Providing Formative Assessment to Students Solving Multi-path Engineering Problems with Complex Arrangements of Interacting Parts: An Intelligent Tutor Approach

Paul S. Steif (corresponding)
steif@andrew.cmu.edu
412-268-3507

Luoting Fu
luoting.fu@alumni.cmu.edu
412-268-2509

L. Burak Kara
lkara@cmu.edu
412-268-2509

Department of Mechanical Engineering
5000 Forbes Av.
Pittsburgh, PA 15213

ACKNOWLEDGMENTS
We thank Jackie Yang, Jeremy Jiang, and Rebecca Piston for their assistance in development, testing and implementation, and Ken Koedinger for insights into the learning curve analyses.

FUNDING
This work was supported by the National Science Foundation under grant DUE-1043241.

ABSTRACT
Problems faced by engineering students involve multiple pathways to solution. Students rarely receive effective formative feedback on handwritten homework. This paper examines the potential for computer-based formative assessment of student solutions to multi-path engineering problems. In particular, a cognitive tutor approach is adopted and tested out on problems of truss analysis, studied in engineering statics. With a cognitive model for solving the class of problems, the tutor allows the student wide latitude in solution steps, while maintaining sufficient constraints for judging the solution and offering feedback. Proper selection of judging points prevents interference with productive student work, while avoiding accumulated errors.
To monitor student learning, efforts to apply distinct skills were extracted on the fly from student work. Using statistical methods developed for intelligent tutoring systems, metrics of the effectiveness of the feedback and areas for further improvements were gleaned from error rates in successive opportunities to apply distinct skills.

Keywords: cognitive tutor, feedback, statics, engineering, interactive problem solve, student learning and assessment

1. Introduction

Many courses in engineering majors involve significant time spent by students solving homework problems and corresponding efforts to grade these problems (Fernandez, Saviz, & Burmeister, 2006). Problems often revolve around assessing a given physical situation or system using concepts and physical principles, which leads to equations that can be solved and conclusions drawn from their solutions. Problems differ in their level of complexity, from those that involve a single concept or step, to problems that require students to coordinate and organize multiple concepts and steps. A student may need to decompose the original problem into inter-related sub-problems, define variables of different types, carry out analyses of sub-problems, and finally combine and interpret the results. Often such problems have multiple pathways to the correct answers.

Clearly, an improved ability to solve problems is the desired outcome from all this effort. Such improvement should depend on practices that are known to promote learning generally, in particular, timely formative feedback to the learner (Anderson, Conrad, and Corbett, 1989; Bangert-Drowns, Kulik, Kulik, and Morgan, 1991; Corbett and Anderson, 2001; Hattie and Timperley, 2007). We take formative feedback, as defined by Shute (2008), as information communicated to the learner that modifies thinking or behavior to improve learning. In particular, this paper addresses the issue of providing effective and timely formative feedback for students confronting problems that have spatially complex arrangements of interacting parts, and in which there is significant latitude in decomposition and construction of solutions. Traditionally, students solve such problems as part of written homework assignments that are hand graded. It is certainly difficult for grading of written homework to provide *timely* feedback. Grading in many circumstances may take a week; students have likely engaged in, and probably completed, the following homework before receiving feedback on the prior homework.

Offering *effective* formative assessment of written homework is also exceptionally challenging. A correct final answer may confirm student work, but an incorrect answer likely provides no information on where a solution was in error. Thus, a grader would ideally recognize the different parts of the solution and seek to judge each part on its own. This is laborious to do: one part can utilize completed work that was incorrect and thus may vary from one student to the next. Given the very limited effectiveness of human grading to provide timely, effective feedback to students on multi-path homework problems, it is natural to inquire whether alternative means, for example by computer, can do better. In this paper we present an approach to providing automated, formative assessment of students' efforts to solve multi-path engineering problems, along with metrics that allow one to judge the effectiveness of the feedback and seek improvements to the formative assessment offered.

To provide automated, formative assessment for multi-path problems, the assessment system should grant the student latitude to follow any of the potential solution pathways, and still be able to judge student work and offer feedback regardless of the path taken. Furthermore, the freedom granted to students should permit them to commit errors commonly found in student work. Cognitive tutors, which have been developed for computer programming (Anderson, Boyle, and Reiser, 1985), and high school mathematics (Koedinger, Anderson, Hadley, and Mark, 1997; Koedinger, 2002), and other fields, offer one approach to enabling the assessment

system to interpret a range of possible solutions. Cognitive tutors are based on a cognitive model for a learner encountering the chosen tasks; they provide feedback based on that model, and they can also yield data upon which to judge whether learning is occurring with ongoing practice. The approach taken here is inspired in part by aspects of cognitive tutors.

The present work also shares the goal of the Andes intelligent tutoring system (VanLehn, Lynch, Schulze, Shapiro, Shelby, Taylor, Treacy, Weinstein, and Wintersgill, 2005), from the closely related domain of physics. In contrast to many cognitive tutors, the Andes tutoring system does not seek to provide instruction in a whole subject; rather it focuses exclusively on helping student learn to solve problems that are typically assigned by instructors. Andes resulted in demonstrable improvements to students learning to solve problems by comparison with traditional paper and pencil. However, by comparison with what is realistically feasible for the vast array of engineering subjects, problem solving in physics has received significant attention from cognitive science. Further, significant resources were devoted to developing Andes, and there is a notable time investment for students to learn to use Andes, an investment that is recouped over the course of an entire semester.

The present work seeks to show that simpler forms of intelligent tutoring can be practically implemented to aid problem solving in domains typical of engineering, which involve complex spatial arrangements of interacting parts and multiple possible solution paths. Unlike Andes and many cognitive tutors, our approach does not involve a high level of knowledge engineering, a full student model, a set of production rules, the necessity of determining in advance a complete set of solution pathways, and so forth. However, the tutor described below shares several features of Andes which its developers believed were most critical to its efficacy: insisting users be clear and explicit in defining variables, guiding students so correct solutions were arrived at upon completion, and the offering of hints that encouraged principle-based repair of errors.

We illustrate our approach with a tutor to help students learning to solve truss problems, which are commonly studied in statics, a course taken in multiple engineering majors. Trusses have complex arrangements of connected members (bars) that interact with each other, and with the external world. There are numerous pathways to solving such problems, with several types of steps taken in various orders sequentially and in parallel. The student selects portions of the truss including multiple whole and partial members, draws a free body diagram and writes down equations representing relevant physical laws for each selected portion, organizes the solving of equations, and interprets results physically in terms of the original truss. Mastery of trusses requires conceptual and mathematical competence, as well as clarity and systematic organization. Recently, computer systems have been developed (Roselli, Howard, and Brophy, 2006; Dannenhoffer and Dannenhoffer, 2009) that allow students to work on some simple statics problem more or less from start to finish, and provide feedback on individual steps. But, such systems do not involve problems with many solution paths, nor do they offer data upon which to judge how much students are learning. Trusses are potentially also a rich domain for studying other learning phenomena, for example evaluating the impact of different types of dialogs on learning and problem solving (Hausmann, 2005), typically by learners new to a domain. By contrast, we focus, like Andes, on helping students learn to derive mathematical relations between key quantities based on direct applications of physical laws. However, unlike physics problems treated in Andes, it is impractical, and we show unnecessary, to identify all solution

paths in advance in devising a tutor.

2. Design Of Tutors for Multi-Path Problems

There is no unique embodiment of a computer tutor for monitoring student solving of truss problems, let alone for solving multi-path engineering problems in general. However, results of prior research can provide guidance for design choices, particularly choices pertaining to how much to constrain user action and when to judge it. To provide some context for the discussion, we show a typical truss problem (Figure 1a) and a small part of the solution involving analysis of one selected portion of the truss (Figure 1b). Trusses have bars, pins connecting them, different means of anchoring the truss to ground (the "supports", symbolized with triangles), and applied forces. The goal is to find the resulting internal forces in the bars. Typically, the solver must consider multiple portions (subsystems) of the truss, draw a free body diagram for each (the set of forces acting on the chosen portion), and then write down equations of equilibrium. The solutions of those equations will affect the free body diagrams of other subsystems and ultimately lead to the results requested in the problem statement. The natural latitude in solving such problems is that the student can choose any portion of the truss, write equations in any order, then choose any other portion, and so forth, thus creating a large space of possible solution paths. Students in statics are typically taught two distinct methods for solving truss problems. In the method of joints (MoJ), depicted in the solution in Figure 1b, students choose subsystems of the truss that include a single pin and the connected partial bars. In the method of sections (MoS), students choose subsystems of the truss that include multiple adjacent pins, the connected bars, and the adjacent partial bars. The tutor presented below enables students to practice both methods of solving.

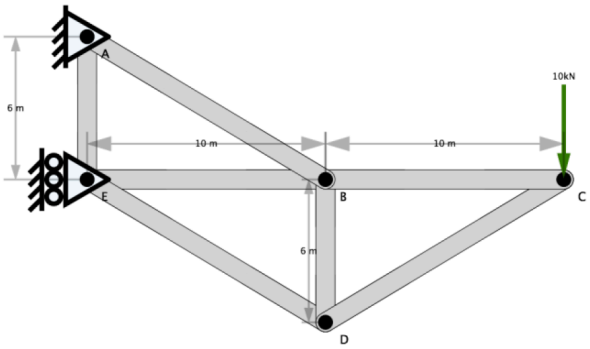

Figure 1a. Typical truss problem.

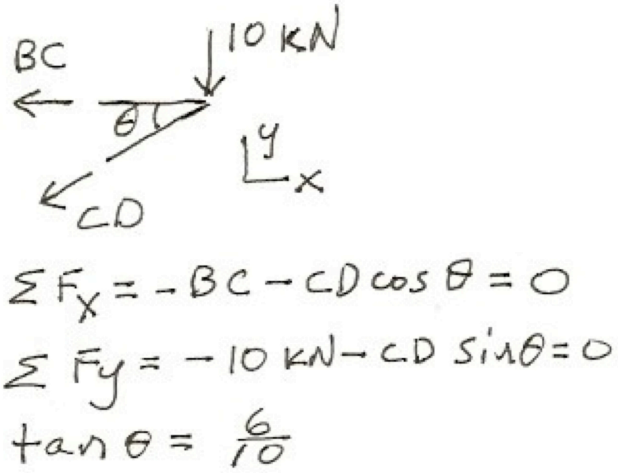

Figure 1b. Portion of handwritten solution to problem from Figure 1a in which joint C is analyzed.

Because learning to solve problems correctly in a free form style, akin to paper and pencil, is the goal, we seek to have user interactions with the tutor similar to those in the targeted task, provided the tutor maintains the ability to judge user work. While completely free form work such as writing with a stylus on a tablet might be ideal, challenges remain to implementing such technologies, although progress continues to be made along this front (Kara and Stahovich, 2004; LaViola, 2007; Peschel and Hammond, 2008; Lee, de Silva, Peterson, Calfee, and Stahovich, 2008; Fu and Kara, 2011). Within the constraints of conventional keyboard and mouse interactions, one can still grant the user opportunities to commit errors similar to those observed when students solve with pencil and paper. The tutor then has opportunities to detect and give feedback to students when they commit such errors.

There are, however, reasonable exceptions to the goal of making user interactions with the tutor as similar to paper-and-pencil solving as possible. Most problem solving involves some tasks that require mental resources, but which are mastered already by students at the level in question. It is worthwhile to identify tedious, non-essential tasks, which unnecessarily add to the cognitive load (Sweller, 1988, 1994) on the learner, and seek to off-load such tasks to the tutor. A key example in the tutor presented below will be removing the need to use an electronic calculator to obtain numerical solutions.

A second exception pertains to a key finding that students who explained example problems to themselves learn more from those examples (Chi, Bassok, Lewis, Reimann, and Glaser, 1989); researchers have termed this the self-explanation effect. This effect has had many implications; in particular it has been applied in some cognitive tutors (Aleven and Koedinger, 2002), where students need to explain their answers, and the tutor potentially evaluates those explanations. Such features in tutors seek to make student thinking more visible. Tutors of multi-path engineering problems can likewise create opportunities to make student thinking visible, to both the student and the tutor, thinking which is rarely visible in pencil and paper solving. Furthermore, the tutor can judge such explanations, and the additional information may better enable the tutor to interpret student work. A key example in the tutor presented below will be requesting the user to designate each defined force as falling into one of several categories.

To the extent that the tutor gives feedback prior to completion of a solution, the tutoring environment is clearly different from paper-and-pencil problem solving. The points in the solution process at which the tutor potentially intervenes and offers feedback clearly constitute significant decisions in the tutor design. Often, it is true that immediate feedback is best (Hattie and Timperley, 2007); this has the benefit of ensuring that the student associates the feedback with the action just taken. In exceptional circumstances, delayed feedback may be justified if it is feasible for students to check their work downstream and if such a skill is deemed worthwhile to develop (Mathan and Koedinger, 2005).

The tutor described below gives immediate feedback with the following caveat. Tutors for solving multi-path problems with limited constraints are distinct from most existing tutors: there is not a pre-determined set of answers which users are expected to supply or set of choices from which to select. The user is gradually adding elements of the solution on what is, in effect, an an initially blank canvas. In contrast to the answer entered into a box, parts of the solution just added to the canvas, such as a force added to a free body diagram, may be tentative. It would be annoying and counterproductive to critique user work that is still tentative. On the other hand, if errors accumulate too long and new work builds upon errors, judging new work becomes ambiguous. Based on observations of written homework, the possibility that students would check work downstream and then discover earlier mistakes was viewed as unlikely, and not worth the significant additional burden on interpretation.

3. Description of Tutor

We assume that students using the tutor have learned about truss analysis through other means, such as lecture and textbook. Thus, the tutor can focus exclusively on helping students solve problems, allowing a solution process such as depicted in Figure 1a to be conducted on the computer with as little constraint as possible, within the confines of a mouse and keyboard user interface, while maintaining the ability to interpret student work. Observations of student work and their typical errors, examples of which are shown in Steif, Fu, and Kara (2013), have guided tutor design. As stated above, the goal is for a student using the tutor to be able to commit most, if not all, errors that are observed in pencil and paper solutions. If some errors are never or rarely observed in student pencil and paper work, then the tutor user interface need not go to unnecessary lengths, at the expense of programming complexity or interpretation uncertainty, to permit such errors. For example, it is virtually always clear which member or partial bar is being drawn on paper and pencil; thus, the interface need offer only limited options of selection, rather than allow ill-formed depictions of bars that are ambiguous to interpret.

To satisfy the above requirements, the tutor limits users to the following actions:

- Any set of pins, members and partial members can be chosen as a subsystem for further analysis.

- In the free body diagram of a subsystem, forces can be drawn only at pins or at the free ends of partial members. Forces are confined to lie along x-y directions or parallel or perpendicular to bars.

- For each subsystem, equations of force equilibrium along x- and y-axes, and equations of moment equilibrium about any joint, can be written.

Figure 2 contains a screen shot of the tutor, with a problem partially solved. The left half of the display contains a menu bar at the top and the problem diagram and statement. The problem diagram can be toggled to display the solution diagram, where results (support reactions and bar internal forces) that have been determined are registered by the student, as described below. The user chooses a subsystem for analysis by clicking on a set of pins, members and partial members, and then clicking on the draw (pencil) icon from the menu bar. The selected group of parts is added as another subsystem and would appear as one of the thumbnails to the right half of the display. Clicking on a thumbnail expands that subsystem, allowing the user to draw its free body diagram (FBD) and write its associated equilibrium equations.

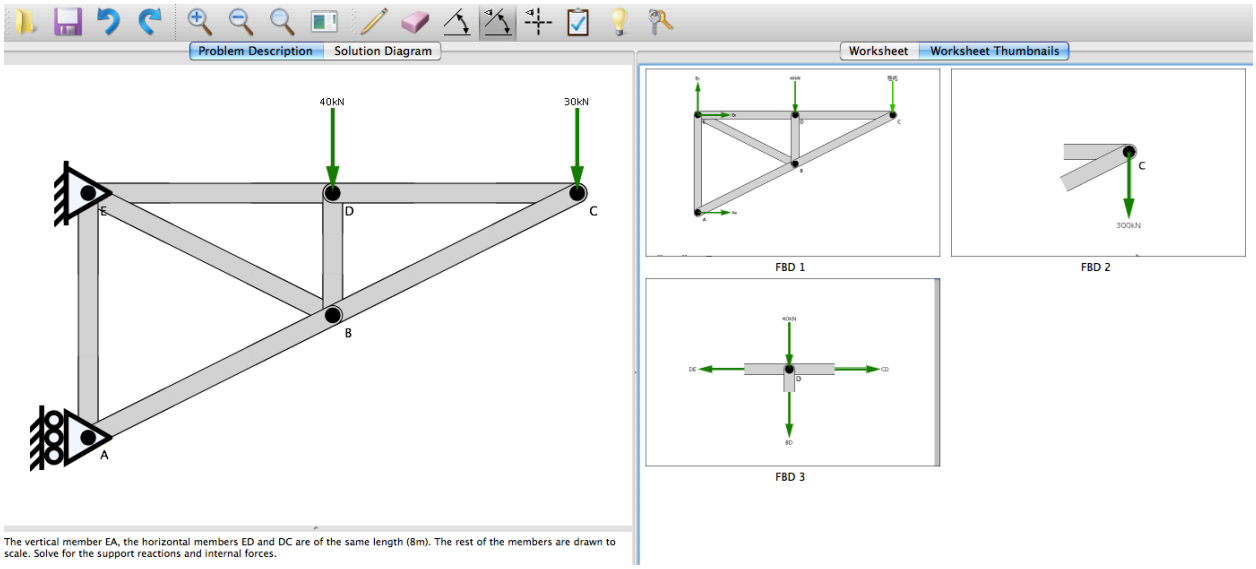

Figure 2. Screen shot of full display of truss tutor.

In Figure 3, we show a subsystem with a pin and the two connected partial members; a new force is being added to a partial member. As seen in the window labeled "Defining a force", the user categorizes each force being drawn. Sometimes more than one category is acceptable, but the category chosen affects the subsequent representation of the force. A category is not specified as part of pencil and paper solving, but it has been included in the tutor as a form of self-explanation. Requiring force categorization makes the user's thinking visible and aids the tutor in interpreting student work.

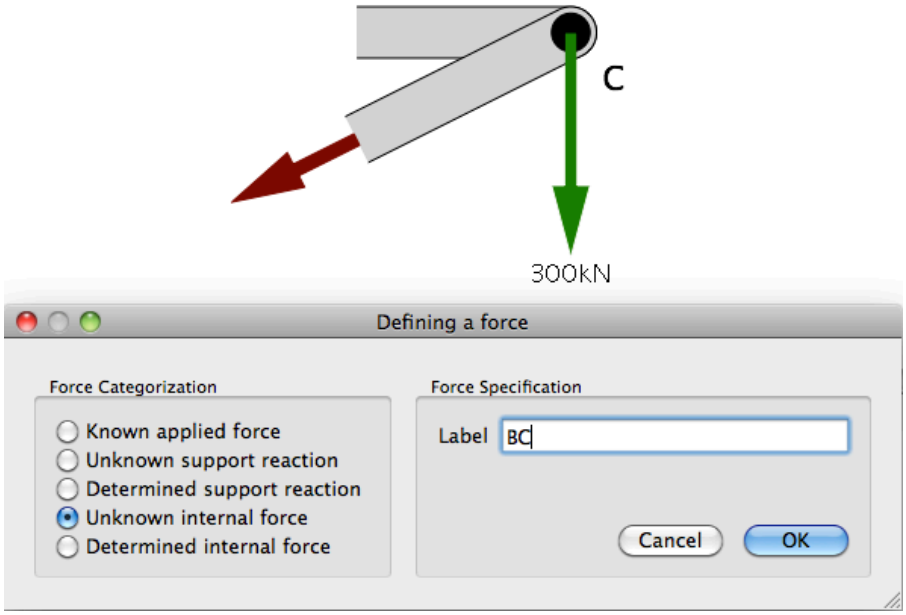

Figure 3. Screen shot of force being added to free body diagram, showing force categorization.

Beneath the free body diagram the user can write equilibrium equations for the subsystem (Figure 4). When the user has written down an equation with one variable (always a linear equation in truss analysis), upon request the tutor can solve the equation for that variable. This eliminates the need to use a calculator and also eliminates errors due to mistyping into a calculator. Once a variable such as a support reaction or an internal force has been determined, the user needs to "register" that force in the solution diagram. Registration serves to declare that a force has been determined, so it can be categorized as a determined force in a subsequent FBD. Registration is also an important opportunity for the student to signal the meaning of what has been solved. Unknown support forces can be drawn on FBD's in any direction; the associated variables may turn out to be positive or negative. But in the solution diagram the support force must be drawn in its actual sense and given a positive magnitude. Likewise, when the internal force of a bar is registered, the user gives it a magnitude and describes it as in tension or compression. More details on the interface have been presented by Steif, Fu, and Kara (2013).

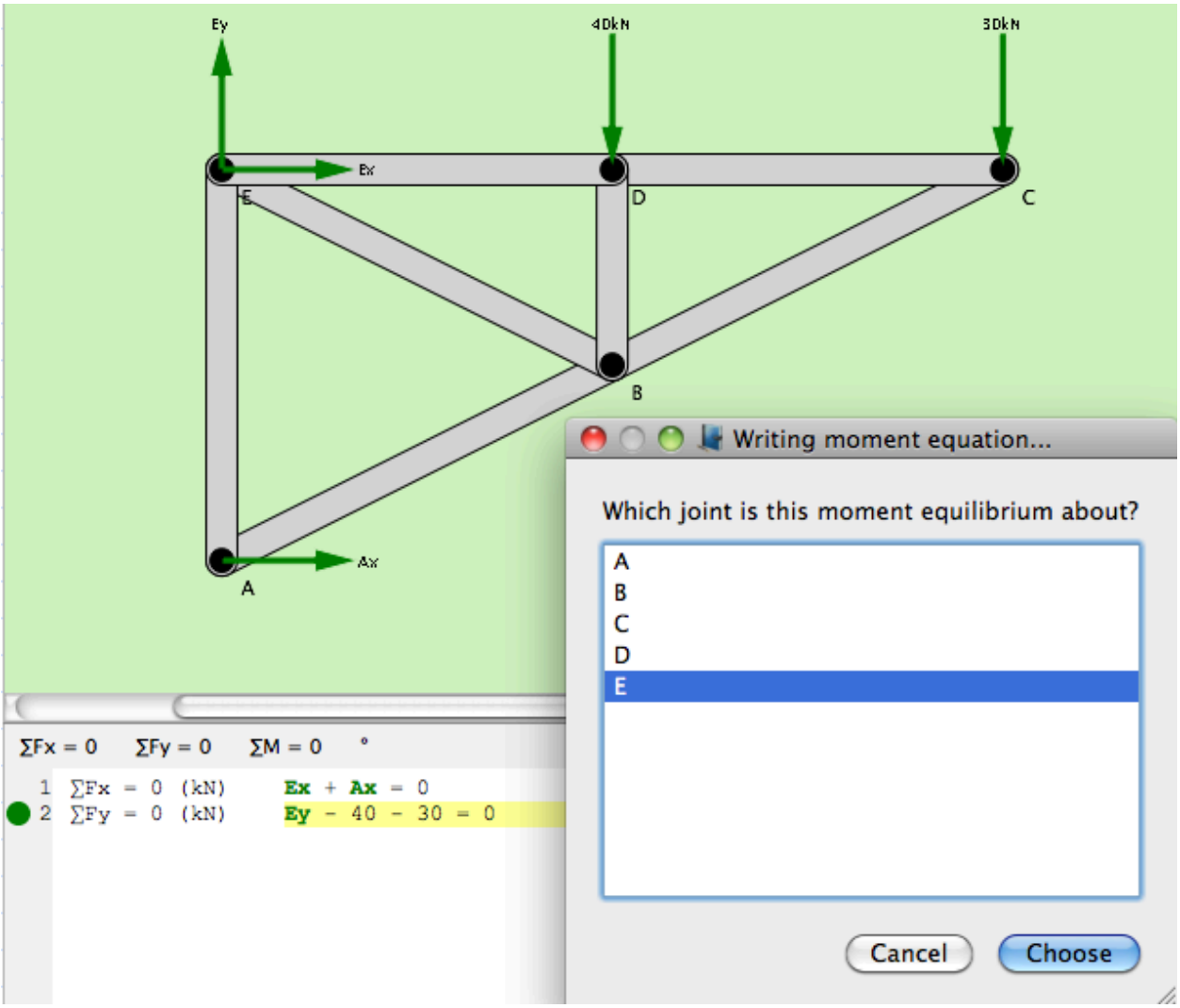

Figure 4. Screen shot of writing equations, and choosing moment center.

4. Judging student work and giving feedback

A key capability of the tutor is to judge work and give feedback on it. The tutor can do this by having algorithms for carrying out the steps for solving truss problems. These algorithms are analogous to a cognitive model of the domain. There are distinct algorithms corresponding to the distinct stages in the solution for a given subsystem:

- SUBSYSTEM: An algorithm to determine if a group of pins, members, and partial members constitutes a valid subsystem.
- FREE BODY DIAGRAM (FBD): Given a valid subsystem, and any forces defined or determined up to that point, an algorithm for the allowable forces that can be drawn on the pins and partial bars of the subsystem. The FBD of a given subsystem is not unique. For example, if an internal force has been determined, the algorithm allows that force in a

new FBD to be represented either as a determined force using the correct value, or as an unknown internal force using symbols consistent with the first definition.

- EQUILIBRIUM EQUATIONS: Given a valid FBD, an algorithm for the correct set of algebraic terms in the summations of forces along x- and y-axes and the summation of moments about any pin in the truss. These summations include variables and constants and must be consistent with how forces appear in the FBD. The terms can be in any order and there are multiple ways of composing terms.
- SOLUTION REGISTRATION: Given a correctly determined support or internal force (from the equilibrium equations), an algorithm for the correct registration of that force in the solution diagram.

As described above, the tutor seeks to offer immediate feedback. But, it is recognized that a student may be in the midst of formulating the current portion of the solution, such as drawing the forces on a FBD or writing a single equation of equilibrium, when interruptions would be annoying. On the other hand, we do not want to wait so long that the student builds upon work that is as yet unjudged and may be incorrect. In the latter, undesirable situation, the tutor might need to indicate that the built-on portion is correct in and of itself, but that it is based on incorrect prior solution steps.

We can offer immediate feedback, while respecting the tentative nature of currently formulated portions of solution, by judging and offering feedback at these points: (i) the subsystem is judged after the user has selected parts and clicked on the draw subsystem button; (ii) the FBD of a subsystem is judged after the user clicks to initiate the writing of the first equilibrium equation for that subsystem; (iii) an equation is judged after the user types return while entering an equation or clicks to initiate the writing of a new equation; and (iv) the registered result is judged after the user has entered a result into the solution diagram and clicked "Ok". In each case, if there is an error, the student receives feedback that points out the error, including information to enable the user to fix the error and to learn why it is an error, thus lessening the likelihood of repetition. This type of feedback, in light of results presented below, is formative feedback by the definition of Shute (2008). Moreover, until the errors are corrected, the user cannot go on to the next stage of solution for the subsystem that has an error. Thus, it is unnecessary for the tutor to have algorithms to judge solution paths that build upon earlier committed errors. The student can pursue many different solution paths, but is halted on a particular path until detected errors are corrected.

With the approach just described, it is possible to provide automated, formative assessment of students' efforts to solve one class of engineering problems with complex arrangements of interacting parts that have multiple solution pathways. To generalize our approach to such problems more generally, a tutor should have three integrated elements. First, it must have a graphical user interface that allows interactions that enable users to pursue solutions and commit errors elsewhere observed in student work. Second, it must have algorithms that can judge the correctness of actions that the interface permits. Third, it must have suitable junctures at which to judge student work and, if need be, halt further progress until algorithms can once again accurately judge student work. The complexity of problems to be handled, and the latitude granted to students while solving them, is a matter of tutor design. The approach is streamlined compared to other intelligent tutors, because it demands only algorithms that judge the

correctness of forward steps, presuming the current state is a correct state. In the remaining sections, we propose an approach to determine if the feedback promotes learning and on how potential changes in the tutor can be targeted to improve learning.

5. Analysis of student work to track learning

As one approach to judging whether the tutor promotes learning, we seek to determine whether types of errors that are initially prevalent are observed less frequently as students progressively solve more problems. Because different sub-tasks may have distinct difficulties, we need to keep track of how students fare with respect to different subtasks. How we choose to view the problem as composed of subtasks is central to developing evidence as to whether learning is occurring. These choices constitute our model of learning to solve problems in the chosen subject or topic: they are the distinct skills or *Knowledge Components* that the student needs to learn. The task analysis underlying the tutor for truss problems has been informed by previously identified concepts and skills in statics (Steif, 2004) and the development of the statics concept inventory (Steif and Dantzler, 2005). The Knowledge Component (KC) model used thus far involves 23 skills each falling into one of the phases of the solution process: selecting a subsystem, drawing a free body diagram, writing equations of equilibrium, and registering a result derived from an equation of equilibrium in the solution diagram. The full set of knowledge components (KC model) for the analysis reported here is given in Table 1.

Table 1. Full set of Knowledge Components (KC Model) used to analyze presented data.

KC Category: Select Subsystem	
KC1	Select full truss (all bars and pins) as subsystem
KC2	Select joint (pin and attached partial bars) as subsystem
KC3	Select section (pins, bars, and partial bars) as subsystem
KC Category: Draw FBD	
KC4	Draw known applied force
KC5	Include no forces on pin that is supposed to be free
KC6	Represent unknown reaction forces at pin support for first time
KC7	Represent unknown reaction forces at roller support for first time
KC8	Represent unknown support reaction forces consistent with prior representation
KC9	Draw now known support reactions forces that were previously determined
KC10	Represent unknown internal force in bar for first time
KC11	Represent unknown internal force in bar consistent with prior representation
KC12	Draw now known internal forces in bar previously determined
KC Category: Write Equilibrium Equation	
KC13	Include in summation terms that contribute to known force (no resolution)
KC14	Include in summation terms that contribute to known moment (no resolution)
KC15	Include in summation terms that include vector resolution of a known force
KC16	Include in summation terms that include vector resolution of a known moment
KC17	Include in summation term that requires resolving variable force
KC18	Include in summation term that requires resolving variable moment
KC19	Replace variable in equation with value found from previously solved equation
KC20	Include in summation term with variable force (no resolution)
KC21	Include in summation term with variable moment (no resolution)
KC Category: Register Result	
KC22	Register value and draw determined support force in solution diagram
KC23	Register value and direction of determined internal force in solution diagram

The tutor described here is distinct from tutors in which there are known-in-advance sets of items that the student responds to, which can be tagged with the KCs in advance. In this tutor, the student is building up the solution on essentially a blank canvas. At the discrete junctures for judging work described above, the tutor records each new instance in which the user undertakes an action corresponding to one of the KCs, whether it is done correctly or not. Any correction by the user of an incorrect action in response to feedback is not counted as a new opportunity to exercise the KC. The student charts his or her own solution pathway, and the tutor extracts on the fly the sequence of KCs attempted, which can be different for each student.

To analyze the progression of learning, we have adopted the terminology, methodologies, and tools from the Pittsburgh Science of Learning Center Datashop (Koedinger, Baker, Cunningham, Skogsholm, Leber, and Stamper, 2011). Data corresponding to the sequence of KC opportunities

for each student in a sample are extracted from the files the student saves while using the tutor; these data are imported into Datashop. Among the various outputs from Datashop pertinent to our study is the so-called learning curve: a plot of the percentage of students in the sample that err in applying a particular KC as a function of the opportunity (first, second, third, etc.) to use that KC.

Learning curves are typically noisy; to determine if such data provide evidence of learning, Datashop tools also fit a statistical model to the sequence of opportunities to apply a KC. In particular, Datashop fits a widely used logistic regression model (Draney, Pirolli, and Wilson, 1995) for capturing the progressive mastery of a skill with practice. For our learning model in which each action is dependent on a single KC, the statistical model predicts error fraction as follows:

$$\ln[(1- e_{ij})/e_{ij}] = \theta_i + a_j + b_j T_j$$

In this equation, e_{ij} is the probability of an incorrect answer by the ith student on opportunity T_j for using the jth KC. Note that e_{ij} can range from 0 to 1, and T_j takes on values of 1, 2, 3, and so forth, for the first, second, and third opportunity. The parameter θ_i captures the overall initial skill level of the ith student. The parameter a_j, referred to as the intercept, reflects the initial probability of correctly applying the jth KC. The coefficient b_j, referred to as the slope, corresponds to the rate at which errors in using the jth KC decrease with successive opportunities to practice it. In this commonly used model, one takes the student parameter, θ_i, to be KC-independent, and KC-parameters, a_j and b_j, to be student-independent. Fitting this model to data for a student sample yields the parameters in the statistical model. Note, in particular, that values for b_j are one measure of the tutor's effectiveness: more effective error messages or hints should lead to higher slopes, that is more rapid decreases in errors with practice.

6. Samples

The tutor described here is appropriate for students in virtually all statics courses. Because the tutor is intended to substitute for completing paper and pencil homework, use of the tutor fits into the rhythm of statics courses generally. Thus, the target population for a tutor such as this corresponds to most students who might take a statics course.

Because we wanted to capture how a tutor can give feedback on multi-path problem solving in the context of real engineering courses, the study was conducted within the scope of regularly scheduled statics courses. The tutor was used in lieu of solving paper and pencil homework problems in two distinct educational environments. Data was collected for all students and information on their completion of problems was returned to the instructor for the purposes of assigning a grade on the homework assignment. When students first registered to receive the tutor software, they were asked if they consented to have their data used anonymously for research; only data from those who consented were included in the analysis presented below.

Sample 1 was from a statics course at a community college, in a class comprising a total of 21 students. Of those students, 18 consented to have their data studied. Sample 2 was from a statics

course at a military academy, in a class comprising a total of 109 students. Of those students, 99 consented to have their data studied. In both classes, students had received lecture on trusses, covering the method of joints and method of sections, and were shown the solution of example problems. Thereafter, students practiced solving trusses exclusively using the tutor (no paper and pencil problems). Students in sample 1 were assigned five problems using the method of joints and five problems using the method of sections; sample 2 students were assigned three problems using the method of joints and five problems using the method of sections. There is no claim that these two samples are broadly representative of students, nor is there any reason to expect them to be atypical.

7. Results and Analysis

Typical learning curves for specific knowledge components are shown in Figures 5 – 7, all pertaining to Sample 2. The data points and solid lines connecting them (red) are the actual error percentages. The dotted (blue) curve is the prediction based on the fit of the statistical model. Successive opportunities in a learning curve can correspond to diminishing numbers of students, because different students have different solution paths and may even solve fewer problems. These three learning curves represent three typical outcomes. For KC9 depicted in Figure 5, utilizing a determined support reaction in a subsequent FBD, the error is reasonably high initially and becomes progressively lower with practice. This suggests that practice is having the desired effect – getting feedback on the errors enables students to gradually make fewer errors. For KC23 depicted in Figure 6, registering an internal force, the error starts low and remains low. There is little need for tutoring on this skill. Finally, for KC15 depicted in Figure 7, which pertains to one facet of writing equations of equilibrium, the error rate is initially high and never improves. (The wild error rate at the end corresponds to a very small number of students making many errors.) Practice is having no observable benefit. Learning curves for knowledge components associated with writing equations of equilibrium were found to be rather erratic generally. By the point in the course when they study trusses, students already have experience writing equations of equilibrium. Tutor feedback on equation writing is not conceptually informative, but simply points to terms that are in error, which are then readily corrected. We, therefore, speculate that the incentive for being careful in writing equations in the tutor is very low, at least in comparison with other stages of solution. Hence, additional results for these knowledge components are not presented.

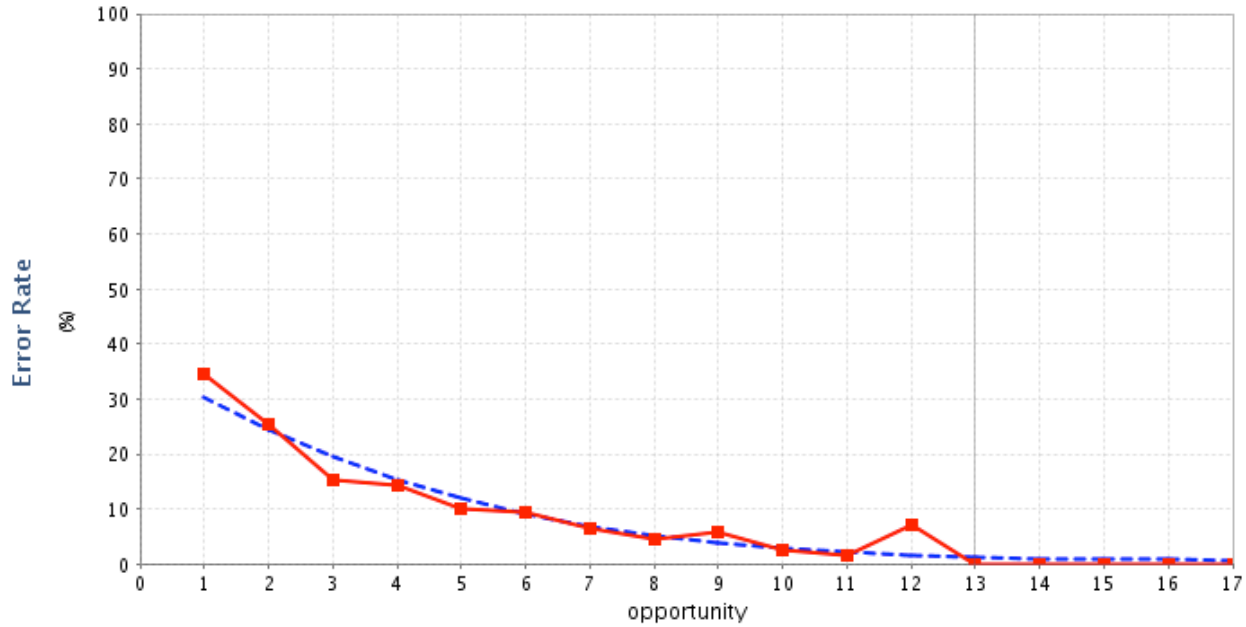

Figure 5. Percentage of students in error plotted as a function of opportunity (Learning curve) for KC9 (representing a determined support reaction) for which the error rate is initially high, but decreases with practice.

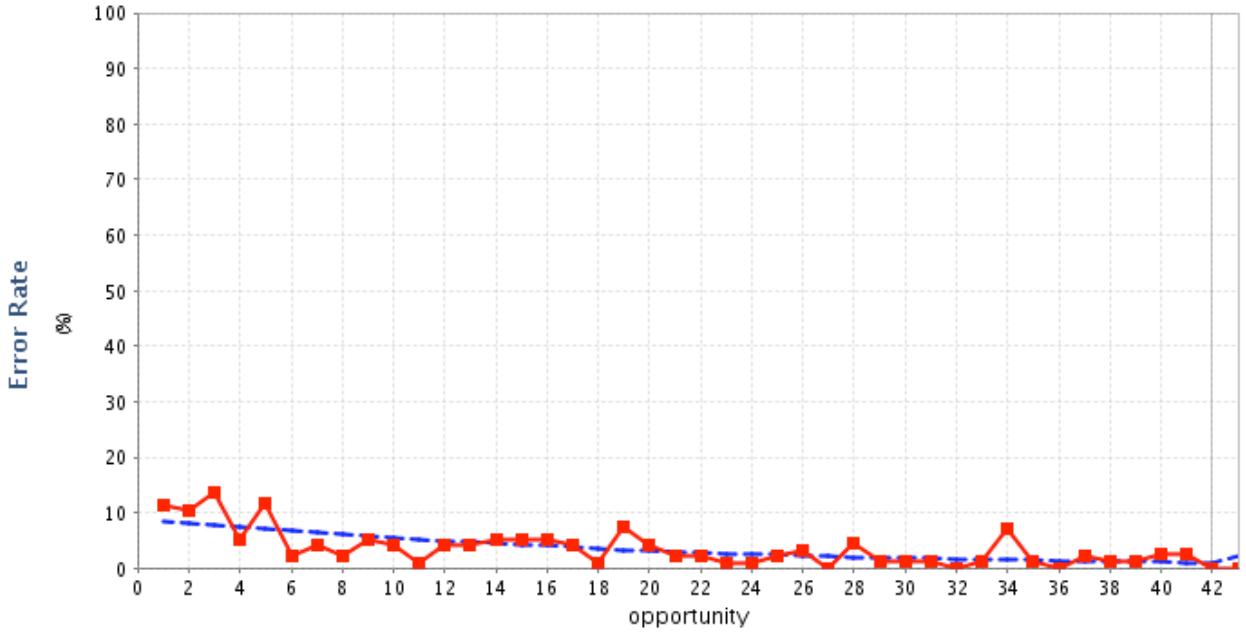

Figure 6. Percentage of students in error plotted as a function of opportunity (Learning curve) for KC23 (registering an internal force) for which the error rate starts low and remains low.

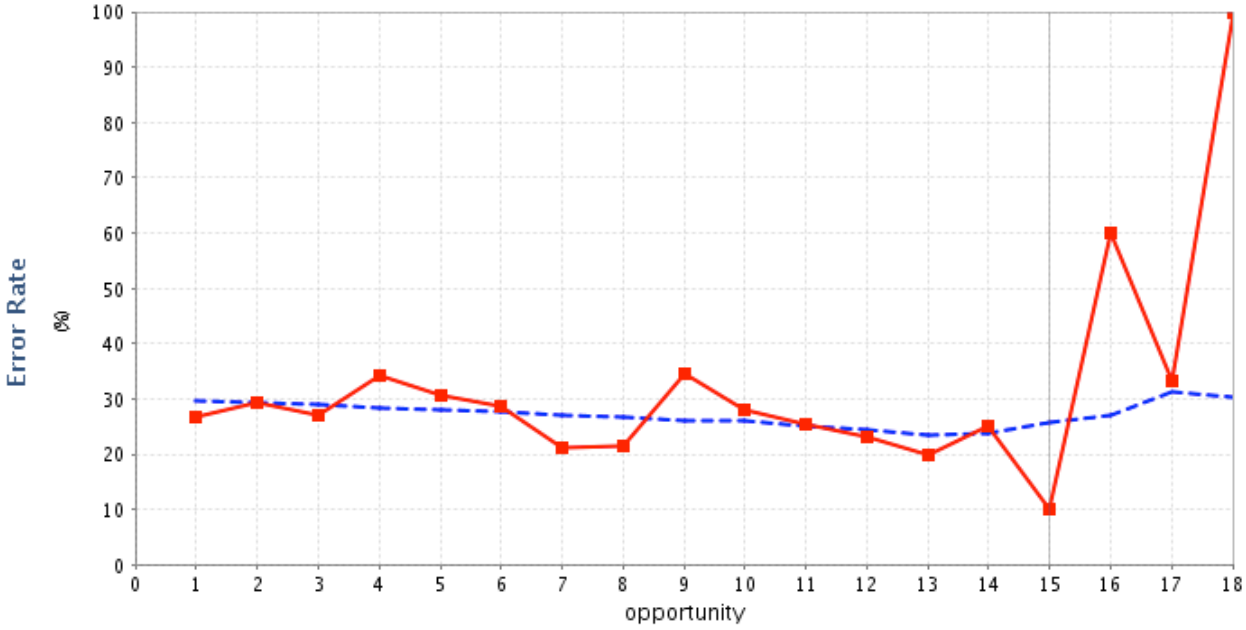

Figure 7. Percentage of students in error plotted as a function of opportunity (Learning curve) for KC15 pertaining to writing equilibrium equations for which the error rate is initially high and remains high.

The possible decrease in errors with practice, the learning rate, corresponds to the KC-specific slope b_j. Typical learning rates with existing tutors (Koedinger, McLaughlin, and Stamper, 2012) correspond to slopes b_j in the range of 0.05 to 0.15. To illustrate the rate of improvement that such slopes imply, let the probability of a student first making an error be 0.5. Then, with a slope of 0.1, the error probability drops to 0.40 at the fifth opportunity and to 0.29 at the tenth opportunity.

The values of the parameters a_j and b_j in the statistical model, when fit to data from each of the two samples, are shown in Table 2. The intercept corresponds to 1 – initial error fraction, as predicted by the model fit. So a skill for which students incur few errors initially would have a high intercept; many initial errors would correspond to low intercept. The KCs have been grouped according to the phases of solution: selecting subsystems, drawing FBDs, and registering results. Within each phase, the KCs have been ordered by increasing intercept (in sample 1). (There were too few opportunities to exercise KC8 to produce a meaningful learning curve.) The following observations hold for both samples. Some, but not all, of the KCs with lower intercepts have quite high slopes, for example, section_as_subsystem, unknown_internal_consistent, and determined_support. The tutor is playing a valuable role if it helps students master skills, such as these, that they did not initially possess. Thus, high slopes are most critical in the case of low intercept KCs. By contrast, other skills tend have a low initial error rate, which corresponds to high value of intercept. For a few of those skills, such as unknown_support_new_pin and unknown_support_new_roller, the slope is again high, but for other skills, the slope is low. In any event, rapid reduction in the error rate with practice (high slope) is less critical if the initial skill level is relatively high. Altogether learning curves and the model parameters of intercept and slope constitute metrics that can be used to gauge whether

learning is occurring while using the tutor. Furthermore, for the tutor described here, these metrics suggest that learning is occurring across a number of key component skills of solving truss problems. Further in-depth studies of metrics are beyond the scope of this paper.

Table 2. Statistical fit of Knowledge Component learning model for two samples: initial fraction correct (Intercept) and decrease of error fraction with practice (Slope) for different Knowledge Components.

	Sample 1		**Sample 2**	
KC (Select Subsystem)	**Intercept (a_j)**	**Slope (b_j)**	**Intercept (a_j)**	**Slope (b_j)**
KC3: section_as_subsystem	0.74	0.20	0.68	0.26
KC2: joint_as_subsystem	0.94	0.00	0.93	0.01
KC1: full_truss_as_subsystem	1.00	4.54	0.98	0.31
KC (Draw FBD)	**Intercept**	**Slope**	**Intercept**	**Slope**
KC11: unknown_internal_consistent	0.26	0.46	0.45	0.16
KC9: determined_support	0.51	0.32	0.64	0.30
KC12: determined_internal	0.79	0.06	0.67	0.10
KC6: unknown_support_new_pin	0.82	0.28	0.91	0.31
KC7: unknown_support_new_roller	0.89	0.12	0.87	0.38
KC10: unknown_new_internal	0.89	0.02	0.91	0.03
KC4: applied_force	0.90	0.04	0.81	0.10
KC5: free_pin	0.98	0.07	0.98	0.14
KC (Register Result)	**Intercept**	**Slope**	**Intercept**	**Slope**
KC13: register_support_force	0.87	0.06	0.91	0.03
KC14: register_internal_force	0.88	0.05	0.92	0.05

Note that results can also be used to decide where improvements to the tutor's feedback are warranted. One can seek out in Table 2 those skills with insufficiently high intercept and insufficiently high slope; that is, skills for which the error rate was initially high and did not decrease rapidly with practice. Most notable is the KC12 determined_internal: this corresponds to the skill of using a bar internal force, which has been already determined, in a new FBD where that internal force also acts. One must use the correct magnitude and interpret the earlier found tension or compression to draw the force in the correct direction in the new FBD. For both data sets, this KC does not have a high intercept (0.79 and 0.67) and does not have a particularly high slope (0.06 and 0.10), at least not high compared to the slopes for some of the other KCs. With the goal of accelerating the learning of this skill, the feedback on the associated error could be altered; whether such alterations lead to improvement can be judged based on the results for intercept and slope of this KC for new samples in which students receive the altered feedback. This will be considered in future research. We note that ultimate effectiveness of such a tutor could only be determined by a controlled study that compared the truss problem solving ability of students who used the tutor with those who had solved only with pencil and paper. Such a study is currently ongoing, with results to be reported in the future.

8. Summary and Conclusions

Problems that engineering students learn to solve often involve spatially complex arrangements of interacting parts and have multiple pathways to solution. It is difficult for human graders to provide effective formative feedback to handwritten solutions that are typically turned in as part of homework assignments. With the goal of devising better means of providing feedback for such problems, we have undertaken the development of a computer tutor that allows students to pursue multiple pathways to solution, and still provides feedback on those efforts.

We have taken an approach inspired by intelligent and cognitive tutors: basing the computer tutor not on preset correct answers, but on algorithms for judging the correctness of steps that students might take to solve the problems of interest. In particular, a computer tutor was devised for the test case of truss problems in statics; the interface permits the user to solve problems correctly following any pathway and to commit commonly observed errors. Immediate feedback is provided, short of annoying interruptions. Furthermore the tutor prevents new work from being built upon previously committed errors, which enables the judging algorithms to be limited to steps from a current correct state. Furthermore, the steps for solving truss problems are cast as a distinct set of skills or knowledge components (KCs). Each action by the student is viewed as an opportunity to exercise a KC, and the effectiveness of feedback can be judged based on whether fewer students incur errors with successive opportunities. The fit of a statistical model to the curves of percent error vs. opportunity for each KC yields values for the parameters in the model, which can serve as metrics for the effectiveness of feedback.

Data was obtained from students in statics classes at two institutions who used the tutor for one week's assignment in lieu of pencil and paper homework. We found that the error rates for various KCs differ significantly. From the fit of the statistical model, most of the KCs either had low error rates from the start, or if the error rate was initially high, it decreased markedly with successive opportunities to practice. Thus, for most skills, students already had the skill at the start or developed the skill through using the tutor. Furthermore, based on those few KCs for which the error rate decreases insufficiently, we have identified aspects of the tutor that could benefit from improvements. In general, this paper has shown that solving of problems with complex spatial arrangements of interacting parts and multiple pathways to solution is amenable to automated feedback with computer tutors. Further, the algorithms for judging distinct steps both can enable the tutor to follow many possible solution pathways, and provide metrics upon which to judge the effectiveness of feedback and pinpoint areas for tutor improvement.

Bibliography

Aleven, V.A.W.M.M. and Koedinger, K. R. (2002). An effective metacognitive strategy: learning by doing and explaining with a computer-based Cognitive Tutor, *Cognitive Science 26*, 147–179.

Anderson, J. R., Boyle, C. F., and Reiser, B. J. (1985). Intelligent tutoring systems. *Science 228*, 456-468.

Anderson, J. R., Conrad, F. G., and Corbett, A. T. (1989). Skill Acquisition and the LISP Tutor, *Cognitive Science 13*(4), 467-505.

Bangert-Drowns, R. L., Kulik, C.-L., Kulik, J. A., and Morgan, M. (1991). The instructional effect of feedback in test-like events. *Review of Educational Research 61*, 213-238.

Chi, M. T. H., Bassok, M., Lewis, M. W., Reimann, P., and Glaser, R. (1989). Self-explanations: how students study and use examples in learning to solve problems. *Cognitive Science 13*, 145–182.

Corbett, A.T. and Anderson, J.R. (2001). Locus of feedback control in computer-based tutoring: Impact on learning rate, achievement and attitudes. *Proceedings of ACM CHI'2001 Conference on Human Factors in Computing Systems*, 245-252.

Dannenhoffer, J. and Dannenhoffer, J. (2009). An online system to help students successfully solve statics problems. Proceedings of the 2009 American Society for Engineering Education Annual Conference and Exposition, Austin, Texas, June 2009.

Draney, K.L., Pirolli, P., and and Wilson, M. (1995). A measurement model for complex cognitive skill. In P. Nichols, S.F. Chipman, and R.L. Brennan (Eds.). *Cognitively diagnostic assessment* (pp. 103–126). Hillsdale: Erlbaum.

Fernandez, A., Saviz, C., and Burmeister, J. (2006). Homework as an outcome assessment: Relationships between homework and test performance. *Proceedings of the American Society for Engineering Education Annual Conference and Exposition*, Chicago, IL.

Fu, L. and Kara, L. B. (2011). From engineering diagrams to engineering models: Visual recognition and applications. *Computer-Aided Design 43*(3), 278-292.

Hattie, J., and Timperley, H. (2007). The power of feedback. *Review of Educational Research 77*(1), 81-112.

Hausmann, R.G.M. (2005). Elaborative and critical dialog: two potentially effective problem-solving and learning interactions. Doctoral Dissertation, University of Pittsburgh, http://d-scholarship.pitt.edu/9144/.

Kara, L. B., Stahovich, T. F. (2004). Hierarchical Parsing and Recognition of Hand-Sketched Diagrams. *17th ACM User Interface Software Technology* (UIST).

Koedinger, K. R. (2002). Toward evidence for instructional design principles: Examples from Cognitive Tutor Math 6. Presented at *Proceedings of PME-NA XXXIII, Annual Meeting of the North American Chapter of the International Group for the Psychology of Mathematics Education* 2002.

Koedinger, K. R., Anderson, J. R., Hadley, W. H., and Mark, M. A. (1997). Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education* 8, 30-43.

Koedinger, K.R., Baker, R.S.J.d., Cunningham, K., Skogsholm, A., Leber, B., Stamper, J. (2011). "A Data Repository for the EDM community: 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.

Koedinger, K., McLaughlin, E., Stamper, J. (2012). Automated Student Model Improvement," In *Proceedings of the 5th International Conference on Educational Data Mining (EDM 2012).* Chania, Greece. Jun 19-21, pp. 17-24.

LaViola, J. (2007). Advances in Mathematical Sketching: Moving Toward the Paradigm's Full Potential", *IEEE Computer Graphics and Applications 27*(1), 38-48.

Lee, W., de Silva, R., Peterson, E. J., Calfee, R. C., Stahovich, T. F. (2008) Newton's Pen: A pen-based tutoring system for statics. *Computers and Graphics 32*(5), 511-524.

Mathan, S. and Koedinger, K. R. (2005). Fostering the intelligent novice: Learning from errors with metacognitive tutoring. *Educational Psychologist 40*(4), 257-265.

Peschel, J., and Hammond, T. (2008) STRAT: A Sketched-Truss Recognition and Analysis Tool, *International Workshop on Visual Languages and Computing (VLC 2008)*, pp. 282-287, Boston, Massachusetts.

Roselli, RJ, Howard, L and Brophy, S. (2006). A computer-based free body diagram assistant. Computer Applications in Engineering Education 14: 281-290

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

Steif, P.S. (2004). An Articulation of the Concepts and Skills Which Underlie Engineering Statics. *34rd ASEE/IEEE Frontiers in Education Conference*, Savannah, GA.

Steif, P.S. and Dantzler, J. A. (2005). A Statics Concept Inventory: Development And Psychometric Analysis. *Journal of Engineering Education 94*, 363-371.

Steif, P.S., Fu, L., and Kara, L.B. (2013). Technical Report: Development of a Cognitive Tutor for Learning Truss Analysis, Retrieved 03/21/15 from http://www.andrew.cmu.edu/user/steif/papers/Truss%20tutor%20Technical%20Paper%202013.pdf

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science 12,* 257-285.

Sweller, J. (1994). Cognitive load theory, learning difficulty and instructional design. *Learning and Instruction 4*, 295-312.

VanLehn, K., Lynch, C., Schulze, K. Shapiro, J. A., Shelby, R., Taylor, L., Treacy, D., Weinstein, A., and Wintersgill, M. (2005). The Andes physics tutoring system: Lessons Learned. *International Journal of Artificial Intelligence and Education 15 (3)*, 1-47.