Exploring Contingent Step Decomposition in a Tutorial Dialogue System

Pamela Jordan, Patricia Albacete and Sandra Katz
Learning Research & Development Center
University of Pittsburgh
Pittsburgh, USA
pjordan@pitt.edu, palbacet@pitt.edu, katz@pitt.edu

ABSTRACT

We explore the effectiveness of a simple algorithm for adaptively deciding whether to further decompose a step in a line of reasoning during tutorial dialogue. We compare two versions of a tutorial dialogue system, Rimac: one that always decomposes a step to its simplest sub-steps and one that adaptively decides to decompose a step based on a student's pre-test assessment. We hypothesize that students using the two versions of Rimac will learn similarly but that students who use the version that adaptively decomposes a step will learn more efficiently. Our initial results suggest support for our hypothesis but the sample size for the experiment is still small and we are continuing to collect more student interactions with the two versions of the system.

CCS Concepts

•**Applied computing** → **Interactive learning environments;** •**Computing methodologies** → *Discourse, dialogue and pragmatics;*

Keywords

Adaptation; Dialogue; Contingency; Scaffolding; Intelligent Tutoring Systems

1. INTRODUCTION

Woods introduced the idea of contingent tutoring in the 1970s after analyzing face to face interactions between children (the learners) and adults (the tutors) [7]. Instructional contingency refers to the amount of help or scaffolding the tutor offers the learner based on the student's current or previous response, while domain contingency refers to the issue of what the tutor should focus on next (e.g., what content in the current task, what the next task should be, what materials to use) and can involve deciding how to decompose a difficult task into potentially easier sub-tasks [7]. There are also different ways in which to adapt to a student which have been explored using tutorial dialogue systems, including adapting to learning style [6] and deciding who should cover a step in the tutoring [2]. However, in our current implementation of a tutorial dialogue system for physics, Rimac [5, 1], we focused on deciding when to decompose a task for the learner (an aspect of domain contingency), which includes: (1) deciding whether to decompose the reasoning needed to answer a post-problem reflection question (RQ), as in Figure 1, and (2) the granularity of that discussion.

Problem:

Suppose you aim a bow horizontally, directly at the center of a target 25.0 m away from you. If the speed of the arrow is 60 m/s, how far from the center of the target will it strike the target? That is, find the vertical displacement of the arrow while it is in flight.

Assume there is no air friction.

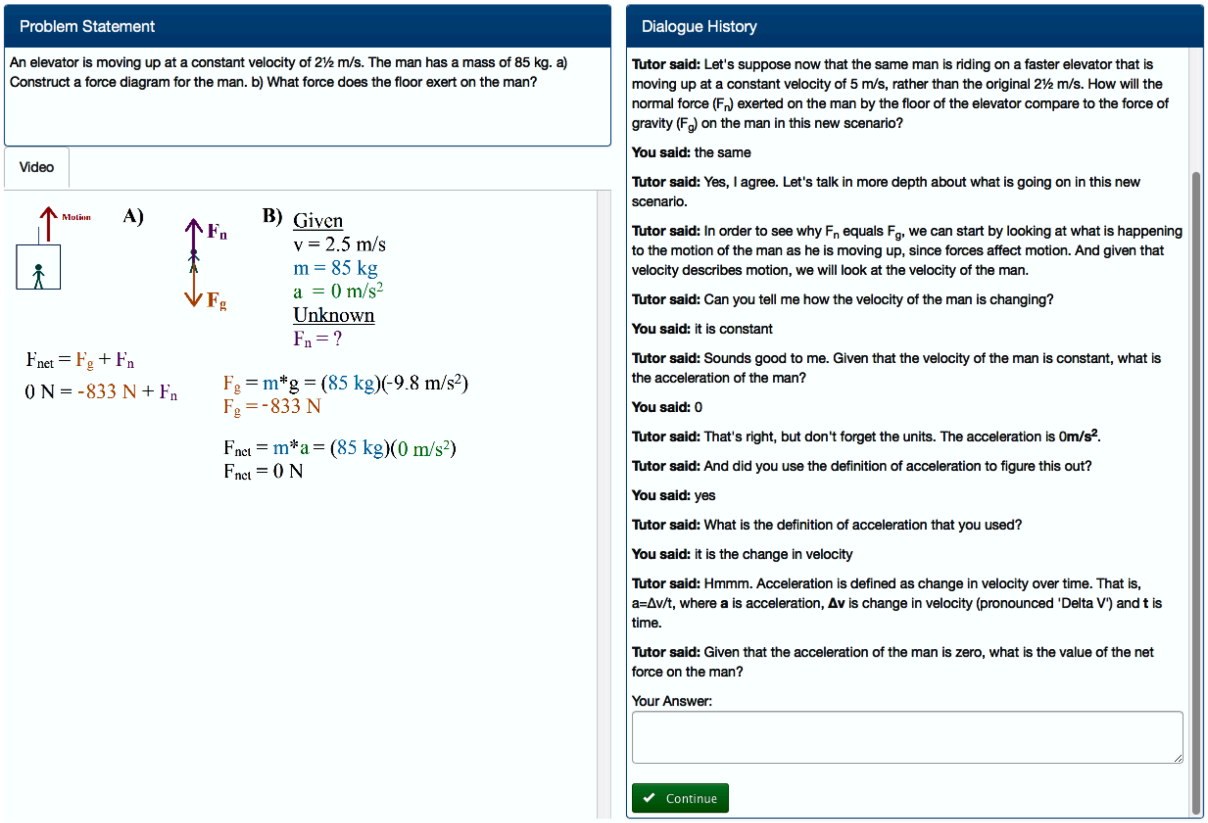
Reflection Question (RQ):

Suppose the same archer shoots an identical arrow from the same spot on the cliff. Again he aims the arrow perfectly horizontally with an initial velocity of 60 m/s. How does the vertical velocity of the arrow change (remains the same, increases, decreases)?

Figure 1: An example problem and post-problem reflection question.

Similar to Wood's EXPLAIN, QUADRATIC and DATA tutors [7], Rimac decides whether to discuss the line of reasoning (LOR) underlying a correct answer to an RQ and, if so, at what grain size (i.e., it decides whether to decompose a step in a task into simpler sub-steps). And similar to Wood's DATA tutor, Rimac bases decomposition decisions on pre-test assessments. Unlike Wood's tutors, help seeking is not left to the learner in that the tutorial dialogue system and the student are engaged in a discussion of the line of reasoning (LOR) that leads to the answer to a reflection question and the system always helps the student co-construct the next step in the LOR. To help the student co-construct the step, Rimac uses hint strategies to elicit the step from the student. If the hint fails, and the student is unable to co-construct the step, then the system either offers a more specific hint, decomposes the step further and hints at each of its sub-steps, or simply completes the step for the student.

In this paper we explore an initial, simple algorithm for adaptively deciding whether to further decompose a step after it has been successfully co-constructed. We compare two versions of Rimac: one that always decomposes successfully co-constructed steps and one that adaptively decides whether to decompose such a step based on students' pre-test assessment. The reason for decomposing a successfully co-constructed step is that the student may have contributed a correct answer using incomplete reasoning or may have simply guessed correctly using intuition and thus it could be beneficial to explicitly cover the underlying reasoning with the student. We hypothesize that if our simple algorithm is



An elevator is moving up at a constant velocity of 2½ m/s. The man has a mass of 85 kg. a) Construct a force diagram for the man. b) What force does the floor exert on the man?

Given
$v = 2.5$ m/s
$m = 85$ kg
$a = 0$ m/s^2
Unknown
$F_n = ?$

$F_{net} = F_g + F_n$
$0\ \mathrm{N} = -833\ \mathrm{N} + F_n$

$F_g = m*g = (85\ \mathrm{kg})(-9.8\ \mathrm{m/s^2})$
$F_g = -833\ \mathrm{N}$

$F_{net} = m*a = (85\ \mathrm{kg})(0\ \mathrm{m/s^2})$
$F_{net} = 0\ \mathrm{N}$

Tutor said: Let's suppose now that the same man is riding on a faster elevator that is moving up at a constant velocity of 5 m/s, rather than the original 2½ m/s. How will the normal force (F_n) exerted on the man by the floor of the elevator compare to the force of gravity (F_g) on the man in this new scenario?

You said: the same

Tutor said: Yes, I agree. Let's talk in more depth about what is going on in this new scenario.

Tutor said: In order to see why F_n equals F_g, we can start by looking at what is happening to the motion of the man as he is moving up, since forces affect motion. And given that velocity describes motion, we will look at the velocity of the man.

Tutor said: Can you tell me how the velocity of the man is changing?

You said: it is constant

Tutor said: Sounds good to me. Given that the velocity of the man is constant, what is the acceleration of the man?

You said: 0

Tutor said: That's right, but don't forget the units. The acceleration is 0**m/s²**.

Tutor said: And did you use the definition of acceleration to figure this out?

You said: yes

Tutor said: What is the definition of acceleration that you used?

You said: it is the change in velocity

Tutor said: Hmmm. Acceleration is defined as change in velocity over time. That is, a=Δv/t, where **a** is acceleration, **Δv** is change in velocity (pronounced 'Delta V') and **t** is time.

Tutor said: Given that the acceleration of the man is zero, what is the value of the net force on the man?

Figure 2: The Rimac Interface and an example dialogue with short answer questions.

effective then students will learn similarly from using either version of the system but that students who use the adaptive decomposition version will learn more efficiently.

The rationale for the hypothesis follows. First, students using either version of the system can spend as much time as they need to complete the assigned problems. If the student fails to successfully co-construct a decomposable step, then the system will respond by eliciting its sub-steps. However, if the student succeeds at co-constructing the step then the student can progress faster through the RQ. If the decision algorithm is successful, then the adaptive system should enable a significant number of users to complete the problem faster because it will often be accurate in its choice not to decompose a step after it is successfully co-constructed. Furthermore, if a significant number of steps are not decomposed after a successful co-construction, then less material is explicitly covered with the student. If it is not detrimental to have "skipped" explicit mention of this material then learning gains for students who used the adaptive system should be similar to learning gains for students who used the non-adaptive system.

While our initial results suggest support for this hypothesis, the sample size is still small and we are continuing to collect more student interactions with the system.

2. RIMAC

Rimac is a web-based natural-language tutoring system that engages students in conceptual discussions after they solve quantitative physics problems [5, 1] and was built using the TuTalk tutorial dialogue toolkit [4]. Thus the dialogues authored for the system can be represented with a finite state machine. Each state contains a single tutor turn. The arcs leaving the state correspond to possible classifications of student turns. When creating a state, the dialogue author enters the text for a tutor's turn and defines classes of student responses (e.g. correct, partially correct, incorrect). A single student response class is defined by entering a set of semantically similar text phrases that correspond to how students might respond. TuTalk's default understanding module ranks the response classes defined for the current tutor state according to the edit distance of the normalized words in the actual student response relative to the normalized words in the text phrases that define each class. It selects the class with the minimum edit distance as the best classification of the student's response. However, if the minimum edit distance is greater than a specified threshold, then the system classifies the student response as unrecognizable.

Rimac's dialogues were developed to present a directed line of reasoning, or DLR [3]. During a DLR, the tutor presents a series of carefully ordered questions to the student. If the student answers a question correctly, he advances to the next question in the DLR. If the student provides an incorrect answer, the system launches a remedial sub-dialogue and then returns to the main line of reasoning after the sub-dialogue has completed. If the system is unable to understand the student's response then it completes the step for the student. Rimac asks mainly short answer questions to improve the recognition of student responses as shown in Figure 2, which illustrates the system's follow-up to correct, partially correct and incorrect answers.

Rimac's dialogues are structured as hierarchical plan networks where a parent node abstracts over its child nodes [8]. For example, a parent node of "travel to Chicago" may be decomposed into more detailed child nodes such as "buy an airplane ticket to Chicago", "go to the airport", etc. which in turn may be decomposed into even more detailed nodes. In the case of tutoring physics, the upper-level parent nodes represent the problem solving strategy. See Figure 3 for an example of part of a plan network for one of the Rimac dialogues we are using in our testing.

The adaptive version of Rimac uses a decision algorithm to decide whether, after eliciting a parent node, to expand the parent node and elicit its child nodes. For this formative evaluation of the algorithm, we selected the nodes where decisions should be made instead of treating each non-leaf node as a potential decision point.

For example, in reference to the plan network in Figure 3, both example dialogues in Figure 4 first elicit the top child nodes of "(2) Determine net force" and "(3) Determine vertical acceleration" for the parent node "(1) Solve RQ". Notice that there are further decisions to make concerning how to elicit each node. When eliciting "(2) Determine net force" the system elicits one of the child nodes "(4) Identify forces" instead of directly eliciting "(2) Determine net force". For this experiment we left the decision about how to elicit each node to our content specialists and this was static and identical across both versions of the system.

Neither of the child nodes "(2) Determine net force" and "(3) Determine vertical velocity" is expanded further in the dialogue example in Figure 4 (left), which was generated by the adaptive version of the system. Instead, the dialogue moves on to elicit a new sibling node not shown in the plan network. However, in the dialogue example on the right in Figure 4, the system decides to expand all decomposeable nodes further [i.e., "(2) Determine net force" and "(3) Determine vertical acceleration"]. The decision about whether to elicit one node or multiple nodes before expanding those nodes is again left to the content specialist and is static and identical across both versions of the system.

Thus, the dialogue for the adaptive version of the system would range between that shown by the dialogue on the left in Figure 4, where none of the target parent nodes is expanded, and that shown by the dialogue on the right where the algorithm decides to expand every target parent node.

3. CONTINGENT STEP DECOMPOSITION

In the adaptive version of Rimac that we are testing, we use a student model that is initialized with the student's pre-test scores for the knowledge components (KCs) that need to be applied to arrive at the correct answer to the reflection questions presented to students. In future versions of the system (but not in this current test) we will update the student model during the discussions with the tutor in an attempt to reflect students' learning.

The adaptive version of the system consults the student model at every decision point to predict whether the student is likely to need the current step decomposed into simpler steps. Two types of decision points occur: (1) after a reflection question (RQ) is answered by the student and (2) when it is possible to further decompose a step into substeps. In the former case the reflection question is the top node in the plan network and is decomposed by engaging in a discussion of the reasoning with the student (i.e., it elicits some subset of child nodes). For every decision point a set of prerequisite KCs have been identified that are expected to predict whether the student sufficiently knows the knowledge expressed in the child nodes (sub-steps). The student's scores for that set of KCs are evaluated to decide whether or not to decompose the node (step) further.

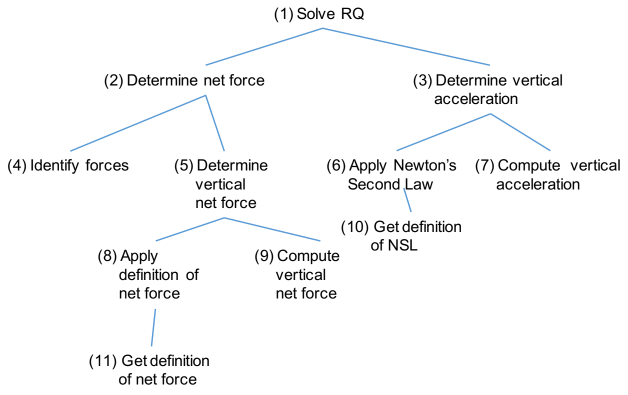

(1) Solve RQ
(2) Determine net force
(3) Determine vertical acceleration
(4) Identify forces
(5) Determine vertical net force
(6) Apply Newton's Second Law
(7) Compute vertical acceleration
(10) Get definition of NSL
(8) Apply definition of net force
(9) Compute vertical net force
(11) Get definition of net force

Figure 3: Extract of plan network for responding to the RQ in Figure 1.

Let KC_D be the set of KCs associated with decision point D where $KC_d \in KC_D$, a_i is the score $\in \{1, 0\}$ for a pre-test item that tests KC_d and n is the number of test items testing KC_d. Let S_D be the set of scores for KCs associated with decision point D where $S_d \in S_D$, S_d is the score for KC_d and S_d is defined as:

$$S_d = 1/n \sum_{i=1}^{n} a_i \tag{1}$$

Finally, let T_D be the score for decision point D where T_D is defined as:

$$T_D = min(S_D) \tag{2}$$

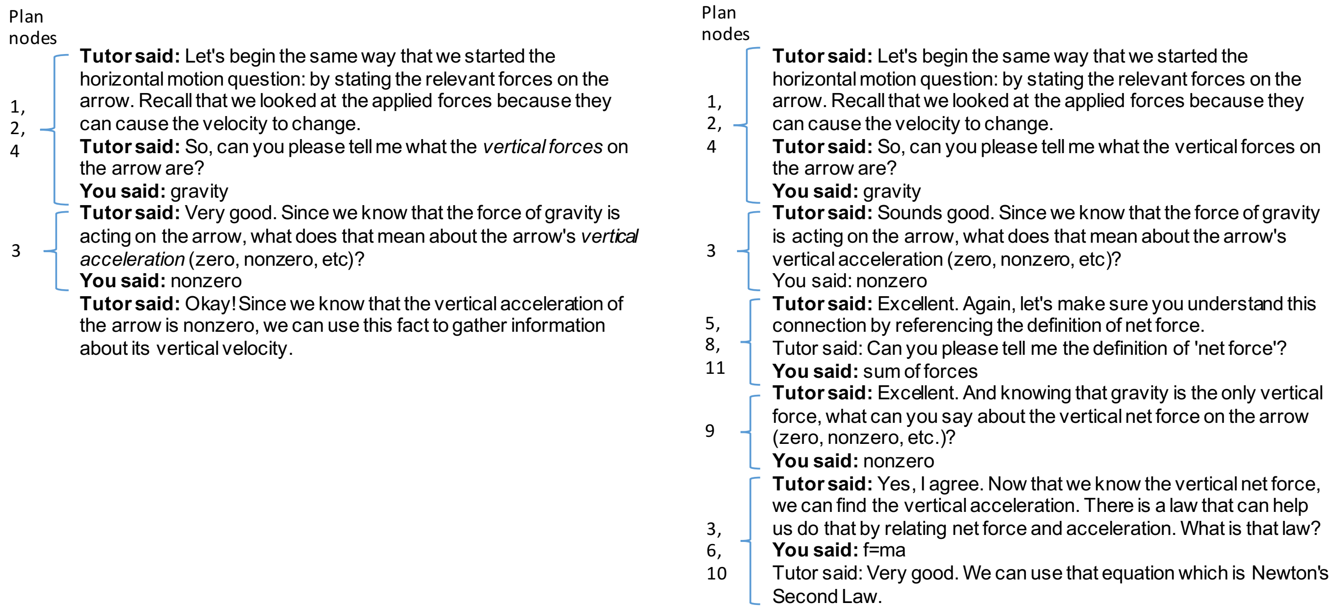
We consider a student with $T_D >= .8$ as very knowledgeable about the content that could be skipped, $T_D >= .5$ as having medium-level knowledge, and $T_D < .5$ as having low-level knowledge.

The algorithm applied at each decision point D is defined as:

```
if RQ node & node correct & $T_D >= .8$ then
    do not expand the RQ node
else if RQ node & (node incorrect OR $T_D < .8$) then
    expand the RQ node
else if node correct & $T_D >= .5$ then
    do not expand the node
else if node incorrect OR $T_D < .5$ then
    expand the node
end if
```

To simplify implementation for this experiment, the child nodes of an RQ node have identical KC_D and are a subset of KC_D for the RQ node. In future work, we will adjust each node's KC_D to more specifically reflect the content of its child nodes.

4. METHODS


Plan nodes
1, 2, 4
3

Tutor said: Let's begin the same way that we started the horizontal motion question: by stating the relevant forces on the arrow. Recall that we looked at the applied forces because they can cause the velocity to change.

Tutor said: So, can you please tell me what the *vertical forces* on the arrow are?

You said: gravity

Tutor said: Very good. Since we know that the force of gravity is acting on the arrow, what does that mean about the arrow's *vertical acceleration* (zero, nonzero, etc)?

You said: nonzero

Tutor said: Okay! Since we know that the vertical acceleration of the arrow is nonzero, we can use this fact to gather information about its vertical velocity.

Tutor said: Let's begin the same way that we started the horizontal motion question: by stating the relevant forces on the arrow. Recall that we looked at the applied forces because they can cause the velocity to change.

Tutor said: So, can you please tell me what the vertical forces on the arrow are?

You said: gravity

Tutor said: Sounds good. Since we know that the force of gravity is acting on the arrow, what does that mean about the arrow's vertical acceleration (zero, nonzero, etc)?

You said: nonzero

Tutor said: Excellent. Again, let's make sure you understand this connection by referencing the definition of net force.

Tutor said: Can you please tell me the definition of 'net force'?

You said: sum of forces

Tutor said: Excellent. And knowing that gravity is the only vertical force, what can you say about the vertical net force on the arrow (zero, nonzero, etc.)?

You said: nonzero

Tutor said: Yes, I agree. Now that we know the vertical net force, we can find the vertical acceleration. There is a law that can help us do that by relating net force and acceleration. What is that law?

You said: f=ma

Tutor said: Very good. We can use that equation which is Newton's Second Law.

Figure 4: The adaptive dialogue on the left represents the non-expanded network in Figure 3 and the adaptive dialogue on the right represents the fully expanded network (as in the control version of the system).

We are testing two versions of the system: one that always decomposes a target parent node into simpler child nodes regardless of the student's knowledge of the content to be discussed and a second one that decomposes target parent nodes into simpler nodes or not, based on the student's pre-test performance on items that target the knowledge needed to answer the RQ correctly. The second (adaptive) version of the system follows the algorithm described in the previous section.

4.1 Participants

The initial comparison of the two versions of Rimac was conducted within high school physics classes at one school in the Pittsburgh PA area. The study followed the course unit on dynamics with a total of 44 students participating. Students were randomly assigned to one of the two conditions: the non-adaptive control condition (N= 22), and the adaptive experimental condition (N=22). We are currently collecting data from additional high school physics classes in four other schools in the Pittsburgh PA area.

4.2 Materials

Students interacted with one of the two versions of Rimac to discuss the physics conceptual knowledge associated with two quantitative dynamics problems. These problems and their associated reflective dialogues (two to three dialogues per problem) were developed in consultation with high school physics teachers.

An online, automatically scored 19 item, multiple-choice pre-test and isomorphic post-test (that is, each question was equivalent to a pre-test question, but with a different cover story) was used to measure learning differences in students' conceptual understanding of physics from interactions with the system. Each test item was assigned a grade between 0 and 1 and scores for each item were totaled so that the maximum score possible was 19.

4.3 Procedure

On the first day, the teacher gave the on-line pre-test in class and assigned the two dynamics problems. During the next one to two class days (approximately 90 minutes total) and as homework, for each assigned problem students solved the problem on paper and then watched a video of a sample, worked-out solution in one of the two versions of Rimac and engaged in two to three "reflective dialogues" after each problem-solving video. The videos demonstrated how to solve the problem only (as shown in Figure 2, which displays the end of video snapshot on the left) and did not offer any conceptual explanations. Hence we do not believe that the videos contributed to learning gains. Finally, at the next class meeting, the teacher gave the on-line post-test.

5. INITIAL RESULTS

We analyzed the data to determine whether students who interacted with the tutoring system learned, as measured by differences from pre-test to post-test, regardless of their treatment condition (i.e., which version of Rimac they were assigned to use), whether there was a difference in learning gains between conditions and whether there was a difference in time on task between conditions to complete both problems and their associated reflection questions and dialogues.

5.1 Learning Performance

When comparing differences from pre to post-test using a paired samples t-test, for all students combined post-test scores were significantly higher than pre-test scores ($t(43) = 6.305$, $p < 0.001$, $d = .805$) and post-test scores were significantly higher than pre-test scores for students in both the experimental condition ($t(21) = 5.881$, $p < .001$, $d = 1.017$), which adaptively decomposes the highest node in the plan network or not (depending on students' pre-test scores) and selected sub-nodes, and the control condition ($t(21) = 3.385$, $p = .003$, $d = .6451$), which always decomposes those nodes that can be decomposed (i.e., all but the leaf nodes) in the plan network. These results suggest that students in the two conditions learned from both versions of the system.

When comparing the performance of the students who used the control version of the system to the students who used the experimental version of the system, using an independent samples t-test, there were no significant differences in the pre to post-test gain ($t(42) = .995$, $p = .325$, $d = .300$) nor in the normalized gain ($t(42) = 1.226$, $p = .113$, $d = 1.124$). Thus, as we hypothesized, the adaptive version of the system was not detrimental to students' learning, which suggests that the adaptive version of the system may have been decomposing just the target nodes that students needed to have decomposed.

5.2 Efficiency of Learning

When comparing the time on task of students who used the control version of the system to students who used the experimental version of the system, using an independent samples t-test, there were significant differences in the time on task to complete both problems ($t(23) = 1.879$, $p = .037$, $d = .567$). The mean time on task for the experimental condition was 2653.9 seconds (about 44 minutes) and for the control condition was 6801.5 seconds (about 1 hour and 53 minutes). The average difference in time spent between conditions was about 1 hour and 9 minutes. Thus students in the experimental condition spent significantly less time yet learned similar amounts to students in the control condition in which all target nodes were decomposed. This suggests that the version of the system used in the experimental condition may have accurately decided to decompose the target nodes that individual students needed to have decomposed.

5.3 Additional Measures

We also explored the frequency with which higher-level target nodes were actually decomposed by examining T_D values for all students in the experimental condition for the second problem. All but 2 of the 22 students needed at least 1 target node decomposed. The average number of decompositions of target nodes was 5.14 with a minimum of 0 and a maximum of 10. Given that most students needed some target nodes decomposed, this further suggests that the decision algorithm in the experimental version of the system may have been accurate in its decisions about when to decompose target nodes.

In future work, we also need to explore the degree to which the algorithm may be deciding unnecessarily that target nodes need to be decomposed (e.g., the thresholds need to be adjusted or the pre-test is not a good measure for determining when a node needs to be decomposed). Missing a necessary decomposition is likely to be detrimental to student learning. We can test for this possibility in future work by creating another control version of the system that never decomposes a target node when a student is able to answer it correctly. If the algorithm is unnecessarily decomposing nodes infrequently, we hypothesize that students who use the experimental version of the system will learn significantly more than students who use the new control version.

6. PRELIMINARY CONCLUSIONS AND FUTURE WORK

We are exploring the effectiveness of a simple algorithm that decides whether or not to decompose a step in a line of reasoning during tutorial dialogue. We developed two versions of the Rimac system to test its effectiveness: one control version that always decomposes a step regardless of the student's knowledge level on the content involved and one experimental version that decides whether or not to decompose a step based on the student's knowledge of the content involved in the step.

We found that students who used the experimental (adaptive) version of the system, which incorporates the simple decision algorithm, learned similarly to those students who used the control (non-adaptive) version of the system, but that the students who used the experimental version of the system were able to complete the same number of problems in less than half the time that it took students who used the control system. This suggests that the algorithm was effective in deciding when a step should be decomposed.

In future work we will continue to analyze the number of node decompositions that occur for students who use the adaptive system and we will test a version of the system in which there are never any decompositions of target nodes that are answered correctly to further test the validity of our decision algorithm. We will also explore additional adaptations that traverse the plan network in different ways. After we have fine-tuned and validated our decision algorithm, we will explore whether the algorithm will transfer to other tutorial dialogue domains.

7. ACKNOWLEDGMENTS

We thank Dennis Lusetich, Svetlana Romanova, and Scott Silliman. This research was supported by the Institute of Education Sciences, U.S. Department of Education, through Grant R305A130441 to the University of Pittsburgh. The opinions expressed are those of the authors and do not necessarily represent the views of the Institute or the U.S. Department of Education.

8. REFERENCES

[1] P. Albacete, P. W. Jordan, and S. Katz. Is a dialogue-based tutoring system that emulates helpful co-constructed relations during human tutoring effective? In *17th International Conference on Artificial Intelligence in Education*, 2015.

[2] M. Chi, K. VanLehn, D. Litman, and P. Jordan. Empirically evaluating the application of reinforcement learning to the induction of effective and adaptive pedagogical strategies. *User Modeling and User-Adapted Interaction*, 21(1-2):137–180, 2011.

[3] M. Evens and J. Michael. *One-on-one tutoring by humans and computers.* Psychology Press, 2006.

[4] P. W. Jordan, B. Hall, M. Ringenberg, Y. Cui, and C. Rosé. Tools for authoring a dialogue agent that participates in learning studies. In *Proceedings of AIED 2007*, pages 43–50, 2007.

[5] S. Katz and P. Albacete. A tutoring system that simulates the highly interactive nature of human tutoring. *Journal of Educational Psychology*, 105(4):1126, 2013.

[6] A. Latham, K. Crockett, D. McLean, and B. Edmonds. Adaptive tutoring in an intelligent conversational agent system. In *Transactions on Computational Collective Intelligence VIII*, pages 148–167. Springer, 2012.

[7] D. Wood. Scaffolding, contingent tutoring, and computer-supported learning. *International Journal of Artificial Intelligence in Education*, 12(3):280–293, 2001.

[8] R. M. Young, S. Ware, B. Cassell, and J. Robertson. Plans and planning in narrative generation: a review of plan-based approaches to the generation of story, discourse and interactivity in narratives. *SDV. Sprache und Datenverarbeitung the Intenational Journal for Language Data Processing*, 37:41–64, 2014.