Visualizing and Classifying Multiple Solutions to Engineering Design Problems

Elena Glassman
MIT CSAIL, UID Group, Rm G707
32 Vassar St., Cambridge, MA 02139, USA
ELG@MIT.edu

ABSTRACT

In engineering design courses, many problems have a specification that the student's implementation must meet, but give the student a broad range of freedom for the internal design of that implementation. There may be several distinct, correct strategies for solving them, some of which may be unknown to the teaching staff or intelligent tutor designer. Visualizing and classifying the multiple solutions that students generate in response to assigned engineering design problems will improve hints and answers to students' questions, whether they are provided by peers, staff, or automation. I log incremental snapshots of students' solutions as they progress toward correct and incorrect solutions. Initial investigations demonstrate that the choice of features to represent solutions is critical, and may be domain- or problem-dependent.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education—*computer science education*

Keywords

Problem Solving Process, Pattern Recognition

1. RESEARCH SITUATION

I am a doctoral candidate in MIT's Electrical Engineering and Computer Science (EECS) Department. I have completed all of my qualifying exams, and am now in my fourth year. I discuss my initial thesis research investigations with several professors, including my advisor, Rob Miller. Together, they cover the fields of machine learning, human computer interfaces, and instructional design. I intend to formalize the relationship by the end of this summer, with the submission of my thesis proposal. I plan to analyze and present the data I collect from students over the next two years, in order to graduate in May 2015.

ICER'13, August 12–14, 2013, San Diego, California, USA.
ACM 978-1-4503-2243-0/13/08 ...$15.00
http://dx.doi.org/10.1145/2493394.2493421.

In engineering design courses, many problems have a specification that the student's implementation must meet, but give the student a broad range of freedom for the internal design of that implementation. There may be several distinct, correct solutions, some of which may be unknown to the teaching staff or intelligent tutor designer. This raises problems for helping students and giving feedback. In face-to-face situations, if a teaching assistant doesn't recognize the student's solution path, then they may redirect the student completely, costing them work and possibly derailing a novel, valid solution. In an intelligent tutor or massively open online course (MOOC), the automated hint generators may not recognize the unexpected solution paths, and will generate unhelpful hints.

A first step toward discovering these alternate correct solution paths is visualizing many student solutions together, so that the teaching staff can understand the space of paths leading to correct solutions. I have taken this approach in 6.004, an undergraduate computer architecture (CompArch) course. I have also observed this approach in a collaborator's work on an online Matlab programming challenge. Plotting dynamic behavior or static features such as parse tree size is enough, in these initial examples, to separate students' solutions into clear clusters representing different strategies. In the Matlab challenge, visualizing code size provides insight into common strategies as well as successful and unsuccessful outliers. In CompArch, this led to better education of the teaching staff. Some now ask a simple question to identify the student's approach before trying to help them.

2. CONTEXT AND MOTIVATION

I am an instructor for an undergraduate introductory course on computer architecture (CompArch). Roughly two hundred students enroll per semester.

One mid-semester lab assignment in this course requires that students create state-transition rules for a Turing machine. Many distinct sets of state-transition rules behave identically, given the same input tape.

The dynamic behavior of students' Turing machines has recurring patterns. I visualized this dynamic behavior for 148 students' two-state Turing machines, and by visual inspection of the Turing machine's movement across a common input tape, identified strategies [1]. The majority (88%) of the solutions employed one of two mutually exclusive strategies: (1) matching the innermost open parenthesis with the innermost closed parenthesis and (2) matching the n^{th} open parenthesis with the n^{th} closed parenthesis.

The fact that there were *two* correct solutions to the Turing machine assignment came as a surprise. I now ask struggling students a question first, to determine the solution they are aiming for. I can then suggest edits that preserve the chosen solution, if I recognize it as correct. Common solution identification also allows me to recognize novel alternative solutions.

This is one example of an engineering design problem where a small number of common, distinct, correct solutions are distinguishable once an appropriate representation of the data is found. My thesis research explores the generality of this phenomenon, with a focus on scaling up to online environments with thousands of students.

3. BACKGROUND & RELATED WORK

Helminen et al. [2] introduced interactive graphs for examining the problem solving process of students working on small programming-like problems. In contrast to our work, problems with multiple solutions were outside the scope of their investigation.

Taherkhani et al. [4] demonstrated the practicality of differentiating between multiple solutions, i.e., different sorting algorithms, in students' solutions to a particular engineering design problem using a supervised machine learning method.

My thesis work complements the work of Piech et al. [3]. They found distinct development paths to solutions by putting students' incremental solution attempts into a pipeline involving classification, milestone discovery, Hidden Markov Modeling of the students' process, and clustering of solution paths. Evaluation focused on predicting midterm exam grades and detecting milestone difficulty. It is not used to help new students through the same lab exercise, in the absence of staff assistance.

Peer-pairing can stand in place of staff assistance. Weld et al. speculate about peer-pairing in MOOCs based on student competency measures [5].

4. STATEMENT OF THESIS/PROBLEM

My thesis statement is: Visualization and automated classification of the multiple solutions that students generate in response to assigned engineering design problems will improve hints and answers to students' questions, whether they are provided by peers, staff, or automation.

I plan to explore the following aspects of this claim:

- What features are useful for visualizing or automatically clustering engineering design solution paths? For programming domains, for example, features could include measures of program complexity, stack depth, and runtime characteristics. For digital logic and analog circuit domains, features could include graph metrics and voltage traces on intermediate nodes.
- How can teaching staff be trained to quickly recognize the solution path of a given student, in order to give tailored feedback?
- If teaching staff are in short supply (as in a MOOC), how can peers help each other in a space where there are multiple good solution paths?
- If peer help is not feasible, then how can we provide automated help based on solution path recognition in a design space where multiple correct paths are possible?

5. RESEARCH GOALS & METHODS

I will design an interactive data visualization environment for teaching staff. By interacting with representations of many student solutions at once, staff can visually identify and investigate patterns. If staff independently find similar meaningful, persistent patterns in their students' solutions using the tool, I will consider it a success.

For peer-pairing, there are a variety of solution path-dependent pairing strategies. One could pair students on the same solution path: students who have implemented a recognized solution or fixed a particular bug would be paired with another student who is struggling to implement the same solution or fix the same bug. By comparing metrics, e.g., grades, of students paired based on solution paths, relative to random pairing and no pairing, I can measure the interventions' educational value.

In CompArch, we capture snapshots of students' intermediate code as they work. These snapshot sequences are solution paths ready for analysis. One semester's data has been collected already. I hope to demonstrate generality on similar datasets from other engineering domains.

6. EXPECTED CONTRIBUTIONS

This research may be particularly helpful to staff who focus on helping each student reach their own envisioned solution. The algorithms, tools, visualizations, and resulting insights from this work are intended to support this constructivist approach to teaching, facilitate students helping each other, and inform automated feedback.

Acknowledgments This work is supported by the National Science Foundation Graduate Research Fellowship under Grant No. 1122374.

7. REFERENCES

[1] E. L. Glassman, N. Gulley, and R. C. Miller. Toward facilitating assistance to students attempting engineering design problems. In *Proceedings of the Tenth Annual International Conference on International Computing Education Research*, ICER '13, New York, NY, USA, 2013. ACM.

[2] J. Helminen, P. Ihantola, V. Karavirta, and L. Malmi. How do students solve parsons programming problems? an analysis of interaction traces. In *Proceedings of the Ninth Annual International Conference on International Computing Education Research*, ICER '12, pages 119–126, New York, NY, USA, 2012. ACM.

[3] C. Piech, M. Sahami, D. Koller, S. Cooper, and P. Blikstein. Modeling how students learn to program. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, SIGCSE '12, pages 153–160, New York, NY, USA, 2012. ACM.

[4] A. Taherkhani, A. Korhonen, and L. Malmi. Automatic recognition of students' sorting algorithm implementations in a data structures and algorithms course. In *Proceedings of the 12th Koli Calling International Conference on Computing Education Research*, pages 83–92. ACM, 2012.

[5] D. Weld, E. Adar, L. Chilton, R. Hoffmann, E. Horvitz, M. Koch, J. Landay, C. Lin, and Mausam. Personalized online education – a crowdsourcing challenge. In *Proceedings of the 4th Human Computation Workshop (HCOMP '12) at AAAI*, 2012.