Adapting Tutoring Behavior to Student Needs
A Study in Intelligent Tutoring Systems

Aleksandra Zakrzewska, Aditi Ramachandran, Sarah Sebo, and Brian Scassellati

Abstract—Students greatly benefit from personalized one-on-one tutoring, but resources do not usually exist to provide each student with a human tutor. For this reason, Intelligent Tutoring Systems (ITS) are an attractive way to provide students with the personalized help they need to learn without straining teacher resources. As people tend to ascribe more human characteristics to robots than purely digital characters, and are more likely to follow their instructions, robot tutors, in particular, are a promising way to provide more students with the benefits of personal tutoring. While many ITS currently provide recommendations for further study material or can generate hints to complex problems, they do not yet deliver the level of personalized help of which human tutors are capable. Human teachers can tailor their actions and help strategies to individual students, so in order to best emulate them, robots must also be able to adaptively change their behaviors to suit the needs of students.

This paper proposes the design of a study to test such an adaptive robot tutoring agent and presents the implementation of an adaptive tutoring system. The platform of the tutoring system includes a robot and a connected tablet application, and will be used in the study. Students will do math problems on the tablet while getting assistance from the connected robot. The robot will, based on the decisions of the model, provide one of its possible tutoring behaviors (including giving a hint, a worked example, a interactive tutorial, or a break to the student) after each attempt to answer a question.

I. BACKGROUND
A. Robots as Adaptive Learning Agents

It is well known that students benefit from one-on-one learning [1], [2]. However, most schools lack the resources to provide every student with a tutor, and students must learn in large groups that do not allow for the personalization of lessons, teaching strategies and materials. Intelligent Tutoring Systems, or ITS, can be a substitute for human tutors by providing personalized computer-based teaching.

Many such ITS are simply computer programs with digital, on-screen characters. Research by Powers et al. shows that humans are likely to view embodied agents as more sociable than disembodied ones, and Bainbridge et al. show that people are more likely to comply with instructions when the agents are embodied [3], [4]. This is especially true in children, who are likely to attribute many human characteristics to robots [5]. Thus, the use of robots in ITS systems seems a way to perfectly balance the need for personalized teaching and the resource limitations of human teachers.

This has been shown to be true by recent studies. [6]. In fact Howley et al. show that in some situations, robot tutors can be more effective than human tutors, because, due to lower social status, they do not cause evaluation apprehension, which can prevent a student from asking for help [7]. In order to be effective tutors, robots must be adaptive and sociable. Saerbeck et al. show that robotic tutors who exhibit socially supportive behavior improve student learning and progress, and Kanda et al. show that stronger relationships are formed between children and robots when the robot is social than when it is not. [8], [9] Robots must also be adaptive and respond to student needs by exhibiting the right behavior at the right time. Merrill et al. state that one key feature of successful human tutors is that they employ a highly interactive and indirect style to point out mistakes and allow the student a second chance to succeed. They must both give students the freedom to figure things out on their own and also prevent excessive floundering [10]. This is a challenge robot tutors must address.

B. Related Work in ITS

There have been many recent developments in online teaching and tutoring systems.

1) Models: Bayesian Knowledge Tracing, or BKT, is a common technique in ITS used to infer how well a student has mastered a topic. It is a hidden Markov model which finds the likelihood that a student understands a concept. Given the sequence of observed correct or incorrect answers and an assumed probability of mistakes or correct guesses, it infers the hidden knowledge states (unlearned or learned) [11]. Knowledge of what concepts a student does or does not understand can be used to suggest future learning tasks or to select questions best suited for continued learning.

Much recent work has involved improvements to this method, such as by parameterizing the mistake and guess probabilities per individual piece of knowledge [12]. Another version, aimed to include the effect of a tutoring system on these parameters, added a set of observable inputs specifying the Instructional Intervention at each point. These were defined as either elicit (asking a leading question about the next step) or tell (giving the student the next step) [13]. Other studies have used BKT methods to calculate the degree to which a tutoring behavior truly aided learning [14].

While understanding of the degree to which a student understands a concept is necessary to decide whether or not to provide help, there have not been meaningful attempts at developing a model to decide how a tutor can help to be most beneficial.

2) Generating help: Tutors must often provide a helpful intervention in the middle of a problem, so ITS research has focused a lot of attention on improving techniques to generate hints and examples to complicated problems. Some
of these systems generate examples from final answers or incomplete solutions of other learners by determining which parts are relevant to the challenge faced by a particular student. [15] Recent work on hint generation in open ended problems with multiple approaches have used correct solutions from other students to search for those with a shortest edit path to a final, completed answer. Then, these examples are used to provide small, relevant hints to the students, to push them further in the direction of the chosen solution [16], [17]. Once it is possible to generate the content of hints for a specific problem, however, it will still be necessary to decide when and how to provide these hints.

C. Previous Work in the Yale Social Robotics Lab

Most of the previously discussed work has been for ITS in general, without focus on an embodied robot, and most do not involve exhibiting any non-task behaviors. In the Yale Social Robotics Lab, Aditi Ramachandran has been working extensively to develop adaptive robotic tutors. Her previous studies in this area have investigated how robots can improve and support learning. In a study which developed adaptive strategies to improve help-seeking behavior, she found that students who interacted with a robot tutor which used adaptive hint-giving strategies performed better than those who could ask for and receive help on demand [18]. While breaks are known to be generally helpful for learning, she has found that adaptively timing breaks to account for students with different attention spans improves their effectiveness [19], [20]. In her most recent work, a robot encourages students by asking them to “think out loud”.

This study will bridge these pieces of adaptive robot behavior. Human tutors must, on top of being able to generate hints and choose helpful exercises, be able to perform the actions that best help an individual student. They must evaluate how well their students respond to any tutoring behavior they exhibit to select the most appropriate one. In order to be a viable alternative to human tutors, robot tutors must exhibit this same adaptive behavior. In this study, we develop an adaptive robot tutoring system that personalizes the selection of various types of tutoring behaviors to individual students.

II. Experimental Design

This study will conducted beginning in late January. When it does so, it will look like this:

A. Experimental Setup

The students in the study will participate in a series of tutoring interactions with the robot tutor. During each session, they will be solving division problems on a tablet. The robot will sit on the desk beside them, providing hints, informing the student whether their answers are correct, and exhibiting other tutoring behaviors. Students will complete a test after each session which will be used to determine how beneficial the sessions were.

There are five levels of questions, spanning a wide range of difficulties. Students will begin with level one, which includes only simple division facts with a two digit numerator and one digit denominator, and progress to the next level if they have successfully answered enough questions without help. Level five, the hardest level, contains three to five digit numerators, two digit denominators and answers may have remainders.

Figure 1 shows the basic tablet interface.

B. Participants

Participants will be recruited from a fourth grade class from a local New Haven elementary school. They are students who are currently learning in class the material with which the robot tutor will help.

C. Robot

This study will use a NAO robot as a tutoring agent. The robot will sit next to the student on the desk, both providing help and acting in a social manner. In the first session, the robot will introduce itself, and in future sessions will behave...
as if it knows the child, using phrases like “great to see you again”. The robot will respond to input in the tablet, congratulating students on correct answers (e.g. saying “great job!” or “well done!”) and encouraging them to try again when they are incorrect (with phrases such as “That’s not quite right. Try again!” or “That was a hard one. Why don’t you try again?”).

During tutoring behaviors such as worked examples, the robot will talk through the solutions to problems. All utterances are accompanied by simple gestures.

D. The Behavior Model

Sarah and Aditi are still completing work on the model that will determine which behavior the robot tutor should exhibit at each time. Students will be given three opportunities to answer a question. On the first attempt, they will receive no help. On the second and third attempts, the robot tutor will exhibit one of the following behaviors, as determined by the model.

1) No Action: Just like a human tutor, the robot can allow the student to try to answer the question again with no intervention. In this case, the robot simply says a phrase such as “Try that one again.” and provides no other assistance.

2) Hints: The robot can provide a hint if the student is stuck. Hints appear in two different ways. Hints for harder questions with longer numerators (levels 2 and up), show the boxes under the division problem that are filled in with intermediate answers. This structure is shown in Figure 2. This will be referred to as the box structure of a problem. If a second hint appears for the same problem, some of the boxes will be filled in.

Hints in level one, which contains only simple fact based division questions for which the box structure is not helpful give the multiplication facts which imply the division facts (e.g. for the problem 24 ÷ 4, the fact 4 * 6 = 24 may appear).

3) Worked Examples: To increase understanding of a concept, the robot tutor can show a worked example. A worked example begins with a problem of similar difficulty to the one on which the student is working. The robot walks the student through a solution while it appears on the screen. This occurs by filling in the box structure of the problem, in the case of difficult questions, or by showing the related multiplication and division facts for easy questions. Worked examples require no input from the student, and rely purely on the robot tutor teaching.

4) Interactive Tutorials: Interactive tutorials require input from both robot tutor and student. As with other help, tutorials come in two different modes for level one questions and the rest. Both begin with a problem of similar difficulty to the one on which the student is stuck. Level one tutorials are composed of an illustration of the problem in balls and boxes. The student is asked to fill in the answers to the multiplication and division facts illustrated. (Figure 4)

Later tutorials again show the box structure and ask the student to fill it in one step at a time. (Figure 5)

Both versions of the tutorials require the students to check their answers to move on to the next step. If they answer a step incorrectly more than 3 times, that step of the problem is filled in for them so that they may progress to the next step.

5) Think Aloud: For the think aloud behavior, the robot simply asks the student to say what the first step in solving the problem will be. This behavior is intended to make the student verbalize their thought process, and follows the recently completed study about the benefits of speaking out loud to a robot during tutoring sessions.

6) Breaks: Finally, the model may trigger a break for the student. Breaks are helpful for learning because people, especially children, have low attention spans [19]. A previous study in the lab also showed that adaptive break timing was beneficial to students in tutoring sessions [20]. For the break behavior, this study will use the same Tic-Tac-Toe game as in the previous version of the study. This allows the student to take a break while also allowing the robot to exhibit social behavior.

III. IMPLEMENTATION

This section can be read as a description of my contributions to this study, and a general overview of the design of each component. For more detailed technical information, please see the repository README files in the supplementary materials.

A. Components

The tablet application is an Android application. The model, server (which I will call the tablet node, as it is the way through which the other components must communicate with the tablet), and robot controller are all implemented as ROS nodes, which communicate with each other through several ROS topics. The tablet node receives direction from the model and controls the functions of the robot and tablet.

B. Message Passing

1) Between Model and Server: The model node and tablet node communicate via ROS messages in two different ROS topics.

The tablet node sends messages to indicate the status of the student. Specifically, the tablet node sends a message to indicate that the student has begun the next question (this would allow the model to record the amount of time a student requires for an answer) and whether or not the student has answered a question correctly. In response to information about correct or incorrect responses, the model sends a message to indicate the next step that should be taken next. Thus, it can send a message instructing the system to:

1) Go on to the next question
2) Go on to the next level
3) Try the same question again with no help
4) Start a Tic-Tac-Toe break
5) Show a hint
6) Prompt the student to think aloud
7) Begin an interactive tutorial
8) Show a worked example
Generally, the model will respond to a correct answer by either instructing to go on to the next question or to progress to the next level. On the first two incorrect answers, the model will provide a tutoring behavior. On the third it will allow the student to go on to the next question anyway.

2) **Between Server and Application:** The server (aka tablet node) and tablet application communicate via TCP messages. All tablet behavior is controlled via the server. There are several base types of messages the server will send to the tablet. They are as follows:

1) “QUESTION;(level):(number);” messages instruct the tablet to prepare to display the specified question from the specified level. The tablet application has direct access to the JSON files containing the questions, and so only this information is needed.

2) “SHOWSTRUCTURE;” messages instruct the tablet to show the box structure for a problem. If sent with no other arguments, the structure for the current question will be shown. If the message has the format “SHOWSTRUCTURE:numerator-denominator”, then the structure displayed will be for the provided example problem. If, additionally, the message has the keyword “TUTORIAL”, the structure will be formatted to be ready for a tutorial. The difference between these cases is described later.

3) “FILLSTRUCTURE;” messages cause the tablet application to fill in the answers to the structure boxes for the student. This message can additionally contain a list of boxes to be filled with their answers, in the format “lineNumber-boxNumber-answer” for each box, separated by colons. If the boxes to be filled in are not indicated in the message, all boxes that are enabled and editable are filled. (This can occur if the student is stuck on a part of a tutorial and needs to progress to the next step.) Otherwise, only those that are indicated in the message are filled with the provided answers.

4) “SHOWTEXTHINT;” messages cause the tablet to display whatever text appears after the semicolon in the hint panel as text.

5) “SHOWEASYTUTORIAL;” must provide a numerator and denominator for the tutorial and prompt the creation of a Level 1 tutorial, in which numerator many balls are divided between denominator many boxes.

6) “TICTACTOE;” messages are sent to start and continue playing a Tic-Tac-Toe game. The first Tic-Tac-Toe message triggers the start of a new Activity in the application that handles the game play.

7) “SPEAKING-START” messages indicate that the robot is speaking. This causes all buttons and inputs to be disabled.

8) “SPEAKING-DONE” messages indicate that the robot is finished speaking, so buttons and inputs are enabled after this message is received.

All model instructions are translated by the server node into a series of messages like those above.

In response, the tablet application may respond with the following types of messages:

1) “START” indicates the beginning of the tutoring session.

2) “CA” or “IA” messages indicate correct and incorrect messages respectively. This information is passed on to the model.

3) “TUTORIAL-STEP;” messages indicate whether the answers to the current step are correct, incorrect, or incomplete. This helps the server control the flow of the tutorial.

4) “TICTACTOE;” messages are used to control the game and control what the robot says during it.

5) “SHOWING-QUESTION” messages indicate that the next question has been shown to the student. This is necessary to send because the next question does not appear until after the student has pressed a button to progress to the next question, which could be some time after the information about the next question was sent to the tablet.

3) **Between Server, Model and Robot Controller:** The robot controller receives messages both from the tablet node and the model. When applicable, the robot can produce an utterance solely from the model instruction (e.g. in the case of the Think Aloud behavior, the robot simply asks the student to think out loud), but in other cases, it needs extra information from the tablet node. For instance, the robot waits until it receives a message indicating the next question has been shown before it says the question out loud. The message structures sent by the tablet node and model node both contain fields that designate robot speech.

The robot controller node publishes exactly two kinds of messages and it publishes them only to the tablet node. It specifies when it begins speaking and when it has finished speaking. There are two ROS topics to which it may publish these messages. One is used normally, and prompts the tablet to disable/enable buttons. The other is used only while running worked examples. The tablet node will use this topic to synchronize the appearance of example steps on the screen by only issuing the next instruction to the tablet when the robot has finished talking.

### C. Hints

The main type of hint provided in the tutoring system is the display of the box structure for a problem. The current problem appears again on the help panel with the boxes displayed underneath. These boxes are editable, so students can work through the problem directly on the screen. This hint is shown in Figure 2.

If a second hint is triggered for the same problem, the boxes corresponding to a first step of the solution are filled in.

The number of boxes displayed is variable and is based on the number of digits in the numerator and denominator.

These behaviors are triggered by messages of the “SHOWSTRUCTURE” and “FILLSTRUCTURE” types.

In level one, where such structure hints are not helpful, text based hints are provided instead.
D. Worked Examples

In a worked example, the partial answers to parts of the problem must be provided in order and in coordination with robot tutor activity. A worked example begins by displaying the box structure for a similar problem, followed by a series of “FILLSTRUCTURE” messages containing small steps of the problem. For each step, the robot tutor explains what is happening (e.g. saying “6 goes into 45 7 times, so we can subtract 42 from 47.”). The robot controller node publishes its speech to a distinct topic during examples, so that the tablet node can wait until it has finished describing the previous step before sending the next one to be displayed on the tablet. It appears as if the robot is placing the answers on the screen as it speaks.

In order to do this, the intermediate division steps must be provided. Creating these manually would be an arduous process, so I have written a script to generate them provided only a numerator and denominator. The number of boxes and placement of numbers depends on the length of each and on previous values generated. The recursive function generates a list of strings, each providing one step of the solution. They have the format “lineNumber-boxNumber-value: ...”. These strings can be directly sent to the tablet in a “FILLSTRUCTURE” command.

While an example is running, all buttons on the application screen are disabled to prevent the student from ignoring the robot tutor and attempting to move on.

As with other types of help, worked examples in level one are more simple. Rather than running through the box structure solution, the robot tutor presents a similar problem and explains the multiplication fact that leads to the division problem. These steps appear on the screen as pure text. The communication patterns between the tablet node, robot controller node and tablet application are the same as in harder examples.

Both types of examples are visible in Figure 3.

E. Interactive Tutorials

Interactive tutorials are meant to guide the student through a similar problem. Both level 1 and later tutorials ask the student to do some work and then check their answers before moving on to the next step.

1) Level 1: Easy, fact based tutorials are triggered by the “SHOWEASYTUTORIAL” message with a provided numerator and denominator. This causes the tablet to show the tutorial problem in the help panel with an illustration. Denominator many boxes appear with numerator many balls split between them. Below there are two input fields for the student to fill in. The first is the multiplication problem showing denominator * quotient = numerator and the second is the problem itself with numerator / denominator = quotient. In both, the child is asked to fill in the quotient. When the student presses the “Check Answers” button, a “TUTORIAL-STEP” message is sent to the server. The boxes change color depending on correctness, and the robot tutor responds, either prompting the child to try again or congratulating it on a correct answer. Figure 4 shows easy tutorials in use.

2) Levels 2 - 5: Harder tutorials again rely on the box structure, but it is formatted differently for interactive tutorials. When the box structure appears, only the boxes corresponding to the first step are enabled. The boxes that
are disabled are gray to indicate visually that they cannot accept input. The student is asked to fill in all of the boxes enabled for the current step and then check their work. See Figure 5 for such tutorials at various stages of completion.

As in easy tutorials, when a student presses “Check Answers” a “TUTORIAL-STEP” message is sent and both the robot tutor and tablet respond to the provided answers. As the box structure may display boxes that are not strictly necessary, an empty box is accepted if the box should contain a zero (e.g. if subtracting two 2 digit numbers gives a one digit result, the student need not enter ‘0’ and ‘3’ into two boxes to indicate a remainder of 3).

In both levels of tutorial, if the child enters an incorrect or incomplete answer enough times, the answer is filled in for them, and they are encouraged to move on.

F. Breaks

Tic-Tac-Toe games are used as breaks during the tutoring session. This provides continuity from a previous ITS study conducted which focused on adaptive break timing [20]. For this reason, I did not implement the code for this activity.

Tic-Tac-Toe games are played primarily directly on the tablet application, with messages passing between the application and the tablet node. The tablet node relays these to the robot controller to simulate a real game between the robot tutor and student.

The Tic-Tac-Toe game interface is shown in Figure 6.

IV. Future work

This study is not yet complete. Aditi and Sarah are finishing work on the model which will adaptively decide which behaviors occur after each answer attempt. When the model is completed, it will only have to send a message.
indicating its decision at each point to the tablet ROS node in order for the action to be implemented correctly.

They will begin collecting data from participants in late January.

V. CONCLUSIONS

This study will investigate the use of an adaptive robotic tutoring agent in the context of teaching division. An adaptive robot tutor who can respond to individual student needs by providing the right amount and right form of assistance will be a great step forward in ITS. For this project, I created the platform of the tutoring system, including a tablet application, server and robot controller code. This platform implements the tutoring behaviors that will be used in the study and has the capacity to react to the model’s decisions about when to activate each behavior.

VI. SUPPLEMENTARY MATERIALS

Code and documentation for the tablet application can be found at: https://github.com/ScazLab/nao_tutoring_behavior_tablet_app

Code and documentation for the ROS node controllers can be found at: https://github.com/ScazLab/nao_tutoring_behavior_ros_nodes

ACKNOWLEDGMENT

Thank you to Aditi Ramachandran, Sarah Sebo, Brian Scassellati and everyone in the Yale Social Robotics Lab for all of their support and work in this project.