Computer Tutors Can Reduce Student Errors and Promote Solution Efficiency for Complex Engineering Problems

Paul S. Steif, Matthew Eicholtz, Levent Burak Kara
Department of Mechanical Engineering
Carnegie Mellon University
Pittsburgh, PA USA

***Abstract*—An ability to solve complex problems, for which a variety of solution paths are possible, is an important goal in engineering education. While feedback is critical to learning, hand grading of homework rarely provides effective, timely feedback on attempts to solve complex problems. Such feedback is also unfeasible in distance education contexts. A technology, based on the approach of cognitive tutors, is presented as a generally applicable method of providing automated feedback on complex problem solving, with truss problems studied in engineering as an example. The tutor maintains a cognitive model of problem solving for this class of problems, and associates various solution steps with distinct skills or knowledge components. One can determine whether students learn individual skills by measuring the error rate as a function of practice. Prior work has shown that for many skills the error rate indeed decreases with practice. New insight into the tutor's effectiveness, pertaining to the efficiency of student solution paths, is presented in this paper. While no explicit feedback is given regarding solution efficiency, it is found that students using the tutor become more efficient with practice. Furthermore, more efficient paths are found to be associated with making fewer errors.**

Keywords-concept inventory; interactive learning; measures of knowledge; pre-post tests; Statics; web-based courseware

I. INTRODUCTION

The development of problem-solving skills is a cornerstone of engineering education. While some problems that students learn to solve are simple, utilizing a single concept or principle, more complex problems are undertaken even in lower division courses. Students may need to coordinate and organize several concepts and steps, and many pathways to correct answers may be possible.

It is recognized in general that learning of any new skill is promoted by timely and effective feedback [1-4]. The opportunity for feedback on complex problem solving traditionally occurs through grading of handwritten homework. With weeklong turnaround such feedback is virtually never timely, nor is it readily made *effective*. Solutions can vary from one student to another, and with an incorrect answer, it is laborious for graders to identify and communicate to the student how the solution deviated from a correct path. Further, in a distance-education setting hand grading would be largely unfeasible.

This paper describes a technology that can provide students learning to solve complex engineering problems feedback on their efforts. The technology must be able to follow and assess student solutions for a variety of pathways pursued. To that end, we adapt the approach of cognitive tutors, which have been developed for computer programming [5], math [6-7], and other fields. Such tutors are based on a cognitive model for a learner encountering the chosen tasks, and so can potentially provide feedback for a range of solution pathways. There do not appear to be previous efforts to devise cognitive tutors to assist students with complex engineering problems. The feasibility of a cognitive tutor style approach to providing feedback on complex engineering problems has been demonstrated [8] through a tutor focusing on truss problems, which are commonly studied in mechanical and civil engineering. This new application of cognitive tutors is distinct from previous tutors in that student work involves the coordination of multiple open-ended, student-initiated vector diagrams and equations, all of which are interpreted on the fly in terms of a set of skills needed to properly solve the class of problems.

In the present paper, we consider in more depth the solution path taken by students solving problems with the tutor. While the tutor gives feedback on individual errors, it does not prompt students, except for rare circumstances, to think about whether their overall solution strategy is efficient. However, one can speculate that more efficient strategies may lead to fewer errors and that, in the course of solving problems, students may discover such solution strategies on their own. Here, efficiency relates to the maximum number of unsolved, yet defined, variables at any given time in the solution path. Using this definition, we investigate whether solution efficiency changes with practice and whether higher efficiency is associated with lower propensity for errors.

II. DESCRIPTION OF TUTOR FOR TRUSSES

Since use of the tutor is intended to ultimately lead to success in solving problems with paper and pencil, user interactions with the tutor should be as unconstrained as possible, provided the tutor maintains the ability to judge user

Support provided by the National Science Foundation through grant DUE 1043241 and by the Department of Mechanical Engineering at Carnegie Mellon University.

work. Although progress continues to be made in computerized interpretation of completely freeform work, for example via writing with a stylus on a tablet [9-13], such technologies may be limited for the foreseeable future; we have therefore defined unconstrained as still within the confines of a mouse and keyboard user interface.

Fig. 1 displays a typical truss problem as it would appear in a textbook. The problem consists of a set of pins (dark circles) and connected bars. There are specified forces (10 kN) and supports (idealized constraints that keep the pins in position). Fig. 2 shows a portion of a solution to the problem in Fig. 1; user input corresponding to such solution elements must be enabled by the technology. A portion of the truss (a subsystem) including point C has been singled out for attention, the unknown and known forces drawn on the diagram (a so-called free body diagram or FBD), and equations of equilibrium (imposing Newton's laws of motion) have been written. In solving truss problems, students select multiple portions of the truss, and for each subsystem draw free body diagrams and write equilibrium equations. Students must also organize the solving of equations and interpret results physically in terms of the original truss. The solver can choose any portion of the truss, write equations in any order, then choose any other portion, and so forth. The technology must grant the user latitude to pursue this large space of solution paths and still be able to judge and give feedback regardless of the path chosen.

Even within the confines of a mouse and keyboard user interface, there are a few additional intentional constraints on how closely students' action with the tutor mimic paper and pencil solving. First, to reduce the cognitive load [14-15] associated with exercising skills already mastered by the student, certain tasks have been offloaded to the tutor; for example, we removed the need to enter numbers into an electronic calculator to obtain numerical solutions. Second, motivated by the self-explanation effect [16] in educational psychology, that students who explained problems to themselves learn more, the tutor introduces selective highly targeted opportunities to make the student's thinking visible, thinking which is rarely visible in pencil and paper solving. Specifically, the tutor requests the user to designate each defined force as falling into one of several categories.

We assume that students using the tutor have learned about truss analysis through other means, such as lecture and textbook. The tutor focuses exclusively on helping students solve problems, allowing a solution process such as depicted in Fig. 2 to be conducted on the computer with as little constraint as possible, while maintaining the ability to interpret student work. Observations of student work and typical errors [17] solving truss problems have guided tutor design. The goal is to allow a student using the tutor to commit most, if not all, errors that are observed in pencil and paper solutions.

Based on an analysis of the required tasks to solve truss problems, informed by prior work on the concepts and skills needed in the overall subject in which trusses are taught [18], and typically observed student errors [17], the computer tutor limits users to the following actions:

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

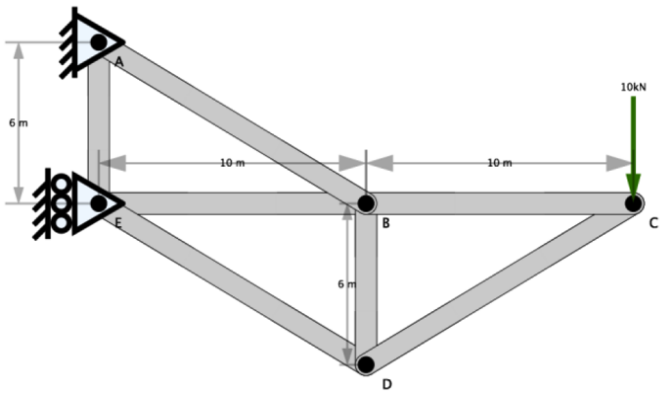

Figure 1. Typical truss problem, in which forces within the bars (members) are to be determined.

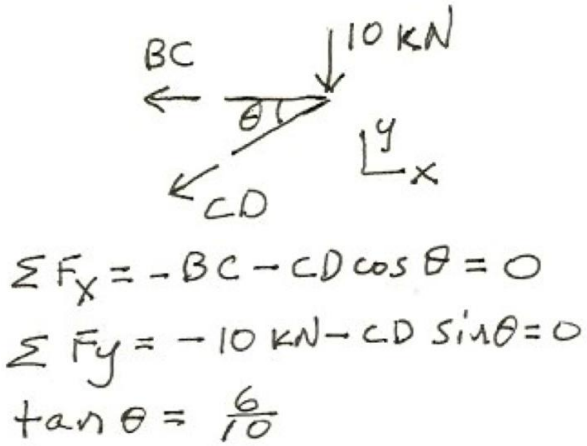

Figure 2. Portion of handwritten solution to problem in Fig. 1, showing the free body diagram of the pin at C and its connected partial bars, and associated equilibrium equations.

- 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 horizontal(x) or vertical (y) directions or parallel or perpendicular to members.

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

A screenshot of the tutor, with a problem partially solved, is shown in Fig. 3. 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 support and bar 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 to the right half of the display, and would appear as one of the thumbnails depicted in Fig. 3. Clicking on a thumbnail expands that subsystem, allowing the user to draw its FBD and write its associated equilibrium equations.

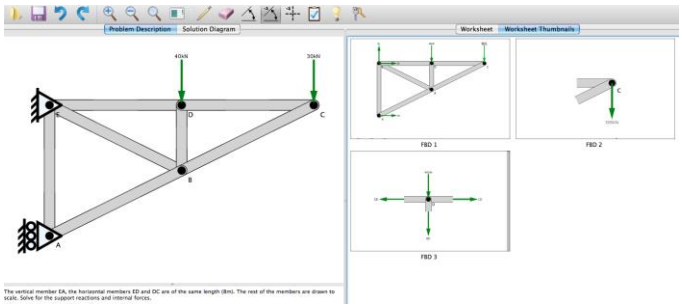

Figure 3. Screenshot of full display of truss tutor. The left panel contains the problem description and solution diagram (where users register solved forces), and the right panel comprises user-defined subsystems for analysis. A toolbar is provided at the top to assist users in problem-solving tasks, such as drawing an FBD or adding dimensions to a diagram.

In Fig. 4, we show a partially completed FBD with a new force being added to a half bar. In freeform solving of trusses, students draw arrows (for forces) and label those arrows with variables or numbers. With the tutor, we also require the user to categorize each force being drawn; the choices, shown in the window labeled "Defining a force", include: known applied force, support reaction (unknown or determined), and internal force (unknown or determined). Depending on the force category chosen, a variable label or number is required. Even though a student in freeform solving may not be thinking in terms of these categories, an expert, such as instructor, is surely clear when drawing a force which of these is being represented.

Requiring force categorization, together with the insistence on including partial members and pins in a subsystem, provides two benefits: (i) it helps students organize their thinking about the various forces in a way that can carry over to paper-and-pencil problem solving after tutor use and (ii) it establishes some clear bases for the tutor to recognize errors in student work, namely that applied and support forces can only act at pins, and internal forces can only act at the ends of partial bars.

Beneath the free body diagram (Fig. 5), the user can write equilibrium equations for the subsystem. Clicking on $\Sigma F_x = 0$, for example, initiates a place for an equation representing the summation of the horizontal (x) forces; the user then enters the equation by typing it. The user can choose to write moments about any pin (as is typical in truss analysis). Note that the interface naturally leads the user to associate any equations with a specific subsystem. Admittedly, early in the subject in which trusses are taught, students do sometimes write down equations of equilibrium without specifying the subsystem or drawing its free body diagram. The design feature of the tutor, automatically associating equations with diagrams, is an instance where the trade-off between constraint and latitude seemed to argue in favor of constraint. The task of interpreting a large set of equations, each unassociated with a free body diagram, seemed overwhelming, with tutor errors likely.

We anticipate that a typical student would use the tutor for several hours over a period of a week or at most a few weeks, depending on the class. Therefore, the tutor must be easy to learn to use. Utilizing several rounds of user testing, we have sought to make its design as intuitive and simple as possible. However, some instruction in its use will inevitably be necessary. When students first start the program, the tutor

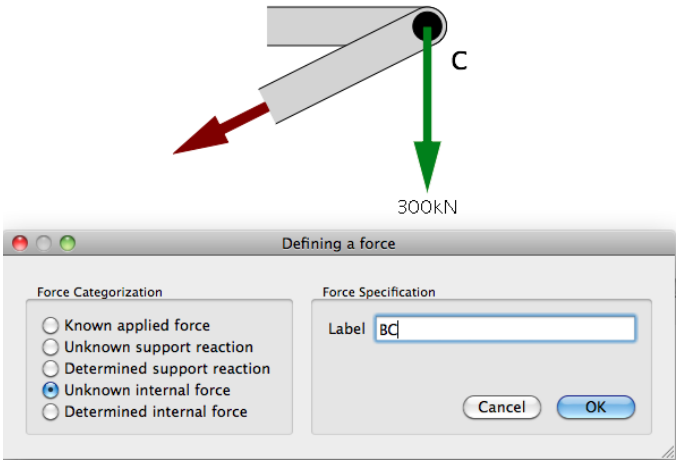

Figure 4. Screenshot of adding a force to a FBD. Upon drawing the force on the diagram (red arrow), the user is queried to categorize and specify the force.

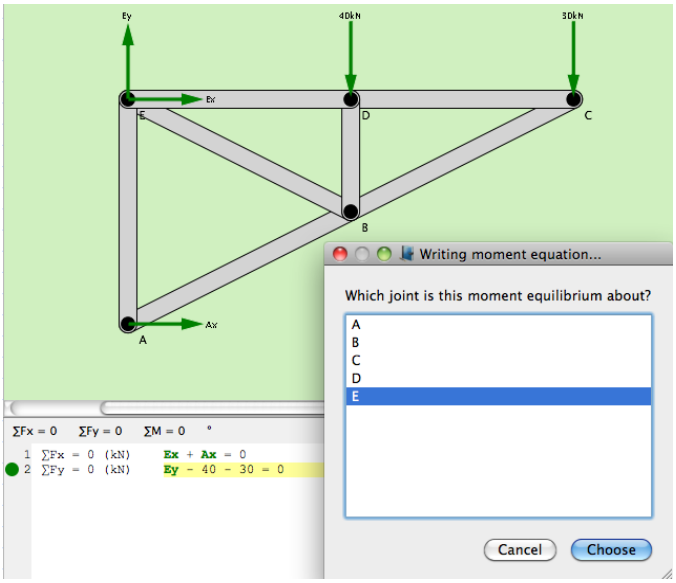

Figure 5. Screenshot of writing equilibrium equations for a given FBD. The user has already added consistent equations for the summation of forces in the x- and y-directions and is now trying to write a moment equation, which requires the selection of a moment center from the popup window.

appears with an example problem loaded, and on top of the tutor window there appears a window with a voice-over instruction video that addresses how to solve the example problem. The instruction video contains four phases, which deal with successive features of using the tutor. The video pauses after each phase, and prompts the user to go to the tutor window and carry out the portion of the solution just described in the video.

The tutor is a standalone application developed in C++; it can run on multiple platforms (Windows or Mac). While typical user interaction comprises mouse clicks and keyboard input, the tutor could be run on tablets or mobile devices using simple finger gestures, although we have presently not pursued this modality. For simplicity, we use the Qt toolkit[1] for

graphics rendering, the Boost Spirit library[2] for parsing equations, and the SymbolicC++ library[3] for solving equations. All computation is done on the fly at interactive speed with no apparent delay on a 2GHz single core desktop computer with 2GB RAM. Truss problems are created by hand and encoded as XML data. A separate XML file is generated to track solution progress; this data structure allows for easy export to the Pittsburgh Science of Learning Center DataShop tools [19] or MATLAB [20] for post-processing analysis. While specifically developed to complement statics instruction, our technology could be extended to other engineering disciplines provided a set of required learning concepts/skills is known and a cognitive model is established to enable timely, effective feedback; the specific cognitive model used in the present tutor is described below.

III. Judging Student Work and Giving Feedback

A key capability of the tutor is to judge student work and give feedback on it. The tutor does this by having a cognitive model for solving truss problems. The cognition in the tutor consists of the following algorithms corresponding to stages in the solution:

- SUBSYSTEM: An algorithm to determine if a group of pins, members, and partial members constitutes a valid subsystem.
- FREE BODY DIAGRAM: 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, but the symbols should be consistent with the first definition.
- EQUILIBRIUM EQUATIONS: Given a valid FBD, an algorithm for the correct set of terms in the summations of forces along *x*- and *y*-axes and the rotational equilibrium equation about any pin. These summations include variables and constants and must be consistent with how forces appear in the FBD.
- 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.

When to offer feedback on errors is a critical part of the tutor design. Prior research has shown that it is typically preferable to give immediate feedback [1], to ensure that the student associates the feedback with the action just taken. The tutor described here gives immediate feedback with the following caveat. Tutors for solving complex problems with limited constraints are distinct from most existing tutors: there is not a predetermined set of answers which users are expected to supply. The user is gradually adding elements of the solution on what is, in effect, a 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.

The tutor balances these competing goals by identifying natural breakpoints at which each task can be viewed as completed and thus ready to be judged. The breakpoints are: (i) the subsystem is judged after the user has selected parts and clicked on the draw subsystem button; (ii) the FBD of the subsystem is judged after the user clicks to initiate the writing of the first equation; (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, the student receives feedback that points out the error, with additional information to enable the user to fix the error and to learn why it is wrong, lessening the likelihood of repetition. 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 chosen path until detected errors are corrected. Fig. 6 displays an overview of the tutor architecture, highlighting core modules involved in the cognitive model for providing feedback.

IV. Tracking Effectiveness of Feedback

As described in prior work [8], we studied the effectiveness of the tutor's feedback in helping students reduce the frequency of errors. To analyze the progression of learning quantitatively, the terminology, methodologies, and tools from the Pittsburgh Science of Learning Center DataShop were adopted [19]. The separate skills, or knowledge components (KC), with each task of the problem solving process were tracked separately. Whether a student correctly exercises the same KC at successive opportunities is monitored. Fig. 7 shows the percentage of students who erred as a function of opportunity for a group of students in a given cohort (e.g. a class of students taught by an instructor).

For each KC, we want to determine whether the error rate decreases with practice (a sign of learning), the error rate is always rather low (the particular skill is not difficult), or the error remains high or has no discernable pattern (feedback on errors appears to have little impact on future performance). Furthermore, a logistic regression model [21], similar to those used in other cognitive tutors, was applied to the data. The statistical model predicts error fraction according to the equation,

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

where 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.

[1] http://qt.digia.com

[2] http://boost-spirit.com/home/doc/

[3] http://issc.uj.ac.za/symbolic/symbolic.html

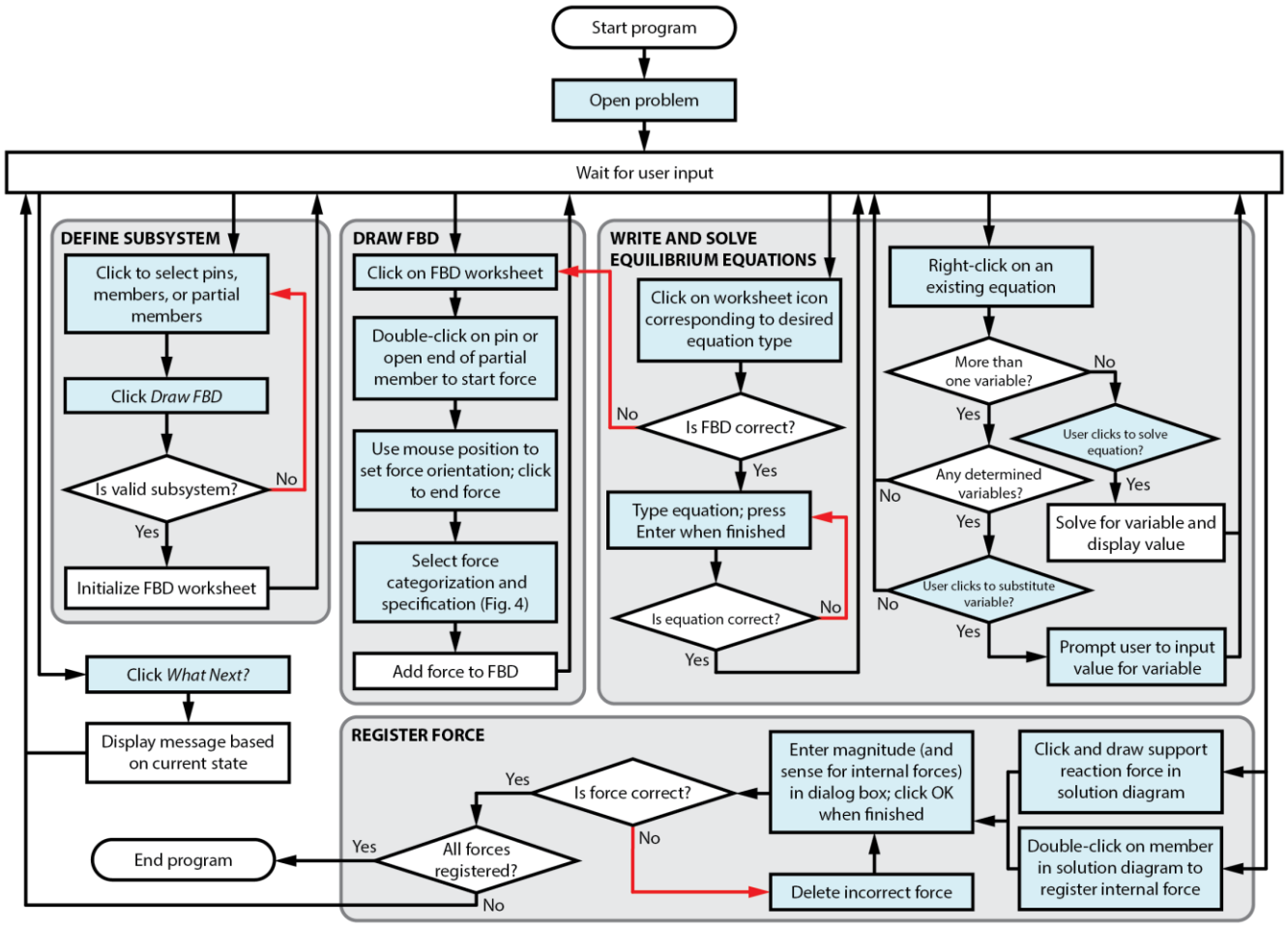

Figure 6. Flowchart of tutor processing routines. The core stages in solving truss problems are contained in gray boxes, tasks requiring user interaction are indicated by blue nodes, and user errors detected by the tutor are identified by red arrows. The parallel nature of the flowchart reflects the tutor's unique ability to allow users to explore multiple paths, only providing feedback at key junctures in the solution process. Some minor functions are not shown here for brevity; such tasks include zoom controls, adding angular dimensions to the problem diagram, and saving solution progress for future sessions or post-processing.

Fitting this model to data for a student sample yields the parameters in the statistical model. The parameter θ_i captures the overall skill level of the *i*th student. The parameter a_j, referred to as the intercept, reflects the initial probability of correctly applying the KC. The coefficient b_j, referred to as the slope, corresponds to the rate at which errors in using the *j*th KC decrease with successive opportunities to practice it. Prior work on the effectiveness of the tutor's feedback focused on the extracted values for a_j and b_j. For many of the skills, but not all, the values indicated that either a skill was relatively easy from the start (a_j high), or that the error rate reduced quickly with successive opportunities (b_j high). Relevant to the study described in the next section is the student skill parameter, θ_i. This style of evaluation of the tutor's effectiveness, comparable to that used for other cognitive tutors [19], does not involve comparison with a control group receiving some standard instruction. Rather, it seeks to determine whether improved performance on the tutor itself results from practice with the tutor.

V. Solving Efficiency Acquired Through Use of Tutor

Explicit feedback given to users pertains to errors they make along the path they choose to pursue. As pointed out above, there are many paths that a user can choose. A path is described by the sequence of joints (each joint is a single pin and its connecting bars) that is chosen, and then, for each joint, the sequence in which the equilibrium equations are written. For each new joint, there are two or more unknowns, which correspond to the unknown forces in the connecting bars. But, for each joint there are only two equations corresponding to equilibrium in the *x*- and *y*- directions. Thus, for each joint, the user can solve for at most two additional unknowns.

One strategy would be to choose joints at random, draw each FBD, and write down two equations of equilibrium for each. Then, one either substitutes into a solver for linear equations (not possible with the tutor and not available for students solving truss problems in exams), or one finds an equation that is solvable (only two unknowns), solves it and

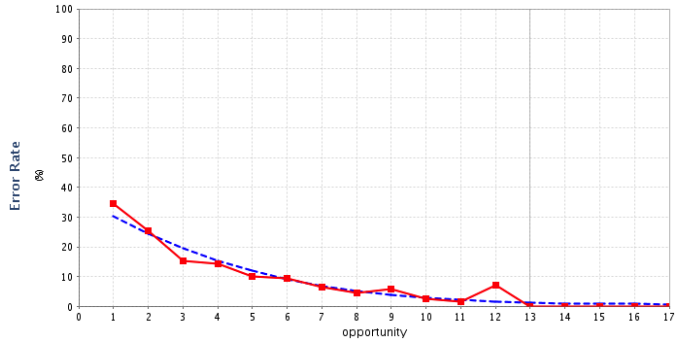

Figure 7. Percentage of students in error plotted as a function of opportunity (learning curve) on a skill (representing a determined support reaction) for which the error rate is initially high, but decreases with practice.

then substitutes sequentially into other problems. An alternative strategy would be to focus first on a joint that will be solvable, draw its FBD, write its equations, and then solve for the unknowns. The solved values can be used in analyzing the next chosen joint. We consider the latter approach a more efficient strategy. We hypothesize that, with it, one is less likely to get lost in the solution, and that one would also make fewer errors.

An instructor teaching students to solve truss problems may suggest that joints be chosen with an eye to introducing only two unknowns at a time. But, with the exception of one uncommon situation described below, the tutor does not directly promote a more efficient strategy or give feedback on the strategy. However, for various reasons, for example because it is messy or frustrating to have too many unsolved equations, students may with time naturally gravitate towards a more efficient strategy. Certainly, another measure of the success of the tutor would be if students developed such an approach through using the tutor.

The efficiency of a solution path was studied by tracking the number of open (unsolved) variables. Variables are created when the user draws and labels undetermined forces on a FBD and remain open until the user registers a value for the force in the solution diagram. An efficient path, as defined above, should minimize the number of unsolved variables at any given time. The results presented here are based on data obtained from a class of 48 students enrolled in a regularly scheduled semester long statics course, 40 of whom completed all three of the problems studied here. Students used the tutor in lieu of a weekly homework assignment. Typical traces are shown in Fig. 8-10. User actions, as described earlier, include selecting a subsystem, drawing a FBD, writing an equilibrium equation, solving an equilibrium equation, and registering a determined force.

All problems can be solved with no more than three unknowns present at any time. Depending on the locations of the supports, one may have to analyze the entire truss prior to analyzing a joint; then there are three equilibrium equations (rotational equilibrium, as well as *x*- and *y*-forces). Therefore, a coarse estimate of efficiency relates to the maximum number of unsolved variables in a solution trace. We computed how many students solved each problem with no more than three unknowns present at any time. The efficiency rate, or fraction

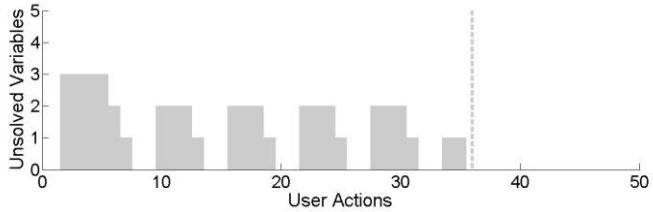

Figure 8. Example of an efficient solution path: the repeated sequence of steps (select subsystem, draw FBD, write and solve equations, register force) results in the "staircase" effect demonstrated here. The dotted line refers to problem completion.

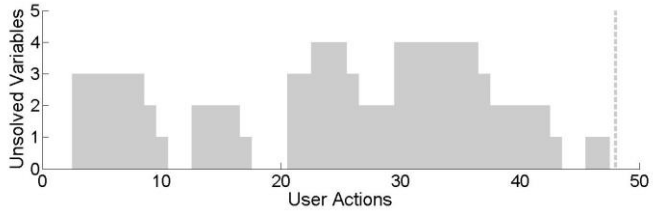

Figure 9. Example of an inefficient solution path: this user started in an efficient manner, but became inefficient after completing two subsystems. The dotted line refers to problem completion.

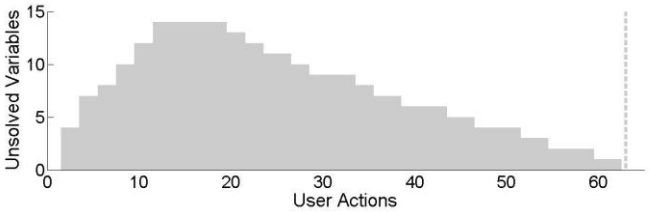

Figure 10. Example of an inefficient solution path: this user solves the entire problem in parallel fashion, drawing all possible FBDs before solving for any forces. The dotted line refers to problem completion.

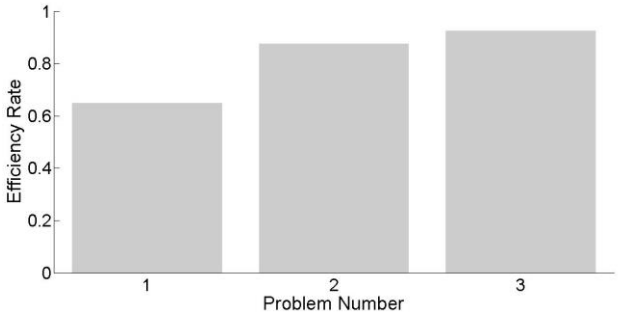

Figure 11. Fraction of students per problem who maintain no more than three unknowns throughout the solution path.

of students who were efficient, for the three successive problems is shown in Fig. 11. The results suggest that users, on average, did in fact become more efficient in the second problem compared to the first. (There is little change from the second to the third problem.) The difference in the efficiencies

of the first and second problem is indeed statistically significant ($z = 2.45$, $p = 0.014$).

There is one rare circumstance when the tutor does promote efficient solving. When a user decides he or she is stuck, there is a button entitled *What Next?*. The message given is context dependent, and it often involves finishing work that has been started. However, if all initiated work is completed, the message tells the user to select a new joint, ideally one that has at most two unknowns. Of students in this sample, only eight clicked on the *What Next?* button, and of them only two received the particular message about choosing a new joint that is solvable, a strategy which is necessary, but not sufficient, to solving efficiently. Those two students received the message in the first problem; they were efficient in the second and third problems.

Finally, it was of interest to determine the relationship between choosing an efficient path and the propensity for errors, as measured by the overall student skill parameter, θ_i, in the statistical model described in Section IV. Fig. 12 displays this relationship by plotting the individual ability of each student (skill parameter θ_i) against the number of problems they solved efficiently. The results demonstrate moderate positive correlation ($r = 0.422$) and are statistically significant ($p < 0.01$). Because the statistical model described in Section IV has only a single student parameter, we do not know whether individual students improve at different rates, even though for many skills we know that students improve with practice. Thus, the question of whether being coaxed to a more efficient path would reduce the number of errors a student makes – perhaps because the work space is less messy – cannot be answered without further study.

VI. Summary and Conclusions

Complex problems that engineering students learn to solve often 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. In this paper, we have described a technology, suitable for a distance education context, which can provide students learning to solve complex engineering problems feedback on their efforts.

By adapting the approach of cognitive tutors, we have developed a computer tutor that has a cognitive model of a student engaged in solving the problems of interest. The tutor interface permits the user to solve problems correctly following any pathway and to commit commonly observed errors. The cognitive model and the judicious timing of feedback give the user wide latitude to pursue different solution pathways, prevent new work from building upon previously committed errors, and still provide feedback on errors to enable students to complete most problems correctly.

As previously described, metrics for judging the effectiveness of feedback have been devised by viewing the solving of problems as a set of distinct skills or knowledge components. Actions by the student constitute opportunities to exercise different knowledge components; the effectiveness of feedback is quantified based on whether fewer students incur

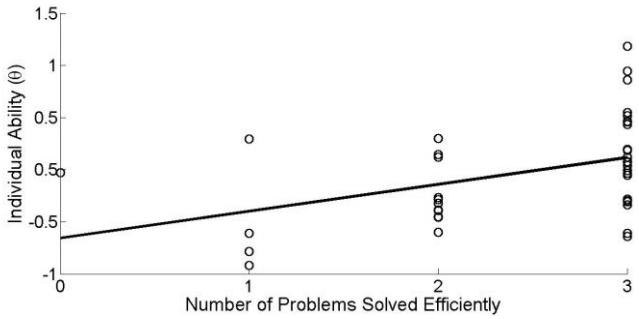

Figure 12. Correlation between individual ability and the number of problems solved efficiently. Each data point represents one student user.

errors with successive opportunities on each knowledge component. The data from a cohort of students had been fit to a statistical model that predicts the percent error as a function of opportunity. The model has several parameters, including initial difficulty and rate at which errors decrease with opportunity for each knowledge component. Fitting these parameters to the data provided evidence that the students do learn: that is, their propensity for errors decreases with practice. The statistical model also includes a single parameter for each student in the cohort, corresponding to the overall skill level.

The present paper has focused on a different aspect of student solutions using the tutor, namely on their overall solution path. By contrast to errors, upon which students receive feedback, the tutor does not give feedback to students on their overall solution path, nor does the tutor promote efficiency in the path. However, students may, through the additional effort needed when solving problems inefficiently, discover more efficient paths on their own. Furthermore, efficiency in the solution path may be associated with fewer errors.

To that end, a measure of efficiency involving the maximum number of unsolved variables at any instant was defined. It was found that the fraction of students who solved efficiently increased by a statistically significant amount from the first problem to the second problem, and held steady in the third problem. Furthermore, efficiency in solution was found to correlate significantly and positively with student skill, as measured by the overall skill level extracted from the statistical model. Thus, perhaps to reduce their workload, students do become more efficient with practice. Furthermore, higher efficiency is associated with making fewer errors. The possibility of causal relationships between efficiency and errors remains to be studied.

In summary, the technology described here has demonstrated that students can be given sufficient feedback, while working without an instructor, to complete complex problems, that their errors reduce with practice, that they become more efficient by virtue of their practice, and that such efficiency is associated with fewer errors. Moreover, because the technology maintains a detailed record of student work, and interprets that work meaningfully by way of a cognitive model, it enables the learning process to be studied and the tutor itself to be systematically improved with time.

ACKNOWLEDGMENT

We thank Luoting Fu for his prime role in developing the tutor, and Jackie Yang, Jeremy Jiang, and Rebecca Piston for their assistance in development and initial testing of the tutor.

REFERENCES

[1] J. Hattie and H. Timperley, "The power of feedback," Rev. Educ. Res., vol. 77, no. 1, pp. 81-112, March 2007.

[2] J. R. Anderson, F. G. Conrad, and A. T. Corbett, "Skill acquisition and the LISP tutor," Cogn. Sci., vol. 13, no. 4, pp. 467-505, 1989.

[3] R. L. Bangert-Drowns, C.-L. Kulik, J. A. Kulik, and M. Morgan, "The instructional effect of feedback in test-like events," Rev. Educ. Res., vol. 61, no. 2, pp. 213-238, Summer 1991.

[4] A. T. Corbett and J. R. Anderson, "Locus of feedback control in computer-based tutoring: impact on learning rate, achievement and attitudes," Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 245-252, ACM, 2001.

[5] J. R. Anderson, C. F. Boyle, and B. J. Reiser, "Intelligent tutoring systems," Science, vol. 228, pp. 456-468, 1985.

[6] K. R. Koedinger, J. R. Anderson, W. H. Hadley, and M. A. Mark, "Intelligent tutoring goes to school in the big city," Int. J. Artificial Intelligence in Education, vol. 8, pp. 30-43, 1997.

[7] K. R. Koedinger, "Toward evidence for instructional design principles: examples from Cognitive Tutor Math 6," Proceedings of PME-NA XXXIII, Annual Meeting of the North American Chapter of the International Group for the Psychology of Mathematics Education, pp. 21-49, 2002.

[8] P. S. Steif, L. Fu, and L. B. Kara, "The potential for computer tutors to assist students learning to solve complex problems," Proceedings of the American Society for Engineering Education Annual Conference and Exposition, Indianapolis, Indiana, June 2014.

[9] L. B. Kara and T. F. Stahovich, "Hierarchical parsing and recognition of hand-sketched diagrams," Proceedings of the 17th annual ACM symposium on User interface software and technology (UIST '04), New York, NY, pp. 13-22, ACM, 2004.

[10] J. LaViola, "Advances in mathematical sketching: moving toward the paradigm's full potential," IEEE Computer Graphics and Applications, vol. 27, no. 1, pp. 38-48, January/February 2007.

[11] J. Peschel and T. Hammond, "STRAT: a sketched-truss recognition and analysis tool," International Workshop on Visual Languages and Computing, Boston, MA, pp. 282-287, 2008.

[12] W. Lee, R. de Silva, E. J. Peterson, R. C. Calfee, and T. F. Stahovich, "Newton's Pen: a pen-based tutoring system for statics," Computers & Graphics, vol. 32, no. 5, pp. 511-524, 2008.

[13] L. Fu and L. B. Kara, "From engineering diagrams to engineering models: Visual recognition and applications," Computer-Aided Design, vol. 43, no. 3, pp. 278-292, 2011.

[14] J. Sweller, "Cognitive load during problem solving: effects on learning," Cogn. Sci., vol. 12, pp. 257-285, 1988.

[15] J. Sweller, "Cognitive load theory, learning difficulty and instructional design," Learning and Instruction, vol. 4, pp. 295-312, 1994.

[16] M. T. H. Chi, M. Bassok, M. W. Lewis, P. Reimann, and R. Glaser, "Self-explanations: how students study and use examples in learning to solve problems," Cogn. Sci., vol. 13, pp. 145-182, 1989.

[17] P. S. Steif, L. Fu, and L. B. Kara, "Development of a cognitive tutor for learning truss analysis," Technical Report, Carnegie Mellon University, 2013, unpublished.

[18] P. S. Steif, "An articulation of the concepts and skills which underlie engineering statics," 34th ASEE/IEE Frontiers in Education Conference, Savannah, GA, 2004.

[19] K. R. Koedinger, R. S. J. d. Baker, K. Cunningham, A. Skogsholm, B. Leper, and J. Stamper, "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, 2010.

[20] MATLAB, version 8.1.0.604 (R2013a). The MathWorks, Inc., Natick, Massachusetts, 2013.

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