International Journal of Artificial Intelligence in Education 22 (2013) 3–17
DOI 10.3233/JAI-130029
IOS Press

Experimental Evaluation of Automatic Hint Generation for a Logic Tutor

John Stamper, *Human-Computer Interaction Institute, Carnegie Mellon University, Pittsburgh, PA, USA*
john@stamper.org

Michael Eagle, *Department of Computer Science, University of North Carolina at Charlotte, Charlotte, NC, USA*
maikuusa@gmail.com

Tiffany Barnes, *Department of Computer Science, North Carolina State University, Raleigh, NC, USA*
tiffany.barnes@gmail.com

Marvin Croy, *Department of Philosophy, University of North Carolina at Charlotte, Charlotte, NC, USA*
mjcroy@uncc.edu

Abstract. We have augmented the Deep Thought logic tutor with a Hint Factory that generates data-driven, context-specific hints for an existing computer aided instructional tool. We investigate the impact of the Hint Factory's automatically generated hints on educational outcomes in a switching replications experiment that shows that hints help students persist in a deductive logic proofs tutor. Three instructors taught two semester-long courses, each teaching one semester using a logic tutor with hints, and one semester using the tutor without hints, controlling for the impact of different instructors on course outcomes. Our results show that students in the courses using a logic tutor augmented with automatically generated hints attempted and completed significantly more logic proof problems, were less likely to abandon the tutor, performed significantly better on a post-test implemented within the tutor, and achieved higher grades in the course.

Keywords. Data mining, machine learning, logic tutor

INTRODUCTION

The Hint Factory is a data-driven automatic hint generator built to augment interactive learning environments with contextualized hints, to help realize the learning gains provided by intelligent tutoring systems while reducing the authoring burden for their creation. The Hint Factory uses an Interaction Network with values using a Markov decision process (MDP), created from past student data, to generate specific contextualized hints for students using CAI. These hints help students solve problems by suggesting the next step a student should take when solving a multi-step problem. We have augmented the Deep Thought logic proofs tutor with the Hint Factory to automatically deliver context-specific hints to students solving logic proofs. For these logic problems, the hints suggest the operator to use and specifically how to use that operator, but these are tailored to the tutor domain.

One criticism of data-driven approaches is that simple step-based hints do not provide students with higher-level problem solving feedback that would promote transfer of learning from one problem to better

1560-4292/13/$27.50 © 2013 – IOS Press and the authors. All rights reserved

performance on later problems. In this research, we conduct a switching replications study to investigate the impact of data-driven hints on educational outcomes of learning and persistence. The Hint Factory was used to add hints to seven of eleven Deep Thought logic proof problems. Deep Thought was used for homework problems in six deductive logic classes across two semesters. To control for the impact of different instructors, each of the three college philosophy instructors taught one semester of Deductive Logic with the Deep Thought tutor augmented with our data-derived hints, and each taught a semester using the tutor without hints.

We hypothesized that providing our data-driven, context-specific hints would improve students' ability to solve the given proof problems, and that having these hints available while students are solving practice problems would improve overall learning. Results show that students with hints attempt and complete significantly more problems than students in the no-hint group. Further, students with hints available on early problems outperformed students in the control group on the post-test.

BACKGROUND AND RELATED WORK

Effective formative feedback should be multidimensional and credible, specific but not evaluative, and infrequent but timely (Shute, 2008). We offer hints on demand, instead of proactively, since this can have positive effects on learning (Razzaq & Heffernan, 2010). While help-seeking behaviors can be unproductive (Aleven et al., 2004), we don't limit hint usage, particularly since using hints can be seen as looking for a worked example, an effective learning strategy (Shih et al., 2008).

Historically, the hints and feedback in most intelligent tutoring systems (ITS) have been built through extensive consultation with subject-area experts. In particular, two classes of effective tutors, cognitive tutors and constraint-based tutors, rely on "rules" that experts create in a time-intensive process (Mitrovic et al., 2003). While this expertise and time are limited, the amount of data being collected from computer-aided instruction continues to grow at an exponential rate. Data-driven methods applied to large data repositories like the PSLC DataShop (Koedinger et al., 2010) can enable the rapid creation of new intelligent tutoring systems, making them accessible for many more students. Several such data-driven methods have been used to gain deep knowledge about student models simply from problem attempt data, including SimStudent (Li et al., 2011) and LFA Search (Koedinger et al., 2012). While these methods focus on the student models and knowledge tracing, the Hint Factory uses student attempt data for model tracing and hints within a particular problem.

As with RomanTutor, an ITS that uses sequential pattern mining on collected data to recommend actions to astronauts learning to operate a robot arm (Nkambou et al., 2008), we provide direct, data-driven feedback in an environment where students can choose from a large space of actions to perform, and many of these actions are correct. We construct Interaction Networks that represent all student approaches to a particular problem, and use them as Markov Decision Processes directly to generate hints with the Hint Factory (Barnes et al., 2008). Barnes and Stamper demonstrated the feasibility of this approach on historical data, showing that the Hint Factory could provide context-specific hints over 80% of the time (Barnes & Stamper, 2008). Our pilot study experiment showed that the Hint Factory can provide sufficient, correct, and appropriate hints for the Deep Thought Logic tutor while helping students solve more logic proof problems (Barnes et al., 2008). Almost the opposite of Bootstrap Novice Device (BND), which bootstraps example-based tutors with student data (McLaren et al., 2004) we create a data-driven tutor that can be bootstrapped with expert solutions (Stamper et al., 2011). The Hint Factory can evolve, providing at least some automatically generated hints initially and improving as additional expert and student problem attempts are added to the model.

Fossati and colleagues have used the Hint Factory to combine data-driven MDPs with human tutor dialog data to provide proactive feedback in the iList linked list tutor (Fossati et al., 2009). We are currently

working to build a Hint Factory to provide hints for novices in an open computer-programming environment (Jin et al., 2011). This paper extends work by Stamper, Eagle, Barnes, and Croy in AIED 2011 (Stamper, Eagle, Barnes, & Croy, 2011) to include a more thorough exposition of experimental data, the Deep Thought tutor, and promising work on applying graph mining methods to our extracted domain models to understand student problem-solving.

HINT FACTORY

The Hint Factory consists of an MDP generator and a hint provider. The MDP generator is created through an offline process that assigns values to states in student problem attempt data. The hint provider uses these values to select the next "best" state at any point in the problem space. The MDP Generator uses historical student data to generate an Interaction Network that provides the basis for Markov Decision Process (MDP) that represents a student model, containing all previously seen problem states and student actions. An Interaction Network transforms all tutor states and actions into a graph with associated frequencies. A Markov decision process (MDP) is defined by its state set S, action set A, transition probabilities T, and a reward function R (Sutton & Barto, 1998). For a particular point in a student's logic proof, a state consists of the list of statements generated so far, and actions are the rules used at each step. Actions are directed arcs that connect consecutive states. Therefore, each proof attempt can be seen as a graph with a sequence of states connected by actions. When combined into one graph, we call this an Interaction Network. When transforming an Interaction Network into an MDP, frequencies are used as estimates of transition probabilities. The goal of using an MDP is to determine the best path through this graph that corresponds to solving the given problem. This is achieved by calculating *V(s)*, the expected rewards to be earned (on average) by following an optimal policy from state s, calculated recursively using equation (1), where *R(s,a)* is the reward for taking action *a* from state *s*, and $P_a(s, s')$ is the probability that action *a* will take state *s* to state s'. $P_a(s, s')$ is the number of times action *a* is taken from state *s* to s', divided by the total number of actions leaving state *s*.

$$V(s) := \max_a \left(R(s, a) + \sum_{s'} P_a\left(s, s'\right) \cdot V\left(s'\right) \right) \tag{1}$$

Once value iteration is complete, the optimal solution in the MDP corresponds to taking an expert-like approach to solving the given problem, where from each state the best action to take is the one that leads to the next state with the highest expected reward value (Barnes et al., 2008). The Hint Factory uses these values when a student is in a particular state to choose the next "best" state which the system selects as the recommended next target state, and it generates hints to help the student move towards that state. When the hint button is pressed, the hint provider searches for the current state in the MDP and checks that a successor state exists. If it does, the successor state with the highest value is used to generate a hint sequence. A hint sequence refers to hints that are all derived based on the same current state. For each state, four distinct hints are generated. If a student requests a hint, then makes an error, and requests a hint again, the next hint generated is the next one in the current sequence. Once a student performs a correct step, the hint sequence is reset.

DEEP THOUGHT LOGIC TUTOR

We have added the Hint Factory to Deep Thought using data from three prior semesters of data taught by one professor from the tutor on the seven problems, taken from college students enrolled in the same

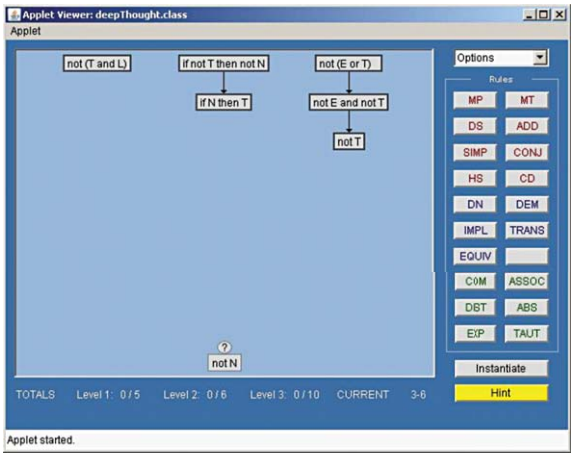

Fig. 1. The Deep Thought tutor showing a partially completed solution to problem 2–4.

introductory deductive logic course. Deep Thought is a Java applet that allows students to visually connect premises and apply logical axioms and theorems to solve proofs (Croy et al., 2007). Each problem provides a set of initial premises (e.g. not(T and L), if not T then not N, not (E or T) in Fig. 1) and a desired conclusion (e.g. not N in Fig. 1). Students can select premises from the proof, and choose logical operations to perform on those premises, such as Modus Tollens (MT), from the toolbar on the right. This interface allows both working forward from the premises or backward from the conclusion. For example, given the conclusion B, a student could assume that the conclusion was derived using Modus Ponens and create two new unjustified statements with an unknown variable '?' added to the proof: ? $\longrightarrow$ B and ?. The student could continue working backwards, or justify these new statements by finding a value for the unknown variable. For problems with hints, a bright yellow hint button appears in the bottom right, allowing the student to request a hint on their current step.

There are several related intelligent tutors for teaching propositional logic: the Carnegie Mellon Proof Tutor (CPT) (Scheines & Sieg, 1994), the P-Logic Tutor (Lukin, et al. 2002) and Logic-ITA (Yacef, 2005). In CPT, students construct a goal tree and Fitch diagrams to develop a proof. In Deep Thought, students also construct a diagram connecting statements to the goal, but do not construct subgoals directly. In Logic-ITA, students write a proof as a sequence of text proof lines, and the tutor checks each step as it is written and provides delayed feedback on the usefulness of each step. Deep Thought allows students to work in a more free-form fashion working both forwards and backwards, and checks each step as it is performed, but does not provide feedback other than automatically detecting a correct completion. P-Logic Tutor has several levels with text and worked examples, providing a richer tutoring environment that spans more than proof writing. Their study with discrete math students showed that students had trouble with the restrictive requirements for rule applications, and while the system suggested several subgoals, students wanted more guidance on what to try. Deep Thought has some similar limitations with restrictive rule applications, but the hint system is highly specific. The MathsTiles automated proofs assistant was written as a general tool for students to use while writing proofs, and allows for free-form, undirected work (Billingsley and Robinson, 2007). The design guidelines of MathsTiles have students write proofs, rather than selecting goals for a system, and allow for the creation of advice (hints) that are tailored based on student feedback on their usefulness. Their research showed that although a few students could learn to use the system in a short amount of time, the general nature of their system combined with requirements for a proof solver and the undirected nature of the tiles made it difficult for students to use. Deep Thought requires students to connect statements in forward or backward directions, making the proof verification simpler.

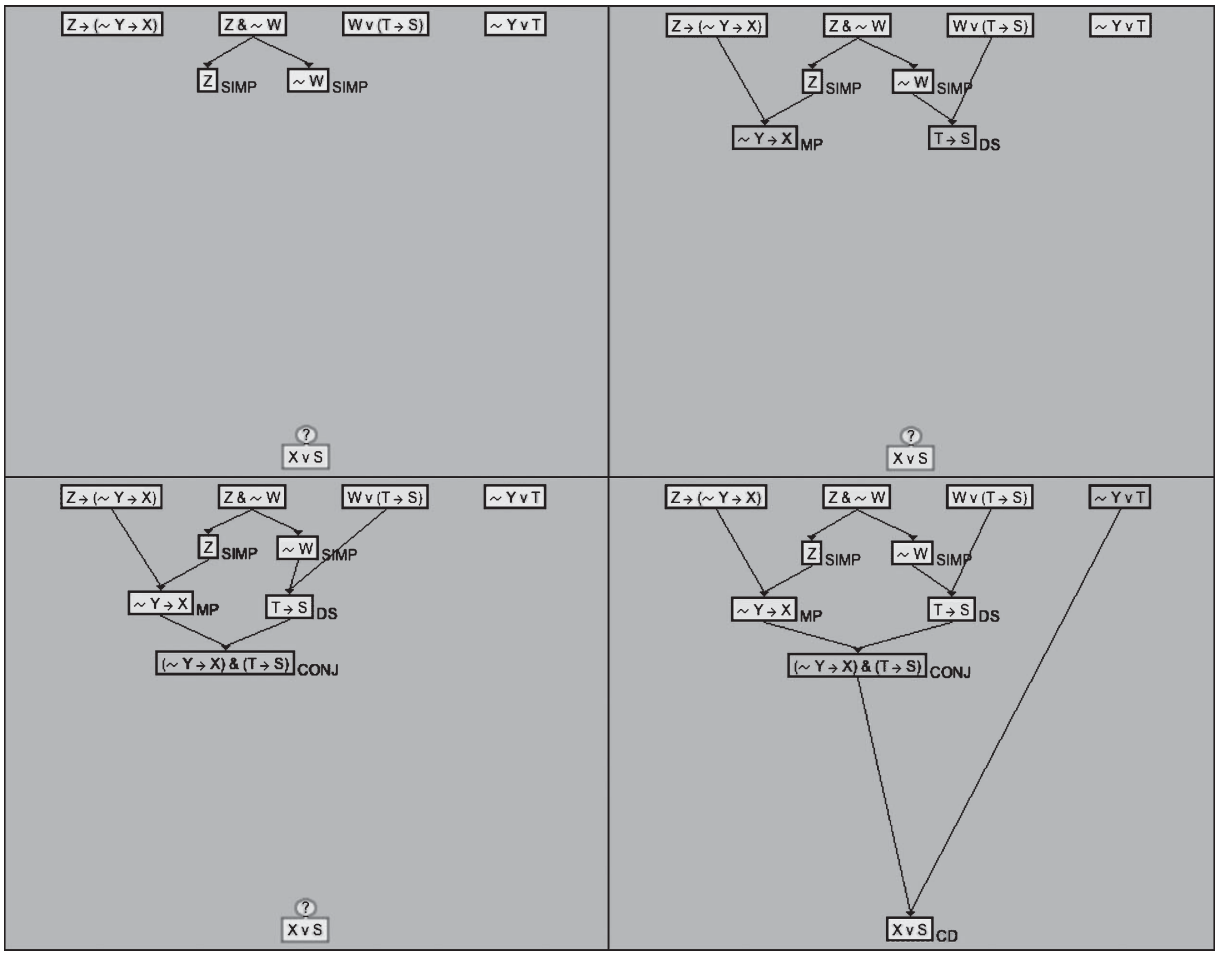

Fig. 2. Solution progress for problem 1–4 at several stages of a student's proof.

Figure 2 illustrates several steps in solving problem 1–4. The first row is the given premises, while the bottom statement is the given conclusion. In the first tile, Z and ∼W (not W) are each derived using Simplification (SIMP) from Z&∼W. In level 1, this is done by students selecting Z&∼W and pressing the SIMP button, and selecting left or right side; Deep Thought then creates new nodes for Z or ∼W. In level 3, students type in the derived statement. In successive tiles, new steps are highlighted in green (for the figure only, not in the tutor). When a derived statement is equal to the conclusion, Deep Thought removes "?" to show the statement is justified, and connects the source statements [(∼Y⟶X)&(T⟶S)] and (∼Y∨T) to the goal with the label of the rule used (Constructive Dilemma, CD).

The Hint Factory for Deep Thought generates a sequence of four hints for each state: 1) indicate a goal expression to derive and direction to work, 2) indicate the premises where the rule can be used, 3) indicate the rule to apply next, and 4) a bottom-out hint combining 1–3. Table 1 shows the hint sequence generated for the problem state shown in Fig. 1. When the student presses the hint button, a pop-up window appears with the text for Hint #1. When the hint button is pressed again, the hint text contains Hints 1 and 2, and so on, until all 4 hints are given. These hints include pointing hints that help focus user attention, and bottom-out hints that tell students the answer (VanLehn, 2006). These hints, inspired by the scaffolds in the

Table 1
An automatically generated hint sequence for the problem state in Fig. 1

Hint #	Hint text
1	Try to derive "not N" working forward
2	Highlight "if N then T" and "not T" to derive it
3	Click on the rule Modus Tollens (MT)
4	Highlight "if N then T" and "not T" and click on Modus Tollens (MT) to get not N

Table 2
Deep Thought tutor problems. * problems where hints were available for the Hint group. Deep Thought supports the symbols: not: ∼ /¬; and: &/∧; or: ∨; implies: ⟶

Problem	Given	Prove
1.1*	A⟶(B∧C), A∨D, ¬D∧E	B
1.2	(¬K∨L)⟶(M∧N), K⟶O, ¬O	N
1.3*	(¬T∧S)⟶ ¬R,¬T, (¬Q ∨ P)⟶S, Q⟶T	¬R∨N
1.4*	Z⟶(¬Y⟶X), Z∧¬W, W∧(T⟶S), ¬Y∨T	X∨S
1.5	B⟶(A⟶J), D∧¬(A⟶¬C), B∧(A⟶¬C), J⟶¬C	A⟶¬C
1.6	(Y∧Z)⟶M, ¬M∧(¬N⟶G), (W⟶J)∨(Y∧Z), J⟶¬N	W⟶G
2.1	T⟶M,¬¬¬M	¬(T∨M)
2.2*	(A⟶¬B)∨C, ¬C, D∨B	¬D⟶¬A
2.3*	K⟶M, Z⟶R, ¬(K⟶R)	M∧¬Z
2.4*	¬(T∧L), ¬T⟶¬N, ¬(E∨T)	¬N
2.5*	Y = P, ¬Y⟶¬C, ¬P = ¬C	Y⟶C
3.1	S∨B, B⟶D, S⟶G	D∨G
3.2	J⟶H, ¬W⟶(¬H∧¬F), J	W

ASSISTments system (Razzaq & Heffernan 2006), were specifically designed to correspond to the ways instructors provide hints on logic proof problems.

Table 2 shows Deep Thought homework problems, in order of increasing difficulty. Level L1 uses primarily inference rules; Levels L2 and L3 require transformations before inference rules can be applied, making these problems more complex. Level 3 is the post-test, with no hints for either group.

EXPERIMENT

Students from six different sections of a deductive logic course used the Deep Thought tutor, half with hints and half without. The sections included three sections in the Spring 2009 semester and three sections in the Fall 2009 semester. Each of three college philosophy professors taught one section each semester, one semester using Deep Thought with the Hint Factory and one semester using Deep Thought with no hints. This controlled for effects from different instructors by switching between the experimental and control conditions between the two semesters. Students in the Hint group classes could receive unlimited hints on the seven L1 and L2 problems that had them and the Control group received no hints throughout. Students

Table 3
Pre-test results and number of students with Deep Thought log files in each class and group. There were no significant differences across professors or groups

Prof.	Hint Sem.	Data	TOTAL # pre-test # DT logs	Hint # pre-test # DT logs	Control # pre-test # DT logs	Hint pre-test Mean (SD)	Control pre-test Mean (SD)
A	Spring	Pre	91	46	45	56.09 (25.82)	52.44 (17.96)
		DT	76	39	37		
B	Spring	Pre	74	36	38	47.22 (19.45)	47.44 (22.57)
		DT	62	29	33		
C	Fall	Pre	76	39	37	53.48 (18.41)	48.65 (17.55)
		DT	65	37	28		
All	Both	Pre	241	120	121	52.61 (21.93)	49.69 (19.38)
		DT	203	105	98		

generally completed the problems, in order, over the course of the semester, but could access the problems at any time.

At the beginning of the course students were given an overall-course pre-test that measured skills for the entire deductive logic course. Table 3 shows the total number of students in each class, listed by professor and group. The table lists the semester each professor gave the course with hints, the number of students participating in the pre-test and the number of students whose Deep Thought log files were available. This disaggregation shows that no section of the class had a disproportionate number of students participating in the pre-test or in the Deep Thought tutor. Below, we compare the pre-test scores to test for differences across classes, semesters, and groups.

To ensure that groups were not affected by Semester or Professor we performed a 2×2 (Semester $\times$ Professor) factorial analysis of variance for the pre-test. Results did not indicate a significant main effect for Semester, $F(1, 225) = 0.06$, $p = 0.81$ or for Professor, $F(2, 225) = 2.54$, $p = 0.08$. This shows a weak effect for professor, but this is mitigated by all professors teaching in both conditions. The interaction of Professor and Semester also showed no significant differences, $F(2, 225) = 0.83$, $p = 0.44$. Finally, the pre-test scores were submitted to a *t*-test to check for differences. The results showed no difference between the Hint group ($M = 54.88$, $M = 19.41$) and the Control group ($M = 51.85$, $M = 16.71$), $t(229) = -1.27$, $p = 0.21$. Figure 3 shows the mean pre-test scores and standard deviations for each group by professor (A, B, and C). Based on these pre-test results we can assume that students in each semester and professor started the semester with similar levels of performance.

RESULTS AND DISCUSSION

We tested the effect of hints on student performance by measuring the average percentage of correctly completed problems, according to level and group, as shown in Table 4. To investigate the differences in performance between groups, we submitted the results of L1, L2, and L3 to between-subjects two-tailed tests with an alpha of 0.05. The Hint group performed significantly better on all three levels, as shown in the last two rows of Table 4.

Figure 4 shows the percent of students who completed each problem for each group, illustrating that the groups had small differences in completion rates of 15 percent or less on problems 1.1, 1.2, and 1.4, but larger gaps on all the other problems.

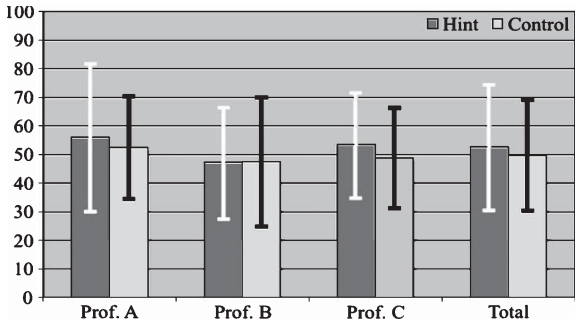

Fig. 3. Pre-test means by professor and group, with error bars to show one standard deviation.

Table 4
Completion rates; the Hint group completed significantly more problems in each level

Group	L1 % Complete (of 6 problems) Mean (SD)	L2 % Complete (of 5 problems) Mean (SD)	L3 % Complete (of 2 problems) Mean (SD)
Hint	78.2 (31.2)	67.4 (37.8)	59.0 (45.0)
Control	61.2 (36.2)	42.4 (40.8)	41.5 (46.0)
Two-tailed *t*-test	$t\,(191.82) = -3.58$	$t\,(196.71) = -4.52$	$t\,(201) = -2.78$
Significance	$p < 0.001$	$p < 0.001$	$p = 0.006$
Effect size	$d = 0.51$	$d = 0.64$	$d = 0.39$

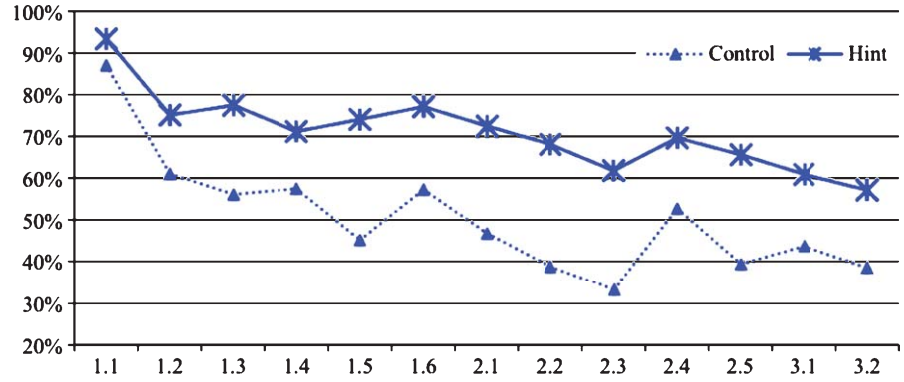

Fig. 4. Percent of students in each group who completed each problem in Deep Thought.

Next we examined the effects on student retention by comparing the number of problems attempted for each group. Table 5 shows the average percentage of problems attempted and standard deviation for each of the three levels. The attempt rates were calculated by student and level and divided by the total number of students. Students were given credit if they attempted to solve the problem, even if they did not find a solution (logs were flagged as solved when students solved a problem but data was collected even if a solution was not found). To investigate the differences in student retention we submitted the attempt rates for L1, L2, and L3 to between-subjects two-tailed tests with an alpha of 0.05. There was no significant difference between the attempt rates for L1, but the Hint group attempted significantly more L2 and L3

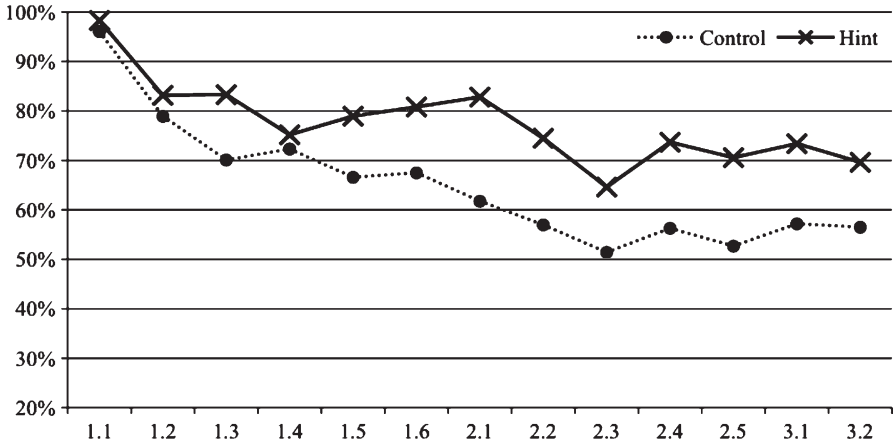

Fig. 5. Percent of students in each group who attempted each problem in Deep Thought.

Table 5
Percent of problems attempted by level. The Hint group attempted significantly more L2 and L3 problems when compared to the control group

Group	L1 % Attempted (of 6 problems) Mean (SD)	L2 % Attempted (of 5 problems) Mean (SD)	L3 % Attempted (of 2 problems) Mean (SD)
Hint	83.2 (27.2)	73.2 (35.0)	71.5 (41.5)
Control	75.3 (31.0)	56.0 (43.2)	57.0 (46.0)
Two-tailed *t*-test	No difference	$t\,(186.94) = -3.07$	$t\,(195.33) = -2.32$
Significance		$p = 0.002$	$p = 0.021$
Effect size	$d = 0.27$	$d = 0.44$	$d = 0.33$

problems, as shown in Table 5. Figure 5 shows the percent of students who attempted each problem for each group, illustrating that the groups had small differences in attempt rates of 5 percent or less on problems 1.1, 1.2, and 1.4, but larger gaps on all the other problems. Lower attempt rates occurred for both groups on problem 2.3, which was a newer, more difficult problem added to the problem set.

To further test the effects of hints on student performance and persistence we looked at the overall rate in which students abandoned the Deep Thought tutor after the first level (L1) as seen in Table 6. Students are considered to have dropped if they attempted problems in L1 but then did not attempt problems in L2 or L3. Twenty-eight percent of the Control group abandoned the tutor after L1 (classified as Dropped below), while only 9% of the Hint group stopped attempting Deep Thought problems. A chi-square test of the relationship between group (Hint, Control) and Dropout (Continued, Dropped) produced $\chi^2(1) = 11.05$, which is statistically significant at $p = 0.001$. This is associated with an odds ratio of 3.61, indicating that the odds of dropping after the first level are 3.61 times higher when the students are not provided hints. This meaningful difference suggests that retention rates for CAI tools may be greatly by improved by using the Hint Factory.

These results suggest that the use of automatically generated hints results in more persistent users and a higher retention rate in the Deep Thought tutor. The mechanism for this effect may lie in the ability of students who are frustrated to ask for more help in Deep Thought plus the Hint Factory, while students in

Table 6
Number of students that continued or dropped out of the tutor after L1

Group	Total	# Continued	# Dropped	% Dropped
Hint	105	95	10	9%
Control	98	71	27	28%
Total	203	166	37	18%

Table 7
Final course grades by group

Group	N	Mean	SD
Control	98	77.82	21.77
Hint	105	83.96	15.74

the Control group had no such alternative. Interestingly, we have observed students using the Hint Button for more than just asking for hints. The Hint Button is visible whenever a problem has hints, and is enabled when a specific hint is available (which is about 80% of the time). When the button is disabled, its label is grey and clicking on it has no effect. When students perform a step that does not already exist in our MDP, the Hint Button is disabled, and some students have observed this. After several steps, if the student gets stuck the student may want a hint but know that hints are not available. We have observed that, rather than give up, students will delete their steps until the Hint Button becomes available again. Since this effect was anecdotally observed during the experiment, data were not collected on how often this occurred. In cases where steps do not exist in our MDPs, it is often the case that the student has tried something correct, but unusual, that may not lead to a solution. We believe that some students may have learned this through experience with the Hint Button and are thus receiving some tacit hints that particular steps in their work are unusual or unorthodox. Since students often generate several steps when they don't know a good strategy for moving ahead in a logic proof problem, this tacit help, while not evaluative, may be adding up to gentle nudges in the right direction. This can be thought of like seeing a subtle worn path while walking in the forest – you can go another way, but some who've gone before have taken the path. It may be useful to explicitly inform students about how hints are generated, and suggest this strategy of deleting steps until the work they are doing matches a prior successful attempt in the Hint Factory.

To evaluate the learning effects of Hint Factory we subjected students' final course grades to an independent groups *t*-test. The final course grades for each group are given in Table 7. There was a significant difference $t(201) = -2.31$, $p = 0.022$, $d = 0.33$, indicating that students in the Hint group earned a higher overall course grade. Since we know that students in the Hint group tended to solve more Deep Thought problems, and these problems make up about a quarter of the course homework, the difference in the time spent in Deep Thought may account for final course performance.

To further investigate hint usage, we examined hint usage by problem. For the problems that have hints, Table 8 lists the number of first, second, third, and fourth hints that students used in solving completed problems. We also give the average number of correct steps taken in student solutions to each problem and the total number of students solving each problem who received hints. These data illustrate that the large majority of hints are depth 1 hints, indicating that a suggestion for what to do next was sufficient in most cases when help was needed.

Table 8
Average number of hints requested per problem and depth, computed by dividing the number of hints requested at each depth by the number of students requesting hints on each problem. "Steps" is the average number of correct steps taken in student solutions to each problem

Problem	Steps	Depth 1 hints	Depth 2 hints	Depth 3 hints	Depth 4 hints
1.1	5.55	1.88	0.89	0.59	0.24
1.2	6.19	2.56	1.36	0.87	0.56
1.3	8.9	2.94	1.70	1.14	0.70
1.4	10.62	3.50	1.72	1.17	0.42
2.2	13.39	4.42	2.58	2.15	1.07
2.3	9.68	6.63	3.09	2.33	1.02
2.4	5.12	1.70	1.03	0.89	0.49
2.5	9.55	4.06	1.83	1.51	0.79

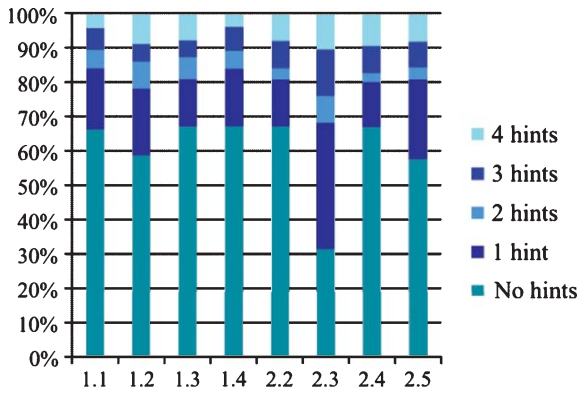

Fig. 6. Profile of hint usage by problem. Each bar is 100% of the average number of steps taken to complete the problem. The No hints portion represents the average percentage of steps students took with no hints. The 1 hint portion is the average percentage of steps achieved with 1 hint on the current step. The remaining portions represent steps where the hint depth was 2, 3, or 4 for that percentage of steps.

Figure 6 illustrates the proportion of steps taken in student proofs that use hints, broken down into groups according to how many hints were requested on a particular step. These results show that students used hints for about one-third of the steps taken in each problem, except for problem 2.3, where 69% of the steps taken needed hints. This reflects that problem 2.3 is more difficult than the other problems in the set (as measured by instructors, problem length, time, and number of hints requested). Even though hint usage was not restricted for the hint group, students did not abuse the ability to request bottom-out (depth 4) hints, requesting such hints in only 5–12% of the steps. Interestingly, in problem 2.3, having one hint was sufficient to help students along for about 35% of the steps. This suggests that the hints are scaffolding students to be able to solve a more difficult problem.

EFFECTS OF HINTS ON STUDENT STRATEGIES

To understand potential differences in problem solving strategies caused by automatically generated hints, we performed an in-depth analysis of student attempts on problem 1.4. Problem 1.4 is interesting since it

occurs right before an increase in Control group dropout (see Fig. 5), suggesting that this problem may lead a large number of students to abandon the tutor. This problem is also interesting as it was designed to teach a specific proof axiom: Constructive Dilemma. To explore this problem we generated the graph of all student problem-solving steps as explained above. We applied the Girvan-Newman algorithm to this network representation of student-log sequences to derive ten community clusters (Eagle et al., 2012), filtering out all edges with frequency less than five. We removed all clusters but those along the three main paths shown in Fig. 7. We labelled the five remaining clusters: Forward CD, Forward Simp, Addition Bugstrat (for buggy strategy), Backwards, and Backward Addition. These clusters form three main strategies: Forward (which contains the Forward Simp, Forward CD, and Addition Bugstrat clusters), Backwards, and Backward Addition. Students were spread roughly equally among these three strategies.

The orange edges in Fig. 7 indicate paths taken primarily by the Control group and blue edges are those taken primarily by students in the Hint group; brown edges are an even mix from both groups. The Backward Addition strategy is incorrect, as it does not lead to solving the problem. As shown in Fig. 7, the majority of students attempting this strategy are in the Control group. The Hint Factory does not have hints for the Backwards strategy, and both Control and Hint group students use this path, as shown in its intermediate color. The Hint group was much more likely to use the Forward strategy than the Control group. Light blue nodes represent states where many students requested hints. These nodes "route" the Hint group toward the correct solution. The Addition BugStrat represents a departure of students from the Forward strategy to a bad strategy, and students in the Control group are more likely to take this path. This analysis demonstrates that our hints can lead students to use a particular strategy. In this case, it appears that when students were stuck and asked for hints they were led to the Forward strategy, while students in the Control group tended to follow bad Backward Addition strategy, which will never lead to a solution.

We believe that this problem may have been pivotal in students' decisions to continue using the Deep Thought tutor. Without hints, students in the Control group were often unable to devise an effective strategy to solve this problem. Constructive dilemma is required to solve this problem in a small number of steps. This rule is particularly complex, and our results suggest that additional scaffolding can be important in helping students learn how to use this rule in logic proofs.

CONCLUSIONS AND FUTURE WORK

This research represents a rigorous study that demonstrates that using the MDP method and Hint Factory to add automatically generated hints to Deep Thought increased the attempt and completion rates and overall learning for students solving logic proofs, independent of instructor or semester. Our results showed that the Hint group had significantly higher completion rates for all three levels of problems. We also showed that, while the two groups were equally likely to attempt solving problems in level L1, the Hint group attempted significantly more problems in levels L2 and L3. Furthermore, students without hints were 3.6 times more likely to quit using the tutor altogether after the first level of problems. This important result shows that the automatically generated hints can help students to remain engaged with the tutor.

Using L3 as a post-test measure that both groups completed without hints after levels L1 and L2, we have shown an overall learning effect for students in the Hint group, who were able to complete significantly more problems. This means that having hints early on helps students later when hints are not available, and suggests that having hints improves overall learning of logic proof solving. An analysis of the number of steps taken for each problem and the number of hints used shows that students were not abusing hints, using them about one-third of the time. Finally, students in the Hint group scored significantly better in the overall course grade. This result is not surprising, since completing all the Deep Thought problems is a good predictor of success in the course. However, this research demonstrates a major finding that

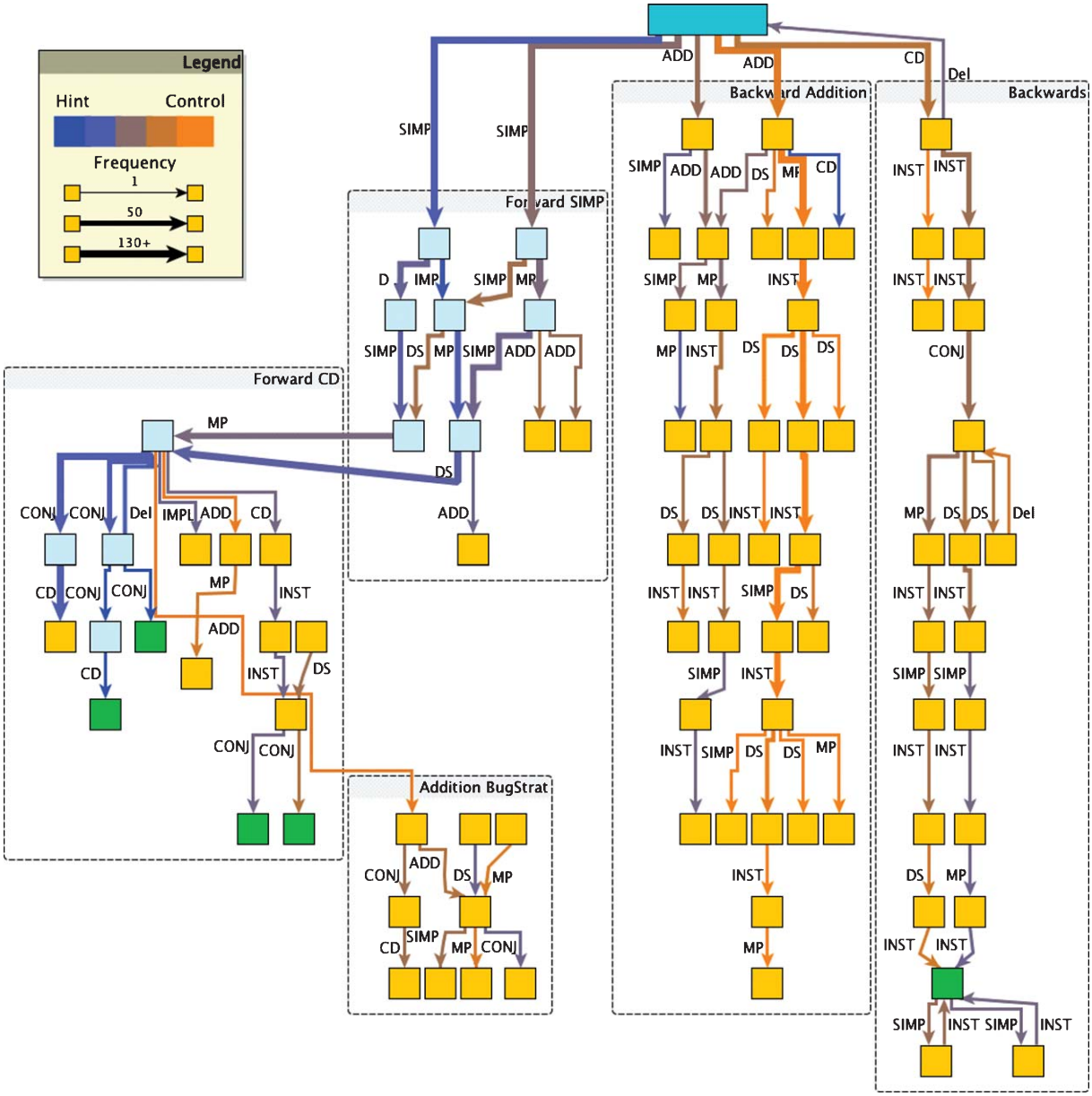

Fig. 7. Five main clusters of approaches to problem 1–4. Edge colors are by group, with each student in the Control group contributing some orange to the edge color and Hint group students contributing blue. Solution states are green nodes, and are only attained by students in the Forward CD and Backwards clusters. The Addition BugStrat and Backward Addition cluster are incorrect strategies.

automatically generated hints can be used to improve overall learning and retention for step-based problem solving.

There are limitations to this work. As we have suggested in prior work, hints could cause a "trail-blazing" effect where students do not fully explore the problem solution space. Therefore, we suggest that

hints always be used, but only on some problems, and that these problems should be rotated to ensure diversity of the student work. Results suggest that having hints available early is important.

In our future work, we plan to further analyse data for these six courses to more fully understand how students used hints. For example, detailed comparisons of hint usage and problem difficulty, and time on task can provide more insight into student learning and behavior. Our analysis of the impact of hints on learning is much less sophisticated than what might be done with knowledge tracing. We are inspired by Beck's excellent work on when to provide help based on his modification of Bayesian knowledge tracing (Stamper et al., 2010). The MDP method and Hint Factory together serve as a model tracer to provide hints for step-based problem solving. However, these methods do not explicitly model student knowledge. In the future we hope to compare methods for modelling knowledge in the logic domain using Bayes Nets and use these to create MDPs tailored to students at different knowledge levels. The observed behavior of deleting steps until the hint button was available suggests a new modification to the Hint Factory to be transparent about the source and mechanism for hints so students may use it better to their advantage. We also plan to extend our methods to create effective data-driven intelligent tutors for other problem solving domains.

ACKNOWLEDGMENTS

This work was partially supported by NSF grant IIS-0845997.

REFERENCES

Aleven, V., McLaren, B., Roll, I., & Koedinger, K. (2004). Toward tutoring help seeking: Applying cognitive modelling to meta-cognitive skills. *Proceedings of the Seventh International Conference on Intelligent Tutoring Systems*.

Barnes, T., & Stamper, J. (2008). Toward Automatic Hint Generation for Logic Proof Tutoring Using Historical Student Data. In E. Aimeur & B. Woolf (Eds.) *Intelligent Tutoring Systems (ITS 2008)* (pp. 373-382). Berlin, Germany: Springer Verlag.

Barnes, T., Stamper, J., Lehmann, L., & Croy, M. (2008). A Pilot Study on Logic Proof Tutoring Using Hints Generated from Historical Student Data. In R. Baker, T. Barnes & J. Beck (Eds.) *Educational Data Mining (EDM 2008)* (pp. 197-201). Montreal, Canada.

Beck, J., Chang, K., Mostow, J., & Corbett, A. (2008). Does Help Help? Introducing the Bayesian Evaluation and Assessment Methodology. *Intelligent Tutoring Systems*, 383-394.

Billingsley, W., & Robinson, P. (2007). Student proof exercises using MathsTiles and Isabelle/HOL in an intelligent book. *Journal of Automated Reasoning*, *39*(2), 181-218. Springer.

Croy, M., Barnes, T., & Stamper, J. (2007). Towards an Intelligent Tutoring System for propositional proof construction. In P. Brey, A. Briggle & K. Waelbers (eds.), In *Proc European Computing and Philosophy Conference* (pp. 145-155), Amsterdam, Netherlands: IOS Publishers.

Eagle, M., Johnson, M., & Barnes, T. (2012). Interaction Networks: Generating High Level Hints Based on Network Community Clusterings. In *Proceedings of the 5th International Conference on Educational Data Mining (EDM 2012)* (pp. 164-167). Chania, Greece.

Fossati, D., Di Eugenio, B., Ohlsson, S., Brown, C., Chen, L., & Cosejo, D. (2009). I learn from you, you learn from me: How to make iList learn from students. In V. Dimitrova, R. Mizoguchi, B. Du Boulay & A. Graesser (Eds.), *Proc 14th Intl Conf on Artificial Intelligence in Education, AIED 2009* (pp. 186-195). Brighton, UK: IOS Press.

Jin, W., Barnes, T., Stamper, J., Eagle, M., Johnson, M., & Lehmann, L. (2012) Program Representation for Automatic Hint Generation for a Data-Driven Novice Programming Tutor. In *Proc of the 11th Intl Conference on Intelligent Tutoring Systems (ITS2012)* (pp. 304-309). Springer.

Li, N., Matsuda, N., Cohen, W. W., & Koedinger, K. R. (2011). A Machine Learning Approach for Automatic Student Model Discovery. In M. Pechenizkiy, T. Calders, C. Conati, S. Ventura, C. Romero & J. Stamper (Eds.), *Proceedings of the 4th International Conference on Educational Data Mining (EDM2011)* (pp. 31-40).

Lukins, S., Levicki, A., & Burg, J. (2002). A tutorial program for propositional logic with human/computer interactive learning. *ACM SIGCSE Bulletin*, *34*(1), 381-385. ACM.

Koedinger, K., McLaughlin, E., & Stamper, J. (2012). Automated Student Model Improvement. In *Proceedings of the 5th International Conference on Educational Data Mining (EDM 2012)* (pp. 17-24). Chania, Greece.

Koedinger, K. R., Baker, R. S. J. d., Cunningham, K., Skogsholm, A., Leber, B., & Stamper, J. (2010) A Data Repository for the EDM commuity: The PSLC DataShop. In C. Romero, S. Ventura, M. Pechenizkiy & R. S. J. d. Baker (Eds.) *Handbook of Educational Data Mining*. Boca Raton, FL: CRC Press.

Mitrovic, A., Koedinger, K. & Martin, B. (2003). A comparative analysis of cognitive tutoring and constraint-based modelling. *User Modelling*, 313-322.

McLaren, B., Koedinger, K., Schneider, M., Harrer, A., & Bollen, L. (2004). Bootstrapping Novice Data: Semi-automated tutor authoring using student log files, In *Proc Workshop on Analyzing Student-Tutor Interaction Logs to Improve Educational Outcomes, 7th Intl Conf Intelligent Tutoring Systems (ITS-2004)*, Maceió, Brazil.

Nkambou, R., Mephu Nguifo, E., & Fournier-Viger, P. (2008). Using Knowledge Discovery Techniques to Support Tutoring in an Ill-Defined Domain. In E. Aimeur & B. Woolf (Eds.) *Intelligent Tutoring Systems (ITS 2008)* (pp. 395-405). Berlin: Springer Verlag.

Paquette, L., Lebeau, J.F., Mbungira, J., & Mayers, A. (2011). Generating task-specific next-step hints using domain-independent structures. In *Proceedings of the International Conference on Artificial Intelligence in Education (AIED 2011)* (pp. 525-527). Springer.

Razzaq, L. & Heffernan, N. (2010). Hints: Is It Better to Give or Wait to be Asked? In V. Aleven, J. Kay & Mostow, J (Eds). *Proc 10th Intelligent Tutoring Systems (ITS2010) Part 1* (pp. 349-358). Springer.

Razzaq, L., & Heffernan, N.T. (2006). Scaffolding vs. hints in the Assistment system. In Ikeda, Ashley & Chan (Eds.). *Proceedings of the Eight International Conference on Intelligent Tutoring Systems* (pp. 635-644) Springer-Verlag: Berlin.

Scheines, R., & Sieg, W. (1994). Computer environments for proof construction. *Interactive Learning Environments*, *4*(2), 159-169.

Shih, B., Kenneth R. Koedinger & Richard Scheines. (2008). "A Response Time Model For Bottom-Out Hints as Worked Examples." In *Proceedings of the 1st International Conference on Educational Data Mining*, EDM2008, pp. 117-126.

Shute, V. J. (2008). Focus on formative feedback. *Review of Educational Research*, *78*(1), 153-189.

Stamper, J., M. Eagle, T. Barnes, & M. Croy. (2011). Experimental Evaluation of Automatic Hint Generation for a Logic Tutor. *15th Intl Conf on AI in Education (AIED) 2011* (pp. 345-35). Auckland, New Zealand.

Stamper, J., Barnes, T., & Croy, M. (2011). Enhancing the Automatic Generation of Hints with Expert Seeding. *International Journal of Artificial Intelligence in Education*, Special Issue "Best of ITS", IOS Press.

Sutton, R. & Barto, A. (1998). *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

VanLehn, K. (2006). The behavior of tutoring systems, *International Journal of Artificial Intelligence in Education*, *16*, 227-265. IOS Press.

Yacef, K. (2005). The Logic-ITA in the classroom: A medium scale experiment. *International Journal of Artificial Intelligence in Education*, *15*(1), 41-60. IOS Press.