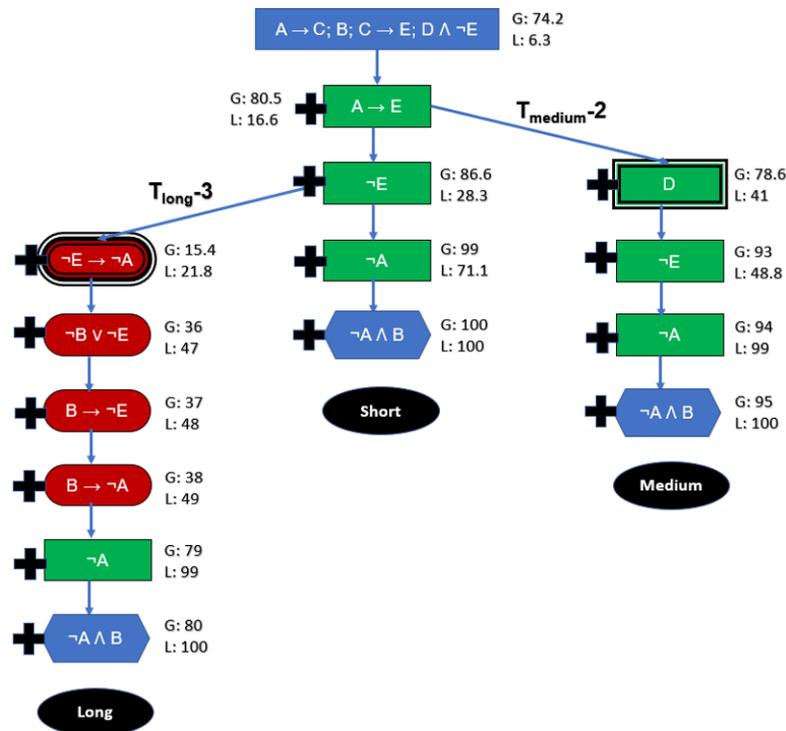(c) Trajectory T_{medium}: Five steps and four used nodes

Figure 3: Illustration for the concepts of State Quality and Productive steps in 3 trajectories T_{short}, T_{medium}, and T_{long}

We now demonstrate the differences between local and global quality metrics using three solution trajectories (series of steps) of varying solution lengths: T_{short}, T_{medium}, and T_{long} in Figures 2 and 3. T_{short} is the shortest solution (four steps), with all nodes (derived problem-solving statements) used. Recall as defined in section 3, a node is said to be *used* if it contributes towards deriving the conclusion of the problem. T_{short} has four used steps, T_{medium} has five steps with one unused node D, and T_{long} has eight steps and all nodes used.

We generated interaction networks to determine the quality values for each problem using our historical data for $N = 796$ students. Figure 3 shows the quality values for the three trajectories in Figure 2. The start state in Figure 3 consists of the four given statements (the topmost state). Arrows between states represent steps, i.e., a transition between states by rule applications. Non-start states are represented by a +(XYZ), where XYZ is the new statement derived in a step. The start state has a high global quality but low local quality. The starting state's global quality is high because all efficient paths contain it, but its local quality is low because it is probabilistically farther away from goals than any other state in the figure. The local quality for states that are only found in incomplete attempts is lower than that for the start state. The local quality of the goal states on all three trajectories is 100. The global quality value for the goal state in each solution trajectory differs, with 100 for the T_{short} goal (since it's the most efficient), 95 for T_{medium}, and 80 for T_{long} goal states.

From the start state to the goal in T_{short}, both local and global quality state values increase monotonically since it is the most efficient solution, and the chances of finishing the proof with an optimal solution increase with each step on this trajectory. Note that not all quality values increase over every trajectory. For example, step $T_{medium} - 2$'s pre-state ($+A \rightarrow E$) global quality is higher than that for its post-state ($+D$) since the pre-state is probabilistically on a more efficient path, but the local quality increases from pre- to post-state. Step $T_{long} - 3$'s pre-state ($+\neg E$) has higher local and global quality values than its post-state ($+\neg E \rightarrow \neg A$) since the post-state is farther from and less likely to reach T_{long}'s goal than the pre-state is to T_{short}'s closer goal. The global quality decreases for two reasons: (1) T_{long} goal is on a less efficient path, and (2) global quality performs a weighted sum over all the subsequent, previously-observed states in the larger (unshown) interaction network, many of which lead to incomplete attempts.

These three example trajectories demonstrate the differences between local and global state quality metrics. The main strength of generating these quality values is the MDP approach, which ensures that each state quality value is based not only on the distance from a solution but also on the probability of transition at each of the successive steps. This allows us to rate steps in a more probabilistic manner than a simple comparison based on the distance from the most efficient expert solution.

4.1.2.3. PROGRESS - CHANGE IN QUALITY

Since state quality is a measure of relative "goodness," we *compare* the quality of the current state with that of the previous and start states to evaluate the efficiency of a problem-solving step. In this section, we define two measures for *progress*: *relative*, the change in state quality from the *previous* problem-state, and *absolute*, the change in state quality from the *start* state.

Relative progress is the difference between the quality values of the current and previous states. Relative progress with local quality identifies whether the previous or current state is probabilistically closer to the goal. When using global quality values, the relative progress identifies which state is probabilistically closer and on a more efficient solution path.

Table 2: Step Efficiency formula based on state Quality and step Progress

Quality-Progress	Efficiency Formula
Global-Absolute	$(GQV_{post-state} - GQV_{start-state}) \geq 0$
Global-Relative	$(GQV_{post-state} - GQV_{pre-state}) \geq 0$
Local-Absolute	$(LQV_{post-state} - LQV_{start-state}) \geq 0$
Local-Relative	$(LQV_{post-state} - LQV_{pre-state}) \geq 0$

Consider a valid but long solution attempt. A relative progress measure reveals whether a student is progressing toward a solution in a step, but not whether their trajectory is efficient. Therefore, we define *absolute progress* as the difference between the current and start states' quality, using either quality measure. Absolute progress using local quality reveals whether a student's current state is probabilistically farther or closer from any goal states than when they began working on the problem. Global quality based absolute progress reveals the amount of efficient progress a student has made since the problem started. For example, if a student is always taking efficient steps, then the absolute progress will increase on every problem-solving step.

4.1.2.4. STEP EFFICIENCY - QUALITY & PROGRESS

We define four kinds of step efficiency measures based on quality {Local, Global} and progress {Relative, Absolute}. A step is considered *efficient* if the progress of its post-state using either quality measure is a non-negative number, and *inefficient* otherwise, as shown in Table 2.

We now compare the four kinds of step efficiency using three solution trajectories (T_{short}, T_{medium}, T_{long}) shown in Figure 2. We checked our ratings of inefficient and efficient steps with an expert logic instructor. Steps that were rated by our expert as inefficient are displayed in red, and others are in green in Figure 3. According to the local quality and absolute progress (local-absolute) efficiency metric, all the steps are efficient because they eventually lead to a solution. However, this metric is not sensitive to variations in solution lengths. When we use local-relative efficiency, only the $T_{long} - 3$ step is inefficient, as it is the only step where a post-state is probabilistically farther from a solution than the pre-state. Using the global-relative measure, steps $T_{medium} - 2$ and $T_{long} - 3$ are inefficient because they have a pre-state on a more efficient path to the solution than the post-state. The global-absolute metric is the only measure that labels the four expert-identified inefficient steps correctly[3]. Note that each type of efficiency captures a different perspective on a step towards the solution. The global-absolute efficiency metric aligns the most with expert labels for the sample trajectories, and our expert verified this alignment on a random sample of trajectories. Note that it would not be feasible to perform expert ratings on all student-derived steps – even in small logic proofs we examined here, we have N = 72,560 unique states in the prior student data for 35 problems and 796 students.

---

[3]Note that these four inefficient states also correspond to the four infrequently used (yellow) nodes in the student solution shown in Figure 2a. However, some inefficient nodes have been observed to be frequently used, and some efficient nodes to be infrequently used in our tutor, suggesting that the use-frequency alone cannot determine step efficiency, as was done in our prior work (Stamper and Barnes, 2009)

Table 3: HelpNeed detected in historical data using each type of Step Efficiency and their Correlation with students' posttest optimality (all correlations are significant with $p < 0.01$)

Step efficiency used to define HelpNeed	% of HelpNeed steps w/ this metric	Corr
Global-Absolute	24.86	-0.36
Global-Relative	23.76	-0.32
Local-Absolute	16.09	-0.29
Local-Relative	18.14	-0.31

4.1.2.5. SELECTING A STEP EFFICIENCY METRIC

To understand which one of the four step efficiency metrics is most indicative of how students' work in the tutor's training section affects their posttest solution optimality, we conducted a correlation test. Note, we evaluate students on an *optimality* score based on the number of steps on the posttest, with higher optimality scores given for fewer steps, explained in detail below in section 5.3.

We used only two of the datasets, f18 and s19 ($N = 437$), to perform our correlation analysis since the problem ordering changed in Fall 2018. Using these two datasets allows us to control for differences arising from problem ordering. For each student, we computed their posttest optimality and the proportion of training steps that are labeled HelpNeed using each efficiency metric. We then calculated the correlation between each type of training HelpNeed with posttest optimality using Pearson's coefficient. Table 3 shows that training HelpNeed is significantly and negatively correlated to posttest optimality for all four of the efficiency metrics. A negative correlation suggests that the higher the proportion of HelpNeed steps in training, the worse the posttest optimality. Among the step efficiency metrics, the global-absolute metric is the most correlated with posttest optimality. Therefore, we selected the **Global-Absolute step efficiency** to detect when students require tutor interventions. The remainder of the paper uses the term HelpNeed to refer to the definition that uses the global-absolute step efficiency.

4.2. HELPNEED PREDICTOR

Our prior work on proactive hints suggests that low prior knowledge students exert productive persistence when the tutor provides proactive hints as partially worked steps (Maniktala et al., 2020). Therefore, we sought to build a predictive HelpNeed classifier that could identify unproductive steps at their start so that we could proactively intervene with a hint and possibly convert unproductivity to productive persistence. This HelpNeed predictor predicts two classes: 1 for predicting HelpNeed, and 0 for otherwise.

To build the HelpNeed predictor, we engineered two types of classifiers: *state-based* and *state-free* in Python. We use a *state-based* classifier when a student's problem-solving state can be matched to historical data to leverage quality and progress-based features for predictions (see Appendix B). The *state-free* classifier is used when we don't have that information. This two-classifier architecture ensures that a HelpNeed prediction can be made regardless of whether a state is present in the historical data.

Feature Engineering and Selection. We used the datasets f18 and s19 ($N = 437$) to develop the HelpNeed predictor. We aggregate features on three levels of granularity: 1) most recent step, 2) current problem, and 3) total (all student-tutor interactions). Appendix B provides more information about our 63 features. We normalized the features and performed feature selection using the scikit-learn feature_selection.SelectFromModel (Pedregosa et al., 2011). Appendix C shows the features selected, along with their descriptive statistics.

Related research on creating data-driven step-level hint policies in multi-step domains for probability (Zhou et al., 2019) and linked lists (Fossati et al., 2015) have used individual models for each problem. However, we developed a generalized cross-problem model and compared it with problem-specific models. The generalized model performed equally well in AUC and recall when compared to problem-specific models for most of our tutor problems. Further, the generalized model performed better than the problem-specific models when less historical data was available, confirming that using a generalized model is a reasonable approach.

We now detail the predictive models we experimented with and our model selection approach. Table 3 shows that the input data is imbalanced as the proportion of HelpNeed steps is small at 24.86%. So, we used 10-fold cross validation with stratified random sampling[4]. We experimented with nine HelpNeed classifiers, including Random Forest (RF), Decision Tree (DT), Support Vector Classifier (SVC), Multi-layer Perceptron (MLP), Quadratic Discriminant Analysis, K-Nearest Neighbours, AdaBoost, Naive Bayes, all via scikit-learn, and XGBoost (Chen and Guestrin, 2016). We used the *sklearn* class_weight to determine *automated* weights for the RF, DT, SVC, and MLP models to account for imbalanced data. So the model would be more likely to correctly identify HelpNeed steps, we created *expert* weights (to improve recall) by performing a grid search starting from the automated weights for class 0, and performed a grid search for class 1 (HelpNeed), scoring the models on both recall and AUC (Arvai, 2018).

Table 4 shows the result of applying 10-fold cross validation on the classifiers with default, automated, and expert class weights for state-based and state-free predictions. We omitted the results for classifiers where class weights could not be adjusted because of their low performance. The table shows the RF models with expert class weights have the highest recall for both state-based and state-free predictions (0.90 and 0.91, respectively)[5]. RF outperforms all other models because it is both an ensemble model and benefits from expert weights.

For the selected RF state-based classifier, quality- and progress-based features contribute to 94.81% of the predictive power. Amongst these, the top three features to predict HelpNeed are: (1) Global-Absolute Progress (GAP = 33.5%), (2) current state's Global Quality = 22%), and (3) Local-Absolute Progress (LAP = 13.1%). And, for the selected RF state-free classifier, the top three features contributing to the classifier's predictive power are: (1) problem time (pTime = 10.7%), (2) total click-based actions in a problem (pActionCount = 8.5 %), and (3) wrong rule applications in the problem (pWrongApp = 7.4%). These three features are likely to be available in most well-structured multi-step domains. In ill-structured domains, there may not exist a program that can detect whether a single rule application is correct or not. More information on feature importance is provided in Appendix D.

The HelpNeed predictor is defined as the combined state-based and state-free RF classifiers that predict the next step's HelpNeed classification as in Table 1 at the start of that step using

---

[4]We also ensured that each student's data was fully contained within either the training or test set of each fold to avoid introducing bias

[5]The automated weights for these RF models are {class 1: 2.09, class 0: 0.66} while the expert weight for class 1 is 4.17 ($\sim$2x) for the *state-based* predictions and 3.75 ($\sim$1.8x) for the *state-free* predictions.

Table 4: Comparison of state-based and state-free predictors: Recall and AUC for 10-fold cross validation; Dflt= Default model; Auto = using automated weights; Exp = using expert weights

Predictor	State-based						State-free					
	Recall			AUC			Recall			AUC		
	Dflt	Auto	Exp	Dflt	Auto	Exp	Dflt	Auto	Exp	Dflt	Auto	Exp
RF	.62	.83	**.90**	.71	.84	**.83**	.20	.51	**.91**	.52	.62	**.62**
DT	.72	.84	.89	.82	.84	.82	.22	.46	.79	.53	.60	.54
SVC	.58	.52	.63	.77	.68	.73	.12	.28	.42	.53	.52	.52
MLP	.64	.62	.44	.75	.74	.67	.08	.24	.32	.53	.55	.54

the global-absolute metric. We performed a semester-based two-fold cross validation with f18 and s19 as the two folds to assess if the predictor can be used across semesters. We observed similar recall and AUC as that observed in the 10-fold cross validation, with recall for state-based predictions = 0.89 and state-free predictions = 0.89; and AUC for state-based predictions = 0.88 and state-free predictions = 0.56. This confirms that the HelpNeed predictive classifiers trained here can be expected to work in the future without the need for re-training.

5. EXPERIMENT

We conducted a controlled experiment to compare our *Adaptive* condition, where participants received proactive hints (Figure 4) when our predictor indicated HelpNeed, and the *Control* condition, where participants worked as usual without proactive hints. Note, students in both conditions could request on-demand hints.

5.1. HYPOTHESES

We have two hypotheses:

- **H1**: Students in the Adaptive condition will have better posttest performance than those in the Control condition, as measured by solution optimality and time.

- **H2**: Students in the Adaptive condition will exhibit better training behaviors, with (a) fewer HelpNeed steps, and (b) lower possible help avoidance, and higher possible help appropriateness (a higher chance of receiving help when it was likely to be needed), as measured using the HelpNeed classifier, when compared to the Control.

5.2. PROCEDURE

The tutor was a homework assignment in a Fall 2019 undergraduate discrete math course. The study was conducted with 123 participants, and the students were given ten days to complete the tutor.

The tutor is divided into four sections: introduction, pretest, training, and posttest. The introduction presents two worked examples to familiarize students with the tutor interface. Next,
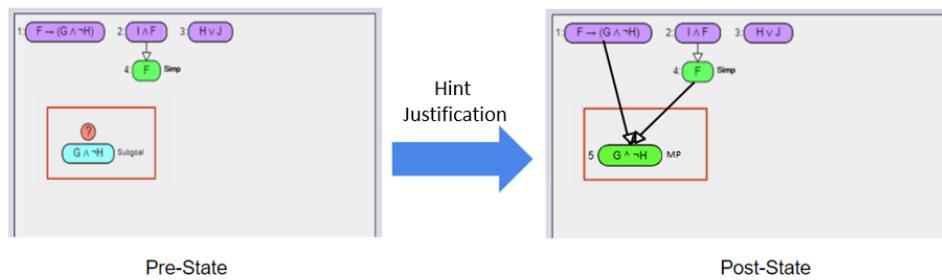
Figure 4: Proactive hint justification: A cyan proactive hint node (labeled Subgoal) is shown in the pre-State on the left. "Hint Justification" is performed by clicking on nodes 1 and 4 and the MP rule button in the pre-state. Since this is a correct justification, the hint becomes incorporated as a green node in the solution as shown in the post-state on the right.

students solve two problems in a *pretest*, which is used to determine students' incoming competence. Students are assigned a condition based on their pretest performance for stratified sampling, as detailed below. The pretest problems are designed to be easy and short, solvable with short optimal solution lengths ($Mean = 3.5$, $SD = 0.71$). Next, the tutor guides students through the *training* section with fifteen problems of varying difficulty. The difficulty of the training problems is between that of the pretest and the posttest based on averaging the optimal solution lengths over all training problems ($Mean = 4.99$, $SD = 1.32$). Finally, students take a more difficult *posttest* with five problems, with longer optimal solution lengths compared to the other sections ($Mean = 7.25$, $SD = 1.89$). Note that students can only receive hints in training, but the tutor is designed to provide immediate feedback on rule application errors in all the sections.

A stratified random sample based on pretest performance is used to partition the 123 participants into two conditions, resulting in 70 in *Adaptive* and 53 in *Control*. The stratified sampling was set up to result in a larger sample size for the Adaptive condition to gather more data on how the intelligent policy was carried out. Among these participants, 111 (66 in Adaptive and 45 in Control) completed the tutor. We used a chi-squared test to assess the impact of tutor completion rates on the group sizes, and found that the impact was not significant ($\chi^2(1, N = 123) = 0.16, p = .69$).

5.3. PERFORMANCE METRICS

The tutor automatically checks for problem completeness, so student solutions cannot be incorrect, but some may be more expert than others. Learning is measured by *performance* measures of optimality, time, and accuracy, reflecting that expert-like problem solutions have fewer steps, take less time, and have fewer mistakes. *Optimality* is an exponential decay function on normalized steps e^{-steps} to account for the small variance in the number of steps. $Steps$ are normalized to the interquartile range for each specific problem to account for varying problem lengths. Very short solutions with steps less than or equal to Q1 (1st quartile) have $optimality = 1$, and those with steps greater than Q3 (3rd quartile) have an optimality score of 0.36 or less based on the exponential decay curve. Next, similar to other studies (Kardan and Conati, 2015; Tchétagni and

[Nkambou, 2002](#)), our performance includes the *time* students spend solving problems. To make the time data into approximate normal distributions, we cap each action (any click performed) time to one minute[6] and sum the times for each action to determine the total (capped) time per problem. Finally, *accuracy* is defined as the number of correct rule applications divided by the total rule applications.

We do not hypothesize differences in the accuracy between the two conditions because the tutor is designed to provide immediate feedback on incorrect rule applications without penalties, even within the pre- and post-tests (see [5.2](#)). We report this performance metric to ensure our intervention does not harm students' accuracy.

5.4. HINT USAGE

Since our study investigates proactive hints, it is imperative to understand students' hint usage. We measure hint usage in the tutor using the Hint Justification Rate (HJR). As shown in Figure [4](#), a hint becomes *justified* if a student selects the correct rule and existing nodes to derive it. HJR is defined as the proportion of the hints given (on-demand or proactive) that were justified.

6. RESULTS

In this section, we investigate our two hypotheses about the Adaptive condition: H1 on improved posttest performance and H2 on (a) reduced HelpNeed steps and (b) reduced possible help avoidance and increased possible help appropriateness during training. We also evaluate the predictor's efficacy in predicting HelpNeed and show how the predictor can be used to assess students' help-seeking behavior.

6.1. H1 - POSTTEST PERFORMANCE

We first investigate hypothesis H1 on whether the Adaptive students have better posttest optimality and time than their Control peers. Table [5](#) shows the distribution parameters of students' pre-, and post-test performance in the two Conditions {Adaptive, Control}. As expected, no significant differences were found between the two conditions in the pretest *optimality*: $t(110) = 0.93$, $p = .18$, *time*: $t(110) = 1.27$, $p = .11$, and *accuracy* (proportion of correct rule applications): $t(110) = 0.81$, $p = .37$. This confirms that our stratified random sampling assignment balanced Adaptive vs. Control conditions' incoming competence (see section [5.2](#)).

Table 5: The distribution parameters for students' pre- and post-test performance in the two conditions. T-test shows that the Adaptive group has significantly higher posttest optimality ($p = .04$), and took significantly less time to complete the posttest ($p < .01$) than the Control group

Test	Optimality		Time (min)		Accuracy	
	Adaptive	Control	Adaptive	Control	Adaptive	Control
Pretest	.54 (.38)	.60 (.27)	39 (20)	34 (16)	.62 (.12)	.62 (.14)
Posttest	.71 (.27)	.59 (.33)	18 (12)	29 (17)	.71 (.10)	.68 (.08)

---

[6]The 3rd quartile of action time in Fall 2019 was 4.6s, and only 4690 out of 277,647 actions had an action time greater than 1 minute

A t-test on the posttest optimality shows a significant difference between the two conditions, $t(110) = 1.74$, $p = .04$, with a moderate effect size, Cohen's $d = 0.4$. Recall that higher optimality values reflect shorter, more efficient solutions. Next, on the total posttest time[7], significant differences were found between the two conditions with a large effect size, $t(110) = 3.99$, $p < .01$, Cohen's $d = 0.8$, with students in the Adaptive condition ($M = 18$ min, $SD = 12$ min) spending significantly less time on the posttest than those in the Control ($M = 29$ min, $SD = 17$ min). No significant differences were hypothesized or found between the two conditions in the posttest accuracy, $t(110) = 0.50$, $p = .31$. These results show that delivering proactive hints using the HelpNeed predictor indeed helped the Adaptive students generate *better* solutions in *less* time than their Control peers on the posttest without negatively impacting their accuracy, confirming our first hypothesis H1.

6.2.   H2A - COMPARISON OF HELPNEED DURING TRAINING

In this section, we investigate hypothesis H2a that the students in the Adaptive condition will have significantly fewer HelpNeed steps in training than the Control condition. Table 6 shows the cumulative step-level behavior of the two conditions during training. A t-test on the total training steps shows that the Adaptive condition took marginally significantly fewer total training steps on average compared to the Control condition: $t(110) = 1.29$, $p = .10$. The Adaptive condition also has significantly fewer quick inefficient steps per student over all training problems than the Control for both Opportunistic (Adaptive: 5, Control: 7, $p < .01$), and Far Off steps (Adaptive: 16, Control: 25, $p = .02$) but there were no significant differences in Futile steps between the two conditions (Adaptive: 13, Control: 12, $p = .45$). This suggests that compared with the Control condition, the Adaptive condition avoided unnecessary Opportunistic and Far Off steps that might distract them away from efficient solutions. Table 6 also shows that there are no significant differences in the Expert or Strategic steps between the two conditions (Expert: $p = .10$, Strategic: $p = 0.17$).

Table 6: Distribution parameters for students' training steps in the two conditions. T-tests show significant differences between the two conditions in the Far Off ($p < .01$) and Opportunistic ($p = .02$) steps. Total training steps are only marginally significant ($p = .10$)

Step Behavior	Description	# Training Steps		p
		Adaptive	Control	
Expert	Quick efficient steps	61 (12)	65 (13)	.10
Strategic	Long efficient steps	25 (19)	21 (9)	.17
Opportunistic	Singular, quick, inefficient steps	5 (3)	7 (4)	<.01*
Far Off	Consecutive quick but inefficient steps	16 (20)	25 (25)	.02*
Futile	Long inefficient steps	13 (12)	12 (19)	.47
	Total Training Steps	121 (38)	133 (39)	.10

---

[7]The posttest time distributions were normal but there was a significant difference in variance between the two groups using the Levene's Test ($p = .02$), so, we conducted a Welch's t-test to test for significant differences on post-test time between conditions.

The significantly higher Opportunistic and Far Off steps in the Control condition may be a result of help avoidance because students may not know when to seek help (Peña et al., 2011; Azevedo and Cromley, 2004). More details on students' help avoidance are discussed in section 6.7.

6.3. Hints Given and Used in Training

In this section, we further investigate the sources of differences between the Adaptive and Control conditions. Table 7 shows the mean and standard deviation of the total number of proactive, on-demand, and overall hints received by all students in the two conditions across training problems (the top part) and the hint justification rate (HJR) of students in the two conditions across different types of hints (the bottom part). Note that the Control condition was not provided with proactive hints, and thus only their on-demand hints count toward their total hints.

For the Adaptive condition, the total hints include both proactive and on-demand hints ($M = 34$, $SD = 12$), whereas, for the Control condition, the total hints only include the on-demand hints ($M = 14$, $SD = 15$). A Mann Whitney U test on the total hints for the two conditions shows a significant difference ($U = 445.5$, $z = 6.24$, $p < .01$). On average, students in the Adaptive condition received 28 proactive hints (22.8% of steps) and 6 on-demand hints (4.9% of steps), while the Control condition received an average of 14 hints on-demand (10.5% of steps). A Mann Whitney U test shows that students in the Adaptive condition requested significantly fewer on-demand hints during training than those in the Control condition ($U = 908.5$, $z = 3.46$, $p < .01$). This suggests that our HelpNeed predictor and proactive hints may have successfully forecasted the Adaptive students' needs for hints. Interestingly, we also found that whenever students in the Adaptive condition requested hints in a step, they did so after working on the step a median of 44 seconds before seeking help ($M = 70s$, $SD = 125s$), whereas the Control condition requested hints after working on the step for a median of 21 seconds ($M = 29s$, $SD = 21s$). A Mann Whitney U test on the time spent in a step before requesting help shows a significant difference between the two conditions ($U = 600$, $z = 1.97$, $p = .02$). This suggests that, on average, students in the Adaptive condition may engage longer on their own than their Control peers before requesting hints on a step. More details on help abuse are discussed in sections 6.5 and 6.7.

Table 7: Distribution parameters for the number of hints given and the hint justification rate in the two conditions

		Adaptive	Control	p
# Hints Received	Proactive	28 (10)	-	-
	On-demand	6 (9)	14 (15)	<.01*
	Total Hints	34 (12)	14 (15)	<.01*
% Hint Justification Rate (HJR)	Proactive	89% (8%)	-	-
	On-demand	87% (22%)	90% (11%)	.23
	Total HJR	89% (8%)	90% (11%)	.26

For HJR, the total hints HJRs are high, around 90% for both conditions, and no significant difference was found between the two conditions ($U = 1074.5$, $z = 1.21$, $p = .11$), which suggests that our hints are well-accepted by students in both conditions. More specifically, no significant differences are found in on-demand HJR between the two conditions (Adaptive: 87, and Control: 90, $U = 513.5$, $z = 1.84$, $p = .03$). Further, the Adaptive condition justified most of their hints regardless of delivery type (proactive HJR: 89%, and on-demand HJR: 87%). This affirms our prior results that students in the Adaptive condition incorporated proactive hints into their solutions as frequently as on-demand hints (Maniktala et al., 2020).

6.4. HINT COUNT AND POSTTEST PERFORMANCE

We found that the Adaptive condition received significantly more hints during training (section 6.3) and also performed significantly better on the posttest in optimality and time (section 6.1). We perform a correlation analysis comparing the Adaptive condition to students who also received frequent proactive hints at random times in previous semesters to determine whether this difference in performance could be due to the increased number of hints provided to the Adaptive group. Ideally, we would compare the HelpNeed policy with a policy that provides the same number of proactive hints randomly. However, this is difficult to achieve since the policy is adapting to individual students and the total number of proactive hints per student is neither predetermined nor consistent. Since we cannot directly compare student performance because of varying hint frequencies, we conducted a correlation analysis instead. Further, we contrast the correlation of hint count with posttest performance while controlling for pretest performance across datasets with Random, a combined dataset with Random and Adaptive proactive hint conditions, and the Adaptive condition alone. This can help us to understand differences in correlation arising from the Adaptive condition.

Table 8 shows the correlation between hint count (total and proactive) and posttest performance (Optimality and Time) with pretest performance as the covariates. First, we analyzed prior datasets, where students were given proactive hints at random intervals with about 40% and 33% frequency in f18R and f19R[8], respectively. Proactive hints in f18R and s19R were given frequently at random times irrespective of prior knowledge or progress on the problem, so *within* each random condition, there was no significant difference in the proactive hint count between students with low and high prior knowledge (Maniktala et al., 2020). For the f18R + s19R dataset, we found that the total hints (proactive + on-demand) did not correlate to the posttest performance metrics. Further, the number of proactive hints significantly correlated to longer posttest time but did not significantly correlate to posttest optimality. This suggests that receiving higher numbers of proactive hints on random steps during training can be detrimental to students' posttest time.

Proactive hints were provided on average of 23% student steps in the f19 Adaptive condition (f19A) of this study, which is lower than the f18R and s19R proportions. We repeated the correlation analysis including the f19 Adaptive condition (f19A) of this study to check if the lower f19A proactive hint count led to better posttest performance than the higher f18R and s19R proactive hint counts. On this combined dataset, the total hints received was not significantly correlated with posttest performance. However, the higher number of proactive hints (that presumably occurred more often in the studies with more frequent random hints) significantly

---

[8]f18R, and s19R were collected in Fall 2018 and Spring 2019 semesters respectively where R represents a subset of the population that received frequent unsolicited hints provided randomly

Table 8: Correlation analysis between hints given and posttest performance metrics

Dataset	Hints Given	Optimality		Time	
		corr	p	corr	p
f18R, s19R	Total	-0.05	.62	0.16	.13
	Proactive	-0.14	.19	0.49	< .01
f18R, s19R, f19A	Total	-0.05	.52	0.13	.11
	Proactive	-0.17	.04	0.27	< .01
f19A	Total	-0.003	.98	-0.15	.24
	Proactive	0.15	.24	0.04	.77

correlated with lower optimality and longer time. Finally, we performed the correlation between hint count and posttest performance for f19A alone. On this dataset, we observed insignificant correlations for all pairs of hint type {total, proactive}, and posttest metrics {optimality, time}. This is as expected because proactive hints were provided adaptively based on individual student needs and led to improved performance for all.

This analysis suggests that providing too many proactive hints can be detrimental to student performance. These results imply that students *do* need more hints than the number they request but providing hints when they are needed is more important than simply providing more.

6.5. GAMING BEHAVIOR

Since studies suggest that some students may game an educational system to complete the problems faster by using hints frequently (Baker et al., 2009), we analyze gaming behaviors. Our Adaptive hints policy is complex, so students are unlikely to guess what causes proactive hints to be triggered. A student trying to game the proactive hints in our tutor would have to carry out Futile or Far Off steps to receive proactive hints. However, Table 6 (training steps) shows that the Adaptive condition students had significantly fewer Far Off and Opportunistic steps during training than Control, and a similar number of Futile steps, giving no indication of increased gaming. Further, proactive hints that are partially worked steps require the student to do some work to use them, reducing the potential for gaming the system or possibly turning such attempts into learning opportunities.

We also analyzed how often students were gaming the system by requesting more hints than needed. Figure 5a shows that the on-demand hints in both the conditions constituted a small proportion of each student's total steps (Adaptive: $M = 4\%$, $SD = 8\%$; Control: $M = 12\%$, $SD = 14\%$). We found only one student from the Control condition who requested hints on more than 50% of their steps. Next, we investigated how often students requested hints in a short step time, where we consider "short" to be the top 25% quickest requests. We found that the 25th percentile (1st quartile) of step time before a student requested a hint in the study is 17 seconds. Figure 5b shows the histogram of students in the two conditions, comparing the number of training steps with too-quick hint requests (requested under 17 seconds). Summed over all problems in the training section, each student in the Adaptive condition had significantly fewer too-quick hint requests (requested in under 17s) than those in the Control condition (Adaptive: $M = 6$ too-quick requests per student, $SD = 10$; Control: $M = 15$ too-quick requests per student, $SD = 15$; $U =$

Figure 5: Comparison of Gaming in Deep Thought between the two conditions
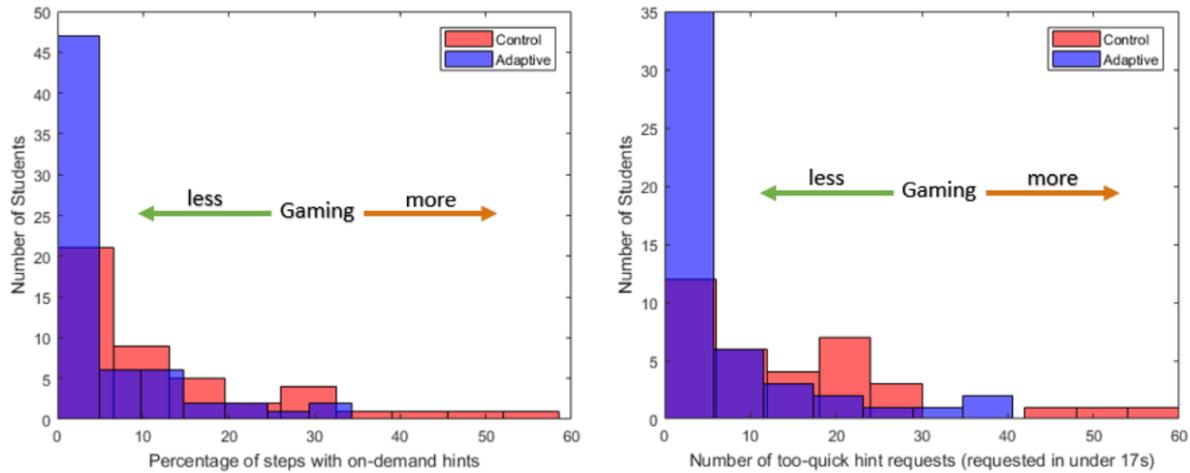
(a) Histogram for the proportion of training steps with hint requests

(b) Histogram for the number of too-quick hint requests

498.5, $z = 3.36$, $p < .01$). This suggests that students in the Adaptive condition are less likely to game the system by requesting hints without engaging long enough to think about the next step than students in the Control condition.

Overall, we observed that students in the Adaptive condition were less likely to game proactive or on-demand hints, with significantly fewer quick inefficient steps. The design of the proactive hints seems to successfully prevent gaming behaviors that detract from learning. However, we provide more details on help abuse in the next two sections.

6.6. THE POLICY'S EFFICACY IN PREDICTING HELPNEED

In this section, we evaluate the HelpNeed predictor by comparing the HelpNeed predictions with the observed step behavior during training for both conditions. Note that this comparison is complex because students can request hints at any time, and a hint received (proactive or on-demand) can change a step's observed HelpNeed if and when it is justified. Comparing the performance of our HelpNeed predictor on the two conditions can provide us some insights into how the predictor performs with and without predictor-driven interventions. We also use the predictor to compare student help-seeking behavior between the two conditions.

We evaluate the performance of the predictor in a manner similar to a confusion matrix, comparing step predictions with observed step behavior on 7970 and 5975 steps respectively for the Adaptive ($N = 66$ students) and Control ($N = 45$ students) conditions[9]. As an exploratory analysis, we compared the total HelpNeed predictions and observed HelpNeed steps between the two conditions. If a step is classified as HelpNeed, we call it an *observed HN* step. Alternatively, a step that is not classified as HelpNeed is called an *observed OK* step. Similarly, if the HelpNeed predictor predicts HelpNeed in a step, we call it a *predicted HN* step; otherwise, we call it a *predicted OK* step. We found 1858 (23.3%) observed HN training steps for the Adaptive group,

---

[9]The number of steps is larger for the Adaptive condition since there are more students assigned to this condition.

and 1660 (27.8%) observed HN training steps for the Control group. A chi-square test on the observed HN vs. OK training steps between the two conditions shows a significant difference ($\chi^2(1, N = 111) = 36.2, p < .01$). The Adaptive condition has a percent HelpNeed steps only 4.5% lower than that for the Control because the Adaptive condition's total training steps are also lower (marginally significant) than that of the Control. Section 6.2 shows that the Adaptive condition has significantly fewer Far Off and Opportunistic steps than the Control, which contributes to their reduced total steps as well. So, even though the Adaptive condition has fewer HelpNeed steps, their lower total steps make the difference in percent-HelpNeed between the two conditions small. We also compared the predictions of HelpNeed between the two groups. Interestingly, the Control condition had a noticeably higher proportion of predicted HN training steps than the Adaptive condition (Control: 2296, 38.4%, Adaptive: 1820, 22.8%). A chi-square test on predicted HN vs. OK training instances between the two conditions shows a significant difference ($\chi^2(1, N = 111) = 399.0, p < .01$). This result is consistent with the findings that students in the Adaptive condition were likely to follow their frequent hints, and this resulted in a higher proportion of observed OK steps, leading to fewer HN predictions.

Next, we investigated the correctness of step predictions and the impact of hint provision on the observed step behavior. Figure 6 shows bar graphs comparing training steps in the two conditions on whether a step is observed OK (top) or observed HN (bottom). Furthermore, each step can be categorized as prediction of HN/OK steps and hinted/no-hints (including both on-demand or proactive)[10]. Therefore, for both observed OK or observed HN steps, the proportions of training steps are further categorized based on the combination of HelpNeed predictor results {pred-OK = predicted OK, or pred-HN = predicted HelpNeed} with Hint provision {noHint = no hint was provided, hinted = a hint was provided} which resulted in four types:

1. *pred-OK + noHints*: predicted OK and no hints were provided
2. *pred-OK + hinted*: predicted OK but hints were provided
3. *pred-HN + noHints*: predicted HelpNeed but no hints were provided
4. *pred-HN + hinted*: predicted HelpNeed and hints were provided

The observed OK steps are shown in the top portion of Figure 6. The Adaptive group (4998, 62.7%) has a much higher percentage of **pred-OK + noHints** steps than the Control group (2974, 49.8%). This suggests that, while both conditions can solve a large number of steps on their own, the Adaptive students can successfully solve more steps on their own than the Control. On **pred-OK + hinted** steps, interestingly, we observe a very small proportion of such steps for the Adaptive group (106, 1.3%), considerably lower than that for the Control group (348, 5.8%). Assuming our HelpNeed predictor was perfect, these numbers would reflect the possible help abuse steps where students asked for hints when they didn't need them. While it is possible that our HelpNeed predictor could have incorrectly predicted these steps to be OK, we believe that these numbers still provide insights into relative possible help abuse between different conditions. For **pred-HN + noHints** steps, no such steps were observed for the Adaptive group simply because proactive hints were automatically provided for all predicted HN steps. However, we observe 761 (12.7%) such instances for the Control condition, where the HelpNeed predictor incorrectly predicted a step to be HN but was observed to be OK without help. Ideally, we want the misclassification rate to be as low as possible, with both the rates of false positives

_____

[10]Note, the graph provides only the percentages. The numbers corresponding to these eight comparisons are presented in Appendix E
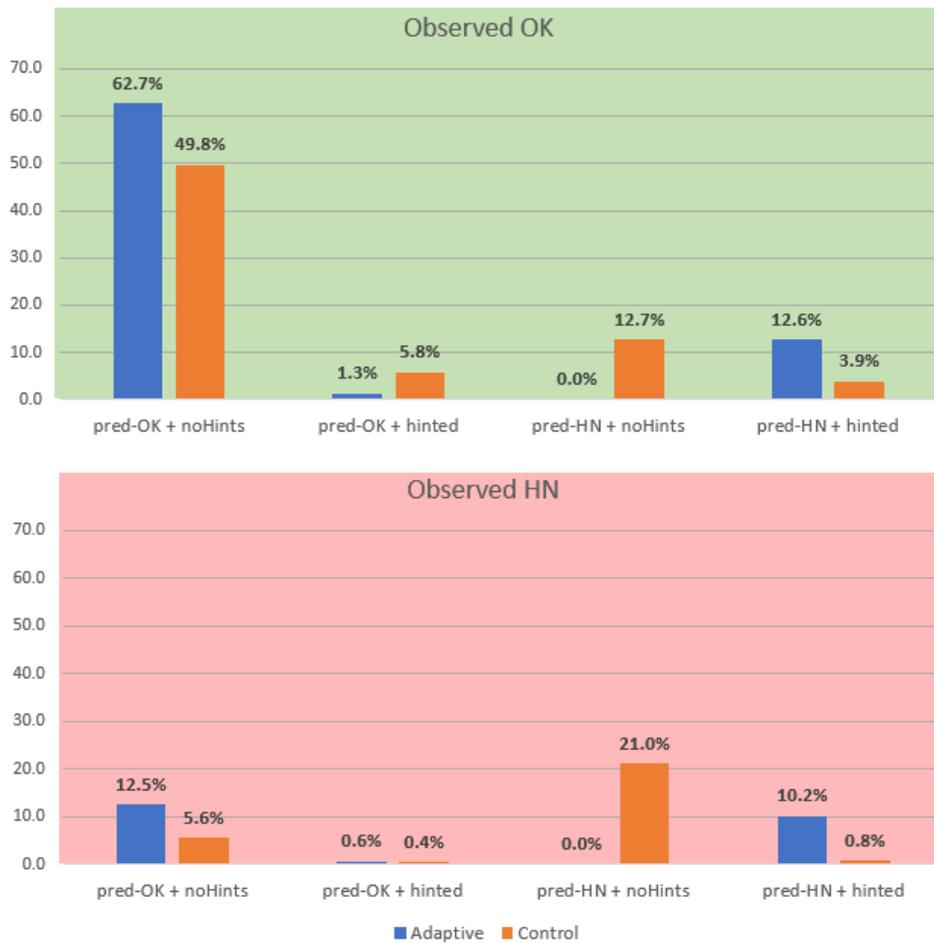
Figure 6: Comparing training steps in the two conditions on a combination of three aspects: predictions of HN or OK, observed HN or OK, and hint provision (on-demand or proactive)

(predicted HN and observed OK) and false negatives (predicted OK and observed HN) low. However, we trained the predictors to improve recall (as discussed in section 4.2) by lowering the rate of false negatives, i.e., we trained the model to prioritize correctly predicting HN steps so we could provide hints when they were most needed. Finally, for **pred-HN + hinted** steps, as expected, we observe a higher proportion of *pred-HN + hinted* steps for the Adaptive group (1008, 12.6%) than the Control group (232, 3.9%). We expected this number would be lower for students in the Control condition, who we thought would avoid asking for help. There are two possible scenarios for such Observed OK steps in the Adaptive group where students were predicted to need help and received a hint: ideally, 1) the hint helped students achieve a more efficient step than predicted, or 2) a student was given help unnecessarily.

Next, we examine the observed HN steps where steps were classified in a posthoc manner as Far Off or Futile. The lower portion of Figure 6 shows that for **pred-OK + noHints** steps, where the predictor failed to identify HelpNeed, is higher for the Adaptive group (995, 12.5%) than the Control group (335, 5.6%). This result is likely because the predictor is strongly biased toward efficient steps present in student solutions, and proactive hints always suggest efficient steps. Therefore, when students used them, their attempts appeared more expert, which, in turn,

Table 9: Definition of possible help avoidance, abuse, and appropriateness using HelpNeed

Help Behavior Metric	Definition
Possible Help Avoidance	%steps with observed HelpNeed but no hints were requested or received
Possible Help Abuse	%steps with no predicted or observed HelpNeed but hints were requested
Possible Help Appropriateness	%steps with predicted HelpNeed and hints were received

caused the predictor to predict that their next steps would also be more expert. This suggests that our predictor needs revision to include hint usage since some expert-like and strategic behaviors by the Adaptive students may be due to previously-received, justified hints. The proportion of training steps in the next **pred-OK + hinted** category is low in both conditions (Adopted: 51, 0.6% and Control: 22, 0.4%). The next category includes 1257 (21.0%) **pred-HN + noHints** steps for students in the Control condition, that were predicted and observed to need help, but no hints were received, representing possible help avoidance steps that the predictor correctly identified. Finally, the **pred-HN + hinted** steps were predicted to need help and received a hint, but still resulted in HelpNeed. The Adaptive group had 812 (10.2%), and the Control group had 46 (0.8%) of these steps. Note that our hints provide partial information to carry out the next step, but a student may not use the hint immediately– they may wait. Therefore, we further analyzed the usage of proactive hints in the Adaptive condition for this category and found that in 613 of these instances (75.5% of *pred-HN + hinted-HN*), students justified the proactive hints in the second step after the hint was provided. This suggests that the proactive hints did help students in these steps, but it took two steps before we could observe this impact.

6.7. H2B - EVALUATING HELP BEHAVIOR WITH HELPNEED PREDICTOR

In this section, we investigate H2b, that students in the Adaptive condition would have lower possible help avoidance and higher possible appropriate help, as measured using the HelpNeed classification, when compared to the Control. We also compare possible help abuse to determine whether the Adaptive hints impacted gaming behaviors. Note, we add a prefix *possible* to these behaviors because HelpNeed does not represent ground truth as classified by experts. Rather, HelpNeed is our classification of steps needing help, and the predictor is a heuristic measure.

Table 9 provides the definitions for the help behaviors we investigated, and Figure 7 shows a comparison in these behaviors between the conditions evaluated using the predictor. For each student, we define *possible help avoidance* as the percentage of total training steps that were observed to be HelpNeed but hints (on-demand or proactive) were neither requested nor proactively provided. Figure 7 shows that the Adaptive condition has a mean of 12.5% possible help-avoidance per student in training problems ($SD = 3.5\%$), and the Control condition has a mean of 26.6% possible help-avoidance per student in training problems ($SD = 5.2\%$). We found significantly lower possible help avoidance in the Adaptive condition than the Control ($U = 138$,
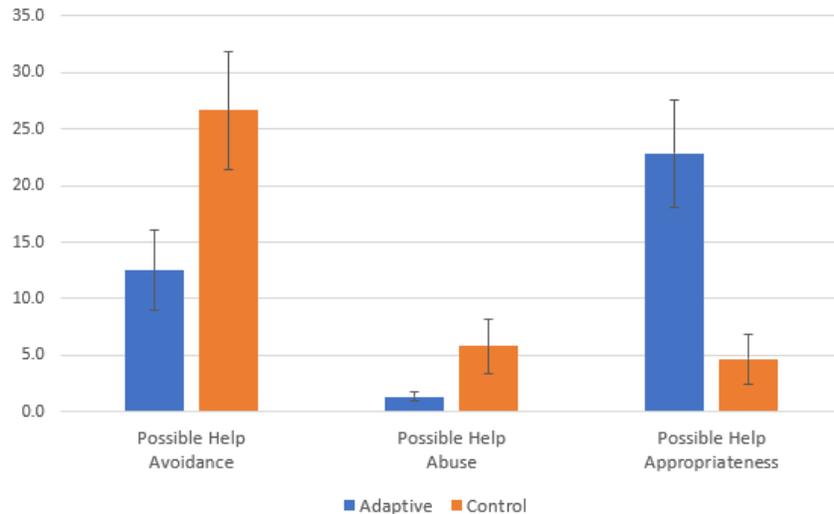
Figure 7: Comparison of possible help avoidance, abuse, and appropriateness between conditions using our HelpNeed classification and predictor

$p < .01$) with a moderate effect size = 0.59[11]. Next, we define *possible help abuse* as the percentage of training steps with neither predicted nor observed HelpNeed but students requested hints, indicating either that the prediction was wrong and help was needed and effectively used, or it was right and help was abused. We found a significant difference between the two conditions ($U = 365$, $p < .01$) and a moderate effect size = 0.40, with the Adaptive condition ($M = 1.3\%$, $SD = 0.4\%$) having lower possible help abuse than the Control ($M = 5.8\%$, $SD = 2.4\%$).

Finally, *possible help appropriateness* is the percentage of training steps predicted to need help and a hint was either requested or provided proactively. Figure 7 shows that the Adaptive condition has a mean of 22.8% possible help-appropriateness per student in training problems ($SD = 4.8\%$), and the Control condition has a mean of 4.7% possible help-appropriateness per student in training problems ($SD = 2.2\%$). We found significantly higher possible help appropriateness in the Adaptive condition than the Control ($U = 155$, $p < .01$) with a moderate effect size = 0.58. These results confirm hypothesis H2b that students in the Adaptive condition had lower possible help avoidance and higher possible appropriate help than students in the Control. We further show that students in the Adaptive condition demonstrated lower possible help abuse than students in the Control.

7. DISCUSSION

In this paper, we provide a unique extension of the Hint Factory to determine productivity on a step-level in an intelligent data-driven tutor. We present a HelpNeed metric and predictor to identify steps where students are likely to need help learning efficient problem-solving strategies. We also show the analysis evaluating the impact of intervening with Adaptive hints using our HelpNeed predictor in a controlled study. In this section, we discuss the HelpNeed approach, the evaluation of the Adaptive hint policy, as well as its limitations and potential for generalization.

---

[11]Effect size for Mann Whitney U test is calculated using $\dfrac{z^2}{n-1}$

7.1. HELPNEED CLASSIFICATION AND PREDICTOR

Our definition of unproductive, HelpNeed behavior is different from the existing literature that either defines problem-completeness-based metrics (Beck and Gong, 2013) or uses pre-defined domain-specific metrics that require expert knowledge or domain modeling (McLaren et al., 2014). We instead focus on solution length (i.e., optimality), which is valued across problem-solving domains. Further, our definition of HelpNeed also takes into account step duration. This is important because we do not want to disrupt helpful guess-and-check patterns that might involve students deriving an inefficient step but learning in the process.

A study by Fossati et al. (2015) on a proactive hint policy for the iList linked list tutor, incorporated step duration, using both problem-based parameters and individual student characteristics. Similarly, we combine a problem-based parameter for a duration threshold, after which steps are considered long, and each student's actual step duration to detect HelpNeed. The HelpNeed design ensures that we do not harm students who take, or need, more time when carrying out a learning task. Such students are likely to carry out longer steps that are classified as Strategic or Futile steps more than the quicker Expert, Far Off, or Opportunistic steps during training. Since we consider Strategic (long but efficient) steps productive, it gives such deliberating students more opportunities to learn by themselves. Further, a Futile (long and inefficient) step indicates that the student could benefit from proactive intervention, saving them time and directing them towards efficient strategies. Our prior work suggests that proactive hints promote learning through self-explanation and foster productive persistence among students with low prior knowledge (e.g., those with long step durations in the pre-test) (Maniktala et al., 2020). While this HelpNeed policy performed well, future work may explore the use of individual student parameters to tune the model.

Central to our HelpNeed approach is our novel data-driven metric of global quality, an extension of the Hint Factory. A core insight of this paper is that we can use the Hint Factory to determine step-level productivity. The original Hint Factory generated local quality values because it assigns state values assuming the best transition between each pair of states. We developed a modified Bellman equation to determine the global quality that employs a more probabilistic approach and varying rewards for goal states based on solution optimality/length. As demonstrated in section 4.1.2, the global quality successfully addresses our need for identifying step efficiency. One limitation of this work is that we consulted only one domain expert to rate a small sample of steps for assessing the ground truth of step-level productivity. As an alternative, we performed the correlation analysis in Table 3, which shows that training HelpNeed defined using each of the four step efficiency metrics significantly correlates to posttest performance. In future work, we plan to further explore the proposed quality and progress metrics to better understand their tradeoffs and determine whether they can be used across domains.

7.2. ADAPTIVE HINT POLICY

In this section, we discuss the results of providing partially worked steps as proactive hints using our HelpNeed predictor and our post-hoc evaluation of the predictor. Our results show that proactive hints using the predictor reduce students' Opportunistic and Far Off steps in training and enable them to form more optimal proofs in a shorter time on the posttest.

Our results showed that combining our HelpNeed predictor with proactive hints can *not only* reduce the number of times students need to explicitly ask for help *but also* reduce help avoidance. While one can argue that the increased number of total hints could have improved

the Adaptive condition's posttest performance, the correlation analysis in section 6.4 suggests that simply receiving more proactive hints at random times can be harmful, so it is important to identify when help is needed.

Prior research shows that gaming the system usually consists of quickly asking for successively more informative hints to get to a bottom-out hint that gives students the answer to the current question (Baker et al., 2008). Since our tutor focuses on multi-step problems where a correct answer is a set of correctly derived statements, no single hint can give a student a correct answer. We found that students in the Adaptive condition were less likely to game the system by requesting hints than the Control condition because the Adaptive group had significantly fewer steps with quick hint requests than the Control group. Another type of gaming occurs when students realize that a system predicts performance and provides proactive hints based on frequent failures (Baker et al., 2008). Since our policy uses both time and inefficient steps to determine when to proactively offer a hint, we believe it is more difficult for students to determine a pattern and game our system. Therefore, we recommend more complex policies to make it more difficult for students to game proactive hints. Our proactive hints are more likely to occur when we predict students to derive non-optimal statements. So, even students whose goal is to game our policy must still apply domain rules correctly, and therefore, still get valuable practice.

We also evaluated students' help behavior using our HelpNeed classification and predictor. We found that the Adaptive condition has significantly lower possible help avoidance and help abuse, with significantly more possible appropriate help than the Control. A limitation of this study is that this comparison relies on our HelpNeed classification and predictor, which are only shown to be correlated to posttest performance but have not been proven to correspond directly to expert measures of help need.

The evaluation of the HelpNeed predictor in section 6.6 suggests ways to identify steps where students did not show effective help-seeking. Further, this evaluation helped us understand how we can improve the predictor. The HelpNeed predictor predicted that students in the Adaptive condition would carry out more steps without demonstrated HelpNeed than they were able to. We believe this resulted from not differentiating efficient steps with hints from those without during the prediction task. Beck et al. (2008) proposed a Bayesian Evaluation model for a Reading Tutor to measure the impact of help on student knowledge. Further, a study by Chaudhry et al. (2018) used a deep learning model to jointly predict the hint-taking and knowledge tracing task. They found that incorporating the hint-taking model improved the deep learning knowledge tracing. Similar to these studies, our predictor should be revised to account for whether or not students received and justified hints.

7.3. Generalization to other domains

A limitation of this work is that we only assess our method in one ITS for one domain. To apply the HelpNeed model and predictor, the requirements for a new domain are to have state and state-transition representations (so we can define steps) and scoring for final solutions so the state-based and state-free classifiers can learn values for the HelpNeed predictor. Deriving state representations, and therefore the HelpNeed methods, should be relatively straightforward in well-structured domains such as multi-step math, physics, or statistics problems, but studies are needed to confirm their effectiveness.

It is possible that the HelpNeed methods will also work in programming-related contexts. For example, for the iList linked list tutor, Fossati et al. (2015) devised the Procedural Knowl-

edge Model (PKM) to identify "critical" states for feedback, using per-problem models. We believe that our HelpNeed model could be used to provide a general cross-problem model that could reduce the time needed to design per-problem models.

Several studies have applied the Hint Factory in the domain of programming by representing problem-solving states as abstract syntax trees (ASTs) (Iii et al., 2014; Rivers and Koedinger, 2013; Price et al., 2016). Researchers have already performed expert analyses (Dong et al., 2019) and preliminary automated methods to determine unproductivity (Marwan et al., 2020) in novice programming. In future work, we propose to compare the effectiveness of the HelpNeed method with these methods. We note that programming problems have large state-spaces (Price et al., 2016; Rivers and Koedinger, 2017), which can lead to fewer state-based matches but the methods proposed here can be used to developed state-free classifiers that do not require state matches to build a HelpNeed predictor.

Generalization will also require adapting the definition of both Opportunistic and Far Off steps. For the specific domain and tutor in this study, we defined Opportunistic steps as inefficient steps that *do not* need help and Far Off as inefficient steps that *do* need help. For other problem-solving domains, these categories should be defined based on how long a student should be allowed to work without intervention. For example, in a block-based programming problem that typically takes 1000 steps, it would be unreasonable to proactively intervene every 2 steps. We would expect to use a sliding window, tallying time and the proportion of inefficient steps, to determine when an inefficient step can be considered Opportunistic or Far Off. We would adjust the definitions of Opportunistic and Far Off steps until the definition applied to a historical dataset shows correlation with posttest performance as done here in Section 4.1.2.5.

8. CONCLUSIONS AND FUTURE WORK

In this paper, we present a novel approach for predicting HelpNeed in a logic tutor by extending the Hint Factory. We present a modification to the Bellman equation for quantifying the global quality of problem-solving states in well-structured open-ended domains. We show how this quality metric can be used to form a novel data-driven HelpNeed classification model for unproductivity. Overall, we found that providing partially worked steps as proactive hints upon predictions of HelpNeed can improve students' posttest problem-solving optimality and time. Furthermore, our Adaptive hint policy led students to improved training behaviors, with fewer steps predicted to need help, lower possible help avoidance and help abuse, and higher help appropriateness (receiving help when it was predicted to be needed). This work demonstrates that our novel, data-driven HelpNeed predictor can address the assistance dilemma for well-structured open-ended problem-solving in logic. The design of our system has revealed the importance of considering pedagogy and theoretical grounding in determining HelpNeed. Specifically, we based decisions about what step-level features may be indicative of HelpNeed on prior literature on learning. Related data-driven systems should also incorporate what is known about learning with the opportunities that data-rich systems afford.

This work has two main limitations. First, the HelpNeed predictor does not incorporate hint usage, meaning that some students with efficient steps based on hints were not given future help when it was needed. Second, the HelpNeed model and predictors here have only been evaluated in a single tutor in a single domain. While the step efficiency and HelpNeed metrics are designed to apply to well-structured, open-ended domains that value shorter solutions, studies are needed to confirm their effectiveness for other domains.

9. ACKNOWLEDGMENTS

REFERENCES

ALEVEN, V. AND KOEDINGER, K. R. 2000. Limitations of student control: Do students know when they need help? In *Proceedings of the 5th International Conference on Intelligent Tutoring Systems*, G. Gauthier, C. Frasson, and K. VanLehn, Eds. Springer, Berlin, Heidelberg, 292–303.

ALEVEN, V., MCLAREN, B., ROLL, I., AND KOEDINGER, K. 2006. Toward meta-cognitive tutoring: A model of help seeking with a cognitive tutor. *International Journal of Artificial Intelligence in Education 16,* 2, 101–128.

ANOHINA, A. 2007. Advances in intelligent tutoring systems: problem-solving modes and model of hints. *International Journal of Computers Communications & Control 2,* 1, 48–55.

ARROYO, I., BECK, J. E., BEAL, C. R., WING, R., AND WOOLF, B. P. 2001. Analyzing students' response to help provision in an elementary mathematics intelligent tutoring system. In *Papers of the Workshop on Help Provision and Help Seeking in Interactive Learning Environments at the Tenth International Conference on Artificial Intelligence in Education*. 34–46.

ARVAI, K. 2018. Fine tuning a classifier in scikit-learn. https://towardsdatascience.com/fine-tuning-a-classifier-in-scikit-learn-66e048c21e65.

AZEVEDO, R. AND CROMLEY, J. G. 2004. Does training on self-regulated learning facilitate students' learning with hypermedia? *Journal of Educational Psychology 96,* 3, 523–535.

BAKER, R. S., CORBETT, A. T., KOEDINGER, K. R., EVENSON, S., ROLL, I., WAGNER, A. Z., NAIM, M., RASPAT, J., BAKER, D. J., AND BECK, J. E. 2006. Adapting to when students game an intelligent tutoring system. In *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*, M. Ikeda, K. D. Ashley, and T.-W. Chan, Eds. Springer, 392–401.

BAKER, R. S., CORBETT, A. T., ROLL, I., AND KOEDINGER, K. R. 2008. Developing a generalizable detector of when students game the system. *User Modeling and User-Adapted Interaction 18,* 3, 287–314.

BAKER, R. S., DE CARVALHO, A., RASPAT, J., ALEVEN, V., CORBETT, A. T., AND KOEDINGER, K. R. 2009. Educational software features that encourage and discourage "gaming the system". In *Proceedings of the 14th International Conference on Artificial Intelligence in Education*, V. Aleven, J. Kay, and J. Mostow, Eds. 475–482.

BARNES, T. AND STAMPER, J. 2010. Automatic hint generation for logic proof tutoring using historical data. *Journal of Educational Technology & Society 13,* 1, 3–12.

BARNES, T., STAMPER, J., AND CROY, M. 2011. Using Markov decision processes for automatic hint generation. In *Handbook of Educational Data Mining*, C. Romero, S. Ventura, M. Pechenizkiy, and R. S. J. de Baker, Eds. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. CRC Press, Chapter 33, 467–480.

BARNES, T., STAMPER, J. C., LEHMANN, L., AND CROY, M. J. 2008. A pilot study on logic proof tutoring using hints generated from historical student data. In *Proceedings of the 1st International Conference on Educational Data Mining*, R. S. J. de Baker, T. Barnes, and J. E. Beck, Eds. 197–201.

BARTHOLOMÉ, T., STAHL, E., PIESCHL, S., AND BROMME, R. 2006. What matters in help-seeking? a study of help effectiveness and learner-related factors. *Computers in Human Behavior 22,* 1, 113–129.

BECK, J. E., CHANG, K.-M., MOSTOW, J., AND CORBETT, A. 2008. Does help help? Introducing the Bayesian evaluation and assessment methodology. In *Proceedings of the 9th International Conference on Intelligent Tutoring Systems*, B. P. Woolf, E. Aïmeur, R. Nkambou, and S. Lajoie, Eds. Springer, Berlin, Heidelberg, 383–394.

BECK, J. E. AND GONG, Y. 2013. Wheel-spinning: Students who fail to master a skill. In *Proceedings of the 16th International Conference on Artificial Intelligence in Education*, H. C. Lane, K. Yacef, J. Mostow, and P. Pavlik, Eds. Springer, Berlin, Heidelberg, 431–440.

BOREK, A., MCLAREN, B. M., KARABINOS, M., AND YARON, D. 2009. How much assistance is helpful to students in discovery learning? In *Proceedings of the 2009 European Conference on Technology Enhanced Learning*, U. Cress, V. Dimitrova, and M. Specht, Eds. Springer, Berlin, Heidelberg, 391–404.

BOTELHO, A., VARATHARAJ, A., PATIKORN, T., DOHERTY, D., ADJEI, S., AND BECK, J. 2019. Developing early detectors of student attrition and wheel spinning using deep learning. *IEEE Transactions on Learning Technologies 12,* 2, 158–170.

BUNT, A. AND CONATI, C. 2003. Probabilistic student modelling to improve exploratory behaviour. *User Modeling and User-Adapted Interaction 13,* 3, 269–309.

BUNT, A., CONATI, C., AND MULDNER, K. 2004. Scaffolding self-explanation to improve learning in exploratory learning environments. In *Proceedings of the 7th International Conference on Intelligent Tutoring Systems*, J. C. Lester, R. M. Vicari, and F. Paraguaçu, Eds. Springer, Berlin, Heidelberg, 656–667.

CAPRARO, M. M., AN, S. A., MA, T., RANGEL-CHAVEZ, A. F., AND HARBAUGH, A. 2012. An investigation of preservice teachers' use of guess and check in solving a semi open-ended mathematics problem. *The Journal of Mathematical Behavior 31,* 1, 105–116.

CEN, H., KOEDINGER, K. R., AND JUNKER, B. 2007. Is over practice necessary? Improving learning efficiency with the cognitive tutor through educational data mining. In *Proceedings of the 13th International Conference on Artificial Intelligence in Education*, R. Luckin, K. R. Koedinger, and J. Greer, Eds. Frontiers in Artificial Intelligence and Applications, vol. 158. IOS Press, NLD, 511–518.

CHAUDHRY, R., SINGH, H., DOGGA, P., AND SAINI, S. K. 2018. Modeling hint-taking behavior and knowledge state of students with multi-task learning. In *Proceedings of the 11th International Conference on Educational Data Mining*, K. E. Boyer and M. Yudelson, Eds. International Educational Data Mining Society, 21–31.

CHEN, T. AND GUESTRIN, C. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 785–794.

CODY, C. AND MOSTAFAVI, B. 2017. Investigating the impact of unsolicited next-step and subgoal hints on dropout in a logic proof tutor. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. ACM, 705–705.

CONATI, C., GERTNER, A., AND VANLEHN, K. 2002. Using Bayesian networks to manage uncertainty in student modeling. *User Modeling and User-Adapted Interaction 12,* 4, 371–417.

CORBETT, A. T. AND ANDERSON, J. R. 1994. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction 4,* 4, 253–278.

DONG, Y., MARWAN, S., CATETE, V., PRICE, T., AND BARNES, T. 2019. Defining tinkering behavior in open-ended block-based programming assignments. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. 1204–1210.

EAGLE, M., HICKS, D., AND BARNES, T. 2015. Interaction network estimation: Predicting problem-solving diversity in interactive environments. In *Proceedings of the 8th International Conference on Educational Data Mining*, O. C. Santos, J. Boticario, C. Romero, M. Pechenizkiy, A. Merceron, P. Mitros, J. M. Luna, M. C. Mihaescu, P. Moreno, A. Hershkovitz, S. Ventura, and M. C. Desmarais, Eds. International Educational Data Mining Society, 342–349.

FOSSATI, D., DI EUGENIO, B., OHLSSON, S., BROWN, C., AND CHEN, L. 2015. Data driven automatic feedback generation in the iList intelligent tutoring system. *Technology, Instruction, Cognition and Learning 10,* 1, 5–26.

FRATAMICO, L., CONATI, C., KARDAN, S., AND ROLL, I. 2017. Applying a framework for student modeling in exploratory learning environments: Comparing data representation granularity to handle environment complexity. *International Journal of Artificial Intelligence in Education 27,* 2, 320–352.

III, B. P., HICKS, A., AND BARNES, T. 2014. Generating hints for programming problems using intermediate output. In *Proceedings of the 7th International Conference on Educational Data Mining*, J. C. Stamper, Z. A. Pardos, M. Mavrikis, and B. M. McLaren, Eds. International Educational Data Mining Society, 92–98.

KAI, S., ALMEDA, M. V., BAKER, R. S., HEFFERNAN, C., AND HEFFERNAN, N. 2018. Decision tree modeling of wheel-spinning and productive persistence in skill builders. *Journal of Educational Data Mining 10,* 1, 36–71.

KARDAN, S. AND CONATI, C. 2015. Providing adaptive support in an interactive simulation for learning: An experimental evaluation. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 3671–3680.

KINNEBREW, J. S., SEGEDY, J. R., AND BISWAS, G. 2014. Analyzing the temporal evolution of students' behaviors in open-ended learning environments. *Metacognition and Learning 9,* 2, 187–215.

KLAHR, D. 2009. "To every thing there is a season, and a time to every purpose under the heavens": What about direct instruction? In *Constructivist Instruction: Success or failure?*, S. Tobias and T. M. Duffy, Eds. Routledge, New York, Chapter 15, 291–310.

KOCK, M. AND PARAMYTHIS, A. 2010. Towards adaptive learning support on the basis of behavioural patterns in learning activity sequences. In *2010 International Conference on Intelligent Networking and Collaborative Systems*. IEEE, 100–107.

KOEDINGER, K. R. AND ALEVEN, V. 2007. Exploring the assistance dilemma in experiments with cognitive tutors. *Educational Psychology Review 19,* 3, 239–264.

MANIKTALA, M., CODY, C., BARNES, T., AND CHI, M. 2020. Avoiding help avoidance: Using interface design changes to promote unsolicited hint usage in an intelligent tutor. *International Journal of Artificial Intelligence in Education 30,* 4, 637–667.

MARWAN, S., DOMBE, A., AND PRICE, T. W. 2020. Unproductive help-seeking in programming: What it is and how to address it. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. 54–60.

MAYER, R. E. 1992. *Thinking, problem solving, cognition*. WH Freeman/Times Books/Henry Holt & Co.

MCLAREN, B., TIMMS, M., WEIHNACHT, D., BRENNER, D., LUTTGEN, K., GRILLO-HILL, A., AND H. BROWN, D. 2014. A web-based system to support inquiry learning. In *Proceedings of the 6th International Conference on Computer Supported Education - Volume 1*. SciTePress, Setubal, PRT, 43–52.

MCLAREN, B. M., LIM, S.-J., AND KOEDINGER, K. R. 2008. When and how often should worked examples be given to students? New results and a summary of the current state of research. In

Proceedings of the 30th Annual Conference of the Cognitive Science Society, B. C. Love, K. McRae, and V. M. Sloutsky, Eds. Cognitive Science Society, Austin, TX, 2176–2181.

MERCERON, A. AND YACEF, K. 2005. Educational data mining: a case study. In *Proceedings of the 12th International Conference on Artificial Intelligence in Education*, C. Looi, G. I. McCalla, B. Bredeweg, and J. Breuker, Eds. Frontiers in Artificial Intelligence and Applications, vol. 125. IOS Press, 467–474.

MOSTAFAVI, B. AND BARNES, T. 2017. Evolution of an intelligent deductive logic tutor using data-driven elements. *International Journal of Artificial Intelligence in Education 27,* 1, 5–36.

MOSTAFAVI, B., LIU, Z., AND BARNES, T. 2015. Data-driven proficiency profiling. In *Proceedings of the 8th International Conference on Educational Data Mining*, O. C. Santos, J. Boticario, C. Romero, M. Pechenizkiy, A. Merceron, P. Mitros, J. M. Luna, M. C. Mihaescu, P. Moreno, A. Hershkovitz, S. Ventura, and M. C. Desmarais, Eds. International Educational Data Mining Society, 335–341.

MURRAY, R. C. AND VANLEHN, K. 2005. Effects of dissuading unnecessary help requests while providing proactive help. In *Proceedings of the 12th International Conference on Artificial Intelligence in Education*, C. Looi, G. I. McCalla, B. Bredeweg, and J. Breuker, Eds. Frontiers in Artificial Intelligence and Applications, vol. 125. IOS Press, 887–889.

MURRAY, R. C. AND VANLEHN, K. 2006. A comparison of decision-theoretic, fixed-policy and random tutorial action selection. In *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*, M. Ikeda, K. D. Ashley, and T. Chan, Eds. Lecture Notes in Computer Science, vol. 4053. Springer, 114–123.

MURRAY, R. C., VANLEHN, K., AND MOSTOW, J. 2004. Looking ahead to select tutorial actions: A decision-theoretic approach. *International Journal of Artificial Intelligence in Education 14,* 3, 4, 235–278.

PARK, S. AND MATSUDA, N. 2018. Predicting students' unproductive failure on intelligent tutors in adaptive online courseware. In *Proceedings of the Sixth Annual GIFT Users Symposium*. Vol. 6. US Army Research Laboratory, 131–138.

PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., BLONDEL, M., PRETTENHOFER, P., WEISS, R., DUBOURG, V., ET AL. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12,* Oct, 2825–2830.

PEÑA, A., KAYASHIMA, M., MIZOGUCHI, R., AND DOMINGUEZ, R. 2011. Improving students' meta-cognitive skills within intelligent educational systems: A review. In *Proceedings of the 6th International Conference on Foundations of Augmented Cognition*, D. D. Schmorrow and C. M. Fidopiastis, Eds. Springer, Berlin, Heidelberg, 442–451.

POLYA, G. 2004. *How to solve it: A new aspect of mathematical method*. Princeton University Press.

PRICE, T. W., DONG, Y., AND BARNES, T. 2016. Generating data-driven hints for open-ended programming. In *Proceedings of the 9th International Conference on Educational Data Mining*, T. Barnes, M. Chi, and M. Feng, Eds. International Educational Data Mining Society, 191–198.

PRICE, T. W., LIU, Z., CATETÉ, V., AND BARNES, T. 2017. Factors influencing students' help-seeking behavior while programming with human and computer tutors. In *Proceedings of the 2017 ACM Conference on International Computing Education Research*. ACM, 127–135.

PRICE, T. W., ZHI, R., AND BARNES, T. 2017. Hint generation under uncertainty: The effect of hint quality on help-seeking behavior. In *Proceedings of the 18th International Conference on Artificial Intelligence in Education*, E. André, R. S. Baker, X. Hu, M. M. T. Rodrigo, and B. du Boulay, Eds. Lecture Notes in Computer Science, vol. 10331. Springer, 311–322.

PUUSTINEN, M. 1998. Help-seeking behavior in a problem-solving situation: Development of self-regulation. *European Journal of Psychology of Education 13,* 2, 271–282.

RIVERS, K. AND KOEDINGER, K. R. 2013. Automatic generation of programming feedback: A data-driven approach. In *Proceedings of the First Workshop on AI-supported Education for Computer Science at the 16th International Conference on Artificial Intelligence in Education*, E. Walker and C. Looi, Eds. CEUR Workshop Proceedings, vol. 1009. 50–59.

RIVERS, K. AND KOEDINGER, K. R. 2017. Data-driven hint generation in vast solution spaces: a self-improving Python programming tutor. *International Journal of Artificial Intelligence in Education 27,* 1, 37–64.

ROLL, I., ALEVEN, V., MCLAREN, B. M., AND KOEDINGER, K. R. 2011. Improving students' help-seeking skills using metacognitive feedback in an intelligent tutoring system. *Learning and Instruction 21,* 2, 267–280.

RUS, V., BANJADE, R., NIRAULA, N., GIRE, E., AND FRANCESCHETTI, D. 2017. A study on two hint-level policies in conversational intelligent tutoring systems. In *Innovations in Smart Learning*, E. Popescu, Kinshuk, M. K. Khribi, R. Huang, M. Jemni, N.-S. Chen, and D. G. Sampson, Eds. Springer, Singapore, 175–184.

SMITH, M. U. 2012. *Toward a Unified Theory of Problem Solving: Views From the Content Domains*. Routledge, New York.

STAMPER, J. AND BARNES, T. 2009. An unsupervised, frequency-based metric for selecting hints in an MDP-based tutor. In *Proceedings of the 2nd International Conference on Educational Data Mining*, T. Barnes, M. C. Desmarais, C. Romero, and S. Ventura, Eds. International Educational Data Mining Society, 181–190.

STAMPER, J., BARNES, T., LEHMANN, L., AND CROY, M. 2008. The Hint Factory: Automatic generation of contextualized help for existing computer aided instruction. In *Proceedings of the 9th International Conference on Intelligent Tutoring Systems Young Researchers Track*. 71–78.

TCHÉTAGNI, J. M. AND NKAMBOU, R. 2002. Hierarchical representation and evaluation of the student in an intelligent tutoring system. In *Proceedings of the 6th International Conference on Intelligent Tutoring Systems*, S. A. Cerri, G. Gouardères, and F. Paraguaçu, Eds. Lecture Notes in Computer Science, vol. 2363. Springer, 708–717.

UENO, M. AND MIYAZAWA, Y. 2017. IRT-based adaptive hints to scaffold learning in programming. *IEEE Transactions on Learning Technologies 11,* 4, 415–428.

VANLEHN, K. 2011. The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist 46,* 4, 197–221.

WOOD, H. AND WOOD, D. 1999. Help seeking, learning and contingent tutoring. *Computers & Education 33,* 2-3, 153–169.

YACEF, K. 2005. The Logic-ITA in the classroom: a medium scale experiment. *International Journal of Artificial Intelligence in Education 15,* 1, 41–62.

ZHOU, G., AZIZSOLTANI, H., AUSIN, M. S., BARNES, T., AND CHI, M. 2019. Hierarchical reinforcement learning for pedagogical policy induction. In *Proceedings of the 20th International Conference on Artificial Intelligence in Education*, S. Isotani, E. Millán, A. Ogan, P. M. Hastings, B. M. McLaren, and R. Luckin, Eds. Lecture Notes in Computer Science, vol. 11625. Springer, 544–556.

A. APPENDIX: PROOF OF CONVERGENCE FOR THE MODIFIED BELLMAN BACKUP FUNCTION

Theorem: The modified value iteration (Eqn 2) converges to GQV^* for any initial estimate GQV, i.e.,

$$\lim_{k \to \infty} GQV_k = GQV^* \quad \forall GQV$$

For any estimate of the value function GQV, we define the modified Bellman backup operator $\hat{B} : R^{|S|} \to R^{|S|}$

$$\hat{B}GQV(s) = GR(s) + \gamma \sum_{s' \in S} P_a(s'|s)GQV(s')$$

Before we provide the proof of convergence, we provide the proof of contraction, i.e., for any two value functions GQV and GQV':

$$||\hat{B}GQV_k - \hat{B}GQV'_k|| \le \gamma ||GQV_k - GQV'_k||$$

where the max norm:

$$||GQV|| = \max_{s \in S} |GQV(s)|$$

$||v - v'||$ = Infinity norm (max difference over all states)

Proof of contraction:

$$||\hat{B}GQV - \hat{B}GQV'||$$

$$= \left|\left|\left[GR(s) + \gamma \sum_{s' \in S} P_a(s'|s)GQV(s')\right] - \left[GR(s) + \gamma \sum_{s' \in S} P_a(s'|s)GQV'(s')\right]\right|\right|$$

$$= \gamma \left|\left|\left[\sum_{s' \in S} P_a(s'|s)GQV(s') - \sum_{s' \in S} P_a(s'|s)GQV'(s')\right]\right|\right|$$

$$= \gamma \left|\left|\left[\sum_{s' \in S} P_a(s'|s)(GQV(s') - GQV'(s'))\right]\right|\right|$$

$$\le \gamma \max_s \sum_{s' \in S} P_a(s'|s)|GQV(s') - GQV'(s')|$$

$$\le \gamma \sum_{s' \in S} P_a(s'|s)||GQV - GQV'||$$

$$= \gamma ||GQV - GQV'||$$

since $P_a(s'|s)$ are non-negative and sum to one

Proof of Convergence:

$$||GQV_{k+1} - GQV^*||_\infty = \left|\left|\hat{B}GQV_k - GQV^*\right|\right|_\infty \le \gamma ||GQV_k = GQV^*||_\infty \le ...$$

$$\le \gamma^{k+1} ||GQV_0 - GQV^*||_\infty \longrightarrow 0$$

B. LIST OF ALL THE FEATURES ENGINEERED ALONG WITH THE LEVEL AT WHICH THEY WERE AGGREGATED (STEP (S), PROBLEM (P), TOTAL (T))

| Feature	Level	Description
GAP	s	Absolute progress on the current state using global quality
GRP	s	Relative progress of the current state using global quality
LAP	s	Absolute progress of the current state using local quality
LRP	s	Relative progress of the current state using local quality
localPrevious	s	Local Quality of the previous state
globalPrevious	s	Global Quality of the previous state
localCurrent	s	Local Quality of the current state
globalCurrent	s	Global Quality of the current state
Time	s, p, t	Time Taken
AvgStepTime	p, t	Average Step Time
ActionCount	s, p, t	Total number of actions performed (e.g. selecting nodes, rule clicks, hint button clicks, etc.).
DirectProof ActionCount	s, p, t	The number of actions performed while working on the problem as a direct proof. Students can switch between direct and indirect proof at any given time.
IndirectProof ActionCount	s, p, t	The number of actions performed while working on the problem as an indirect proof. A logic proof is said to be indirect if we first assume the negation of conclusion to be true and then arrive at a contradiction.
DirectionChange	s, p, t	The number of time a student switched between direct and indirect proof using the direction switch button.
FDActionCount	s, p, t	The number of actions performed while working on a forward step. The student can switch between forward and backward step at any time
BDActionCount	s, p, t	The number of actions performed while working on a step backwards using instantiation and generalization logic rules.
StepCount	p, t	The number of steps performed
SolSize	p	The solution size of a problem, calculated as the number of the logic statements in the current state
RuleDescription	s, p, t	The number of times a student clicked the description button of any logic rule.

HintRequest	s, p, t	The number of time the hint was requested by the student using the "Get Suggestion" button
ProactiveHintCount	s, p, t	The number of proactive hints given.
OnDemandHintCount	s, p, t	The number of on-demand hints given.
Deleted	p, t	The number of nodes deleted.
RightApp	p, t	The number of Right logic rule applications.
WrongApp	s, p, t	The number of Wrong logic rule applications.
Accuracy	s, p, t	Proportion of total logic rule applications that are correct.
SessionCount	t	Total number of sessions - each time a student logs back in, we increment the number of sessions.
NewSession	p	Binary variable: 1 for the first problem in any session
Skips	t	Total number of problems skipped.
Restarts	t	Total number of problems restarted.
EasyProblems	t	The total number of easy problems solved.
DifficultProblems	t	The number of difficulty problems solved.

C. Descriptive Statistics for the Features Selected for State-Based and State-Free Random Forest Classifier with Expert Weights (Grey cells suggest inapplicable)

Feature	State-Based	State-Free	Mean	SD
GAP	Yes		9.91	32.77
GRP	Yes		4.26	21.27
LAP	Yes		33.85	37.70
LRP	Yes		9.05	20.90
localPrevious	Yes		31.22	33.20
globalPrevious	Yes		50.68	32.06
localCurrent	Yes		40.26	37.76
globalCurrent	Yes		54.94	34.31
pTime	Yes	Yes	718.55	4051.65
pAvgstepTimePS	Yes	Yes	130.39	1306.00
tAvgstepTimePS	Yes	Yes	266.36	429.40
sActionCount	No	Yes	5.98	6.47
pActionCount	Yes	Yes	58.60	98.97
tActionCount	Yes	Yes	1304.02	870.32
pDirectProofActionCount	Yes	Yes	54.06	89.52
tDirectProofActionCount	Yes	Yes	1204.48	795.68
pInDirectProofActionCount	No	Yes	4.55	33.72
tInDirectProofActionCount	No	Yes	99.54	216.65
pDirectionChanges	Yes	No	0.17	0.97
pFDActionCount	Yes	Yes	58.03	97.75
tFDActionCount	Yes	Yes	1283.66	855.38
pBDActionCount	Yes	Yes	0.58	2.97
tBDActionCount	Yes	Yes	20.36	33.04
tStepCount	No	Yes	119.40	76.72
pSolSize	Yes	Yes	7.45	3.00
sRuleDescription	Yes	No	0.57	2.34
pRuleDescription	Yes	Yes	7.46	25.68
tRuleDescription	Yes	Yes	196.96	230.58
pHintRequest	Yes	Yes	1.64	5.48
tHintRequest	Yes	Yes	11.60	19.83
sDeleted	No	Yes	0.13	0.36
pDeleted	Yes	Yes	1.71	4.53

tDeleted	Yes	Yes	28.95	27.32
pRightApp	Yes	Yes	6.42	7.10
tRightApp	Yes	Yes	119.40	76.72
sWrongApp	Yes	Yes	0.45	1.24
pWrongApp	Yes	Yes	4.92	10.51
tWrongApp	Yes	Yes	246.16	162.78
sAccuracy	No	Yes	0.69	0.41
pAccuracy	No	Yes	0.73	0.25
tSessionCount	Yes	No	3.11	2.22
pNewSession	Yes	No	0.05	0.21
pRestarts	Yes	Yes	0.13	0.52
tRestarts	No	Yes	4.79	5.42
tEasyProblemsPS	Yes	Yes	3.59	2.75

D. Feature Importance Graphs

Feature Importance (%) for the Selected State-Based
Random Forest Classifier with Expert Weights
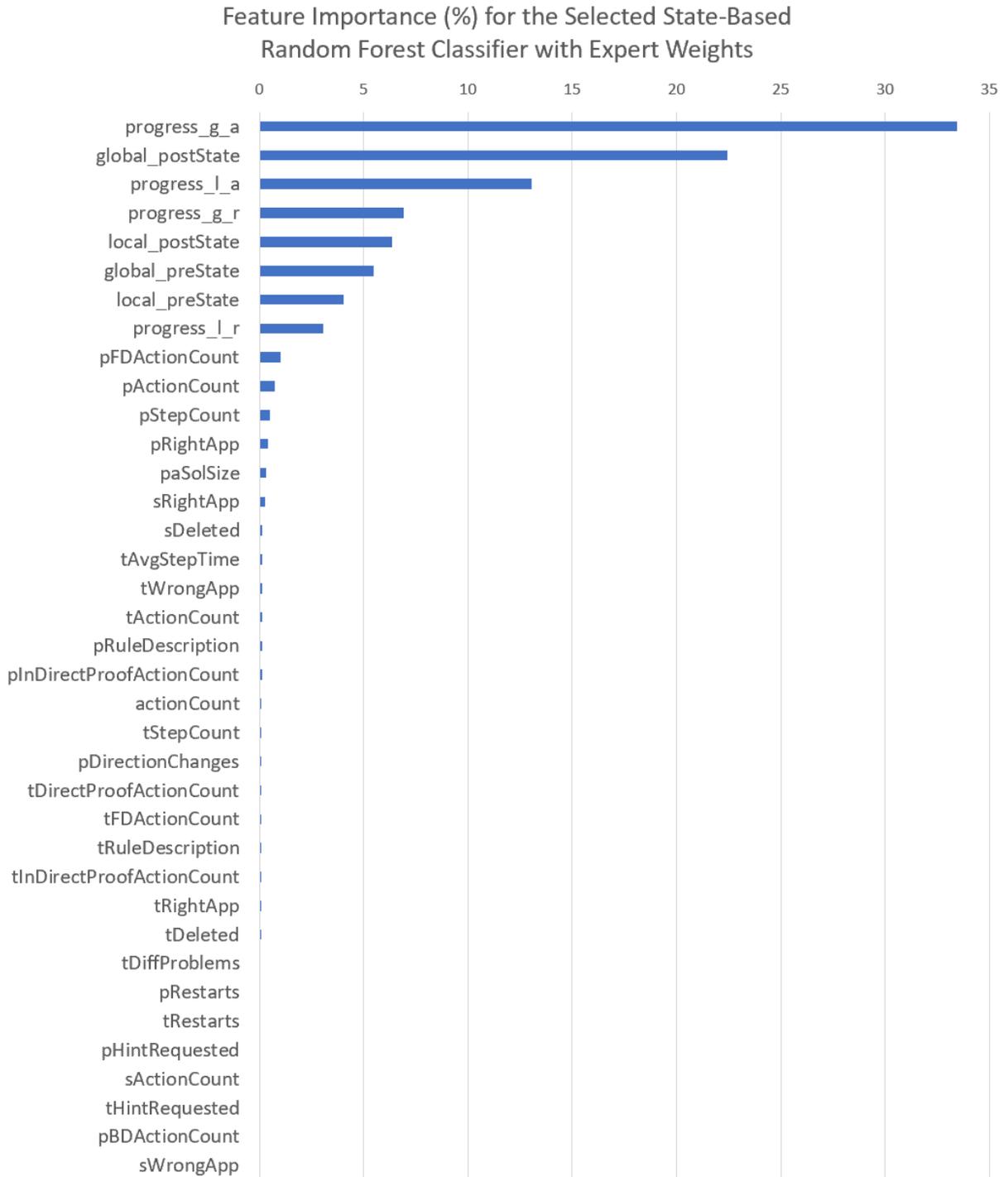
Feature Importance (%) for the Selected State-Free Random Forest
Classifier with Expert Weights
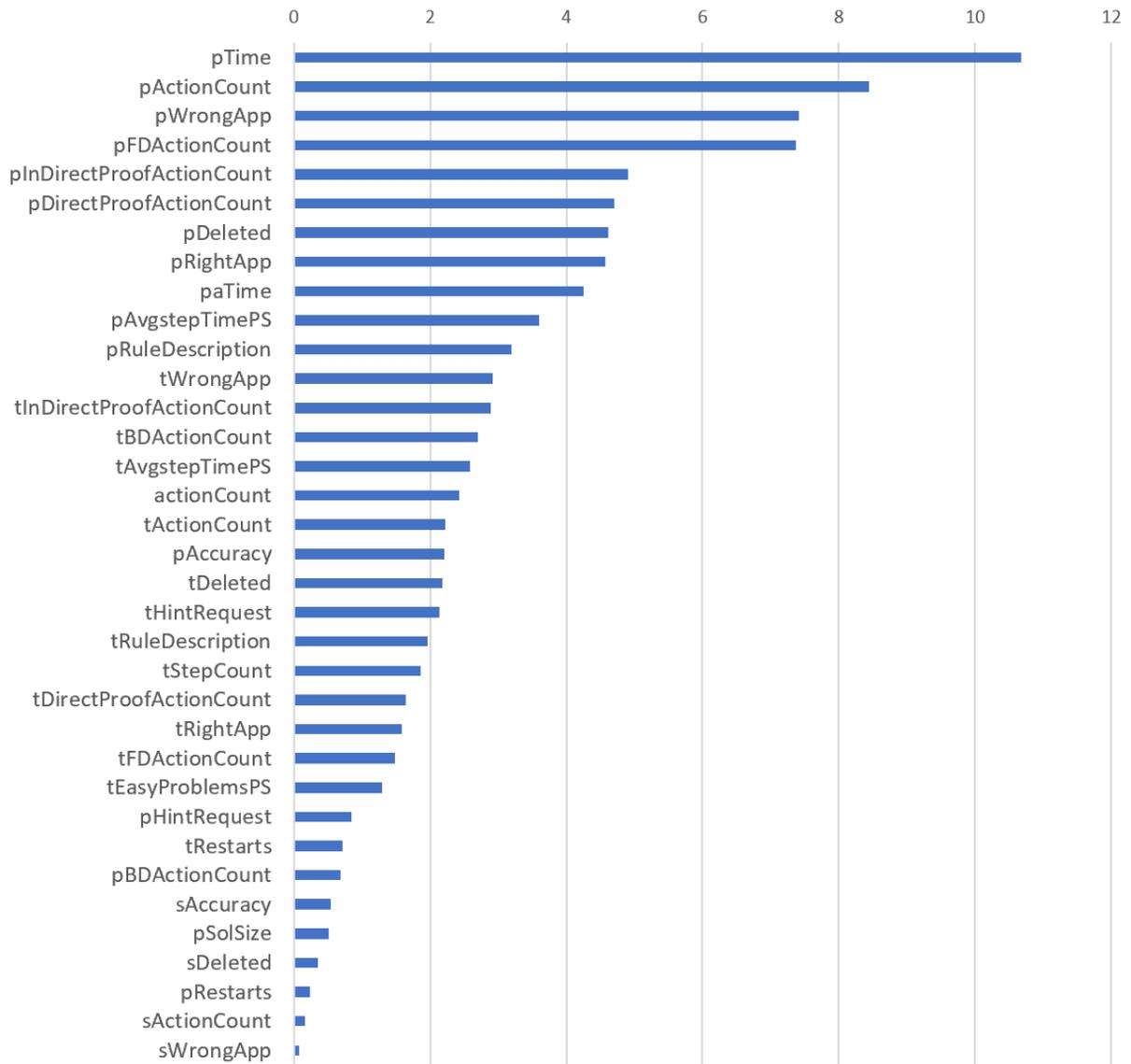
E. COMPARING TRAINING STEPS IN THE TWO CONDITIONS ON THREE ASPECTS: PREDICTION OF HN/OK STEPS, OBSERVED HN/OK STEPS, AND HINT PROVISION (ON-DEMAND OR PROACTIVE)

Category	Adaptive	Control
pred-OK + noHints-OK	4998	2974
pred-OK + hinted-OK	106	348
pred-HN + hinted-OK	1008	232
pred-HN + noHints-OK	0	761
pred-HN + noHints-HN	0	1257
pred-HN + hinted-HN	812	46
pred-OK + hinted-HN	51	22
pred-OK + noHints-HN	995	335