Work in Progress: Parsons Problems as a Tool in the First-Year Engineering Classroom

Brooke C Morin, The Ohio State University

Brooke Morin is a Lecturer in the College of Engineering at the Ohio State University, teaching First-Year Engineering for Honors classes in the Department of Engineering Education. Brooke earned her bachelor's degree and master's degree in Mechanical Engineering at Ohio State.

Dr. Krista M Kecskemety, The Ohio State University

Krista Kecskemety is an Assistant Professor of Practice in the Department of Engineering Education at The Ohio State University. Krista received her B.S. in Aerospace Engineering at The Ohio State University in 2006 and received her M.S. from Ohio State in 2007. In 2012, Krista completed her Ph.D. in Aerospace Engineering at Ohio State. Her engineering education research interests include investigating first-year engineering student experiences, faculty experiences, and the connection between the two.

Dr. Kathleen A Harper, The Ohio State University

Kathleen A. Harper is a senior lecturer in the Department of Engineering Education at The Ohio State University. She received her M. S. in physics and B. S. in electrical engineering and applied physics from Case Western Reserve University, and her Ph. D. in physics from The Ohio State University. She has been on the staff of Ohio State's University Center for the Advancement of Teaching, in addition to teaching in both the physics and engineering education departments. She is currently a member of the ASEE Board of Directors' Advisory Committee on P-12 Engineering Education.

Mr. Paul Alan Clingan, The Ohio State University

Senior Lecturer Department of Engineering Education

Work in Progress: Parsons Problems as a Tool in the First-Year Engineering Classroom

Introduction

Teaching coding in a common first-year engineering program presents a variety of challenges. Students arrive with varied coding experience, from those who have never coded before to those proficient in several languages. Many students whose intended majors are perceived to involve less coding struggle to understand why coding instruction is part of their first-year experience. Additionally, it can be challenging to balance syntax instruction with the development of programming logic and to incorporate first-year program goals such as problem solving, teamwork, and communication.

In order to address some of these concerns, the first-year engineering program at the Ohio State University recently introduced Parsons Problems and Parsons-Problem-like activities into the coding instruction. Parsons Problems present students with segments of code from a complete or partial program that are scrambled out of order [1]. Students are tasked with placing the code segments in order to recreate the original program. Research on Parsons Problems has suggested that completing these activities may have the same learning gains as writing code from scratch [2],[3], but with a reduced cognitive load that leaves room for learning [4]. They allow students to focus on the structure and logic of a program independent of the particulars of syntax. Some authors have presented variations on the Parsons Problems, such as including incorrect or unnecessary code segments, called distractors [1]; providing a framework for the general structure of the code [5]; and using custom software to provide real-time feedback [6], [7]. However, the use of Parsons Problems in the first-year engineering classroom is largely unreported, as is any attempt to incorporate groupwork into the activity.

This paper presents an effort to incorporate Parsons Problems into a first-year engineering course sequence including the novel use of paper-based problems and groupwork. It also provides analysis of the preliminary feedback provided by instructional staff and students. The authors will detail several approaches, difficulties, and formats for these activities, discussing which components were successful and what components should be further developed. This work-in-progress paper represents the first step toward analyzing the perception of and effectiveness of these activities which will be completed in future work.

Methods

Course Structure

The first-year engineering program at the Ohio State University consists of a two-semester sequence. All incoming engineering freshmen participate in this course sequence, regardless of intended engineering discipline. The first semester focuses on problem solving, primarily within the context of MATLAB coding. This paper primarily focuses on the honors version of the course, which also provides instruction in C/C++. Each section has approximately 36 students. In most sections, students are seated at tables of four with a desktop computer at each seat (Figure 1). A whiteboard on the table facilitates group work and communication.

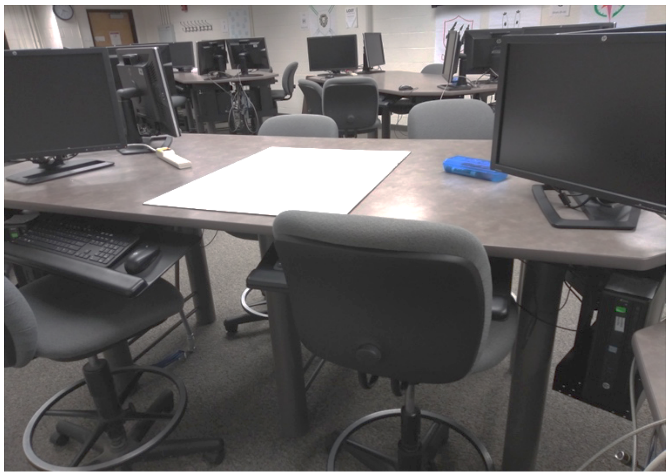


Figure 1. The first-year engineering students sit at a table of four, with computers for individual work and a whiteboard and plenty of space for collaboration.

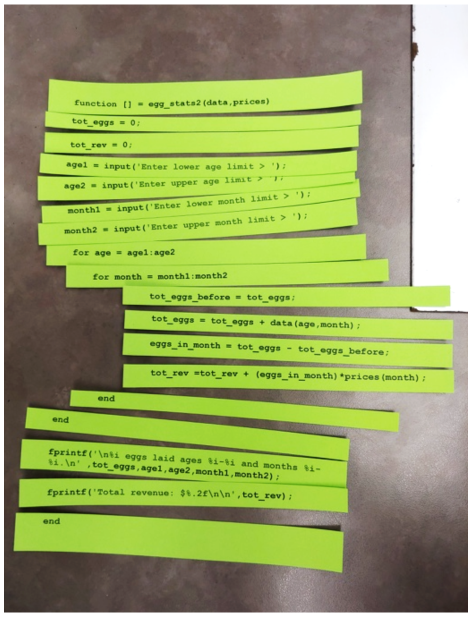
The program currently employs an inverted classroom structure [8], [9]. The students complete preparation work, which is a combination of videos, reading, and tutorials, at home before class. The instructor presents a short lecture at the beginning of class, reviewing key topics and addressing points of concern, and then the remainder of the class time (125 minutes) is dedicated to activities and coding assignments. A teaching staff consisting of one instructor and two undergraduate teaching assistants are present during this time to assist the students.

All students in the first-year program practice structured approaches to engineering problem solving before beginning programming instruction in MATLAB. The material is initially presented following a more traditional computer science approach that transitions to more MATLAB-specific features such as vector and matrix manipulation. Instruction then shifts to C/C++.

Parsons Problems

As the first few weeks of the course are devoted to other fundamental engineering topics such as structured approaches to problem solving and technical communication, Parsons Problems were introduced starting the fifth week of instruction and continuing for a total of 10 “Weekly Activities”. In order to facilitate a groupwork approach, students were provided with a packet that contained an instruction sheet and physical strips of paper to reorder to create the final code (Figure 2). The structure of the activities varied based on content and to reduce tedium but could broadly be considered to fit in four categories: (1) Problems with all code segments scrambled and no comments. Students were provided with a description of the function of the code and were tasked with forming the full program from the provided code segments. (2) Problems with all code segments scrambled and some comments provided, also scrambled but printed on a different color of paper. (3) Problems with some code scrambled and some code required to be written by the students. (4) Problems with a partial program and/or structure printed on full sheets of paper and scrambled code segments to fill in the blanks.

```
function [] = egg_stats2(data,prices)
age1 = input('Enter lower age limit > ');
age2 = input('Enter upper age limit > ');
month1 = input('Enter lower month limit > ');
month2 = input('Enter upper month limit > ');
tot_eggs = 0;
tot_rev = 0;
for month = month1:month2
tot_eggs_before = tot_eggs;
for age = age1:age2
tot_eggs = tot_eggs + data(age,month);
end
eggs_in_month = tot_eggs - tot_eggs_before;
tot_rev = tot_rev +
(eggs_in_month)*prices(month);
end
fprintf('\n%i eggs laid ages %i-%i and months
%i-%i.\n', tot_eggs,age1,age2,month1,month2);
fprintf('Total revenue: $%.2f\n\n',tot_rev);
end
```



```
function [] = egg_stats2(data,prices)
tot_eggs = 0;
tot_rev = 0;
age1 = input('Enter lower age limit > ');
age2 = input('Enter upper age limit > ');
month1 = input('Enter lower month limit > ');
month2 = input('Enter upper month limit > ');
for age = age1:age2
    for month = month1:month2
        tot_eggs_before = tot_eggs;
        tot_eggs = tot_eggs + data(age,month);
        eggs_in_month = tot_eggs - tot_eggs_before;
        tot_rev =tot_rev + (eggs_in_month)*prices(month);
    end
end
fprintf('\n%i eggs laid ages %i-%i and months %i-
%i.\n' ,tot_eggs,age1,age2,month1,month2);
fprintf('Total revenue: $%.2f\n\n',tot_rev);
end
```

Figure 2. A portion of the Weekly Activity for week 8. The original MATLAB function is given on the left and the corresponding code segments, placed in order, are on the right.

As the semester progressed, the intended difficulty of the problems increased. Initial problems were simple programs with minimal repetition and selection. Later problems were more complex and often contained distractors, which were logically or syntactically incorrect code segments, aimed at correcting common errors. Some weeks contained only one main function, while others had multiple color-coded components such as user-written functions. (Table 1) The activities were designed to synthesize the topics covered during the week and to address common logic and syntax mistakes. Depending on the complexity and structure of the activity, students worked in groups of 2 (both students on one side of the table) or 4 (all students at a table) to unscramble the code. They were encouraged to pass the code segments back and forth, discuss the placement of the code segments, and evaluate the logic of their proposed solution.

Table 1. The ten Weekly Activities varied in type and format. Through the semester, the activities generally became more complex and were varied in form to reduce tedium.

Week	Language	Topic	Type	Distractors?
5	MATLAB	I/O, File I/O	1	No
6	MATLAB	Repetition, Graphing, Logical Operators	1	Yes
7	MATLAB	Functions, Vectors, Matrix Math	1	Yes
8	MATLAB	Vector/Matrix Extraction, Special Features	1	Yes
9	C/C++	Intro to C, I/O	2	Yes
10	C/C++	Repetition, Selection, Arrays, File I/O	1	Yes
11	C/C++	Pointers, Strings	2/3	Yes
12	C/C++	Time-Aware Programming, Functions	3	No
13	C/C++	Structs	4	No
14	C/C++	Object-oriented Programming	4	Yes

Once the students had reached what they believed to be the correct ordering of the code segments (Figure 3), the instructor presented one correct solution. The class then discussed what sections of code could move to produce the same outcome and what code progression must be preserved to maintain the appropriate logic.

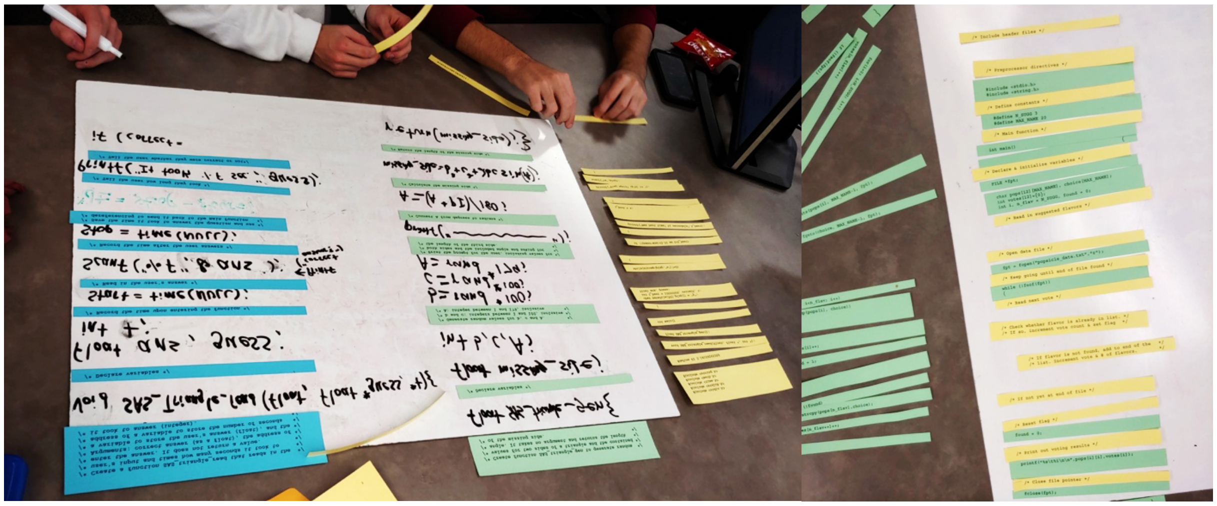


Figure 3. Students worked together to solve the problems. Both a Type 3 problem, where students added written code to scrambled comments (left) and a Type 2 problem, where the students were interweaving code and comments (right) are shown.

Evaluation

As this was the first semester attempting to incorporate Parsons Problems, the faculty discussed each activity at their weekly course meetings. The length and difficulty of the assignment, along with perceived student engagement, were discussed. Additionally, students had two opportunities to provide feedback about their experiences. The students complete a weekly anonymous journal in response to a prompt regarding aspects of the course along with other topics relevant to first-year students [10]. One such prompt asked students to evaluate their experiences with the first activity and provided sufficient positive feedback to encourage future activities. Additionally, the weekly activities were mentioned on the end-of-course survey that all students complete. This feedback informed changes to the upcoming activities in real time as well providing a basis on which to modify this component of the course for future semesters. The remainder of this paper focuses on the end-of-course feedback.

Results and Discussion

This first implementation of the Parsons Problems as an activity was primarily to explore a new evidence-based teaching tool and to gauge initial faculty and student response to the problems. Thus, there are limited quantitative results available at this time. The remainder of this section presents the available quantitative results along with qualitative assessments.

Student Engagement

The faculty response indicated that most students participated in the Parsons Problem activities. Faculty reported observing students engaging with one another at the table, discussing logic and working together to identify which things must "go together" based on logic and syntax patterns.

The students changed seats weekly, which prevented a given pair or table from always containing one or two strong programmers who did most of the work and allowed other students to "sit back." Additionally, some of these more experienced programmers did not wish to engage in the activity as they thought it would not benefit them and had to be encouraged to participate.

A particularly interesting observation during the activities occurred when students received code for which no indentation was provided. Typically, the indentation in well-formatted code makes the code more readable because this structure allows one to interpret which lines of code belong to the same block. Faculty observed students creating their own indentation for the strips of paper to facilitate their understanding of the code's organization (Figure 2 in Methods).

However, there were obvious signs of student disengagement under certain circumstances. Some activities were too long or complex for the students to complete and understand in a reasonable time. When this happened, students expressed frustration for two reasons. First, those students who struggled to understand the problem were frustrated that they could not solve it and often remained confused after being guided toward the correct answer. Under these circumstances, the activities may not have appreciably reduced cognitive load as intended. Second, students who could solve the problem were frustrated waiting for other students to finish.

Student Feedback

The students' feedback was polarized. Student responses included that the students liked the activities because "they allowed me to focus on the logic and theory behind different concepts without worrying about syntax" and "I wish we had one of these everyday". Meanwhile, other students provided responses such as "it was harder to problem solve using somebody else's code rather than writing your own and understanding it your own way" and "they were more tedious and difficult than constructive". A summary of the survey results, grouped into positive, negative, and mixed responses, is presented in Figure 4.

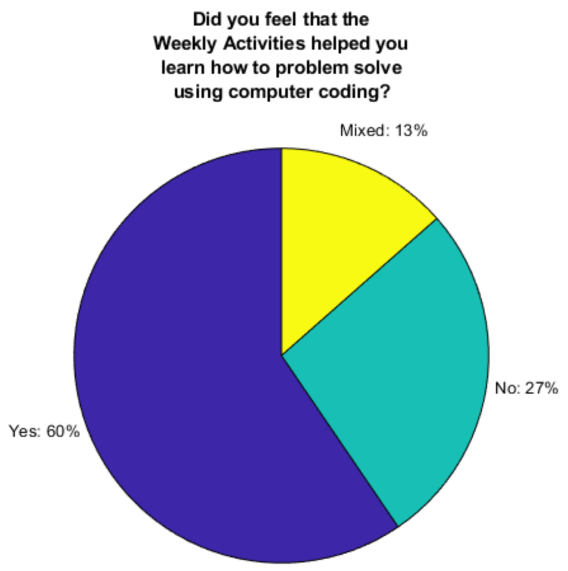


Figure 4. Most students responded positively to the Parsons Problems, with 60% writing that the activities helped them learn how to problem solve using computer coding.

Those students who responded negatively or provided a mixed response often referenced one of a few concerns: (1) Many failed to see the link between the current activity and writing a program themselves. One student compared the activity to a collegiate football coach asking his quarterback to practice using a football videogame. (2) They were concerned with the amount of time the activity took in the classroom. Though the problems were intended to take approximately 20 minutes and not exceed 30, that did not always happen. This was largely a fault in the problem design, and it caused many students to dismiss the activities as a whole. (3) Many students stated that they did not like following the logical or structural choices that another person made. One student wrote, "My main qualm with these activities is that, in my eyes, coding is a very individual activity; in other words, there are tons of solutions to a singular problem, and people's solutions often differ by a great deal. I find it sort of ridiculous that I was forced into thinking of only one solution for these problems. I didn't usually think of the problem in the same way that we were guided to think about this, so I didn't think that these activities were particularly helpful." (4) Students often felt that they were not getting enough support or feedback during the activity. Some instructors treated the activity as a race, which made the activity more stressful for several students who said it impeded their learning. Other students felt that not enough time was devoted to discussing the solutions or receiving assistance during the activity. Some students who responded positively indicated that frequent feedback concurrent with the solving process made the activity more beneficial. (5) Some activities involved topics such as baseball statistics. Students who were unfamiliar with the topic felt that adding the task of understanding unfamiliar processes and equations made the Parsons Problem unnecessarily difficult. This is consistent with previous observations as to how novices solve problems [11].

These results are somewhat expected. Educators who have implemented Parsons Problems in other contexts have responded that students often react negatively to them, despite the proven learning gains [2]. However, identifying specific complaints about the activities used in this course will allow for development work to reduce these problem areas. Some comments, particularly those regarding activity length and complexity, were already apparent to faculty and identified as areas for improvement. Meanwhile, a complaint such as disliking the requirement to follow another designer's code is likely not a true problem and was even identified by several students as a beneficial component of the activities.

In addition to the more open-ended questions regarding their experience with the Parsons Problems, students were asked for additional quantitative responses. First, they were asked to rank the types of Parsons Problems used throughout the semester from most to least helpful. Next, they were asked whether distractors were helpful for their learning. These results are presented in Figure 5. Students did not show a strong preference for one type of activity. However, many students indicated that they did not remember many specifics of the activities, so it may be that these were not informed choices. Despite many students expressing that distractors were unhelpful and frustrating, 50% of students stated that they were either helpful or very helpful and only 28% responded negatively. This suggests that around half of students perceive the benefit of eliminating incorrect logic and syntax through extraneous code segments.

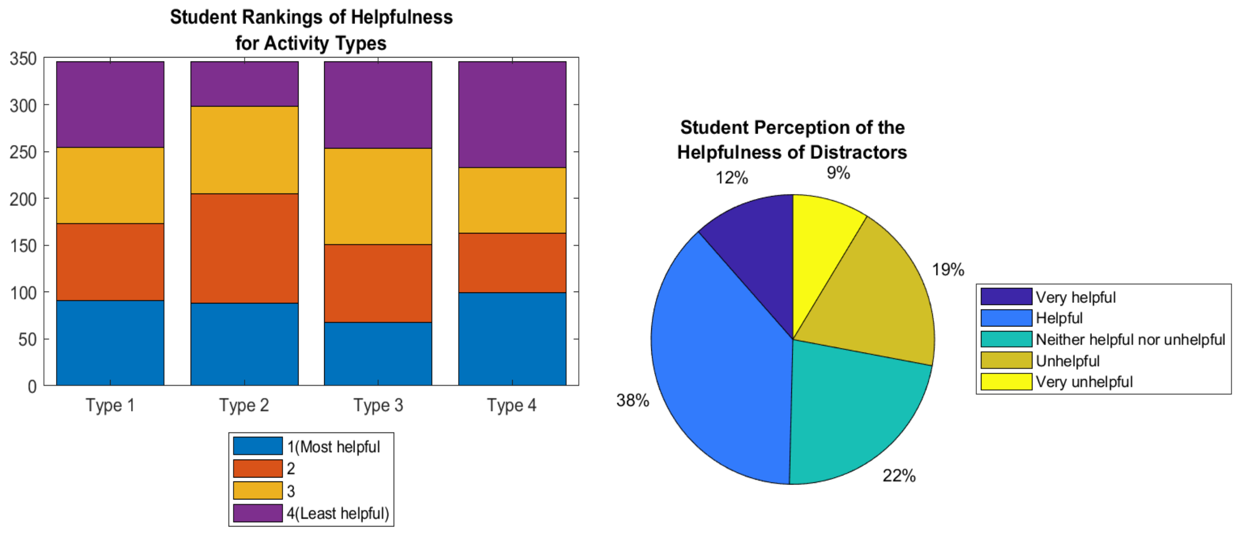


Figure 5. Students ranked their perceptions of the helpfulness of various activity types (left). They did not exhibit strong preferences for any given activity, though Type 2 problems seemed to be the most well-liked. Students also rated the helpfulness of distractors (right). Half of the students found them to be helpful or very helpful.

Findings for Future Activities

Based on student and faculty feedback, the following elements were identified as areas for improvement for future Parsons Problem activities: (1) Reduce the complexity and length of some of the hardest problems. Activities that run too long will cause students to disengage and become discouraged. (2) Create standard classroom procedure guidelines for the instructors. Consider discouraging competitive environments and ensure that there is sufficient time for support and explanation, while simultaneously expanding teaching assistant training to equip them to better engage with the students. (3) Adjust problem scenarios to be more accessible to as many students as possible and reduce unnecessary barriers to being successful in the activity.

Future Work

In addition to the modifications presented in the previous section, future activities should incorporate additional insights from literature. Many authors use software, often browser-based, to implement Parsons Problems [6],[7],[12]. As a complement to the paper-based group activities, the Parsons Problems could be used as part of the preparation activities that students complete before class. In addition to providing a more effective evaluation of the students' preparation for class, exposure to the code organizing process would assist students when faced with the more complex problems in class. Additionally, Parsons Problems have shown potential to be effective components of course examinations. [2] Exam questions using the Parsons Problems paired with similar preparation activities would allow for a more thorough examination of their efficacy in this context. Finally, researchers have observed that pairing distractors with their correct counterparts is often less frustrating for students, particularly on longer puzzles [2], [13]. Finding a way to pair the distractors may reduce the negative attitudes of some students who were overwhelmed by the problems.

Beyond the honors courses discussed above, a standard first-year engineering course instructor implemented some of these problems into their sections. However, the different course setup created some challenges in implementing them in the same way. The standard course has twice the number of students (72) and less than half the contact time per week (two 55-minute sections compared to three 125-minute sections). Additionally, the students are only taught MATLAB programming and not C/C++. These differences meant that the instructor could only devote small amounts of times to the Parsons Problems and was only able to complete a few of them. The instructor in this course was able to integrate three Parsons Problems throughout the semester, each in different ways. Only one of the problems used was the same as the honors class described above. It was one of the more difficult assignments using functions and required students to work on it one day, take a picture of their progress, and finish it on a different class day. The other problems were much simpler. One used pieces of paper and the other had the students look at the scrambled code lines on a power point and indicate the correct order. Future work could more thoroughly explore the similarities and differences in the use of Parsons Problem between these two course offerings.

In addition to improvements to the Parsons Problems, their use in both honors and standard first-year engineering classrooms should be evaluated more quantitatively and rigorously as this work progresses. Questions that will be addressed in future iterations include the following:

- How are the students approaching these problems?
- Are some instructional choices better at promoting student engagement and enjoyment?
- Is there any way to improve students' perception of the connection between these activities and coding?
- How do learning gains with paper-based activities compare to computer-based ones?
- What is an appropriate length and complexity of activity to balance varied student backgrounds and needs, providing enough challenge to engage students with some programming experience while not overwhelming novice programmers?
- How does the groupwork component affect learning gains?

Conclusion

Parsons Problems have been shown to be an effective way to provide programming instruction with a reduced cognitive load compared to writing code. However, most institutions use a computer-based tool to implement the problems and do not include a groupwork component. Additionally, the use of Parsons Problems as learning tools in the first-year engineering classroom is largely unreported. In the first-year engineering program at the Ohio State University, most faculty and 60% of students expressed that group-solved, paper-based Parsons Problems aided learning gains and engaged students in collaborative work. However, problems that took too long and were too obscure or difficult, along with inconsistent instructional approaches to the activities, caused many students to disengage and develop negative attitudes toward the problems. Additionally, the current data do not allow for rigorous analysis of whether these activities have a demonstrable effect on student learning gains or how these gains are reached. Future work in this area will attempt to address these shortcomings and continue to explore what components create a successful Parsons Problem activity of appropriate difficulty and complexity.

References

[1] D. Parsons and P. Haden, "Parson's programming puzzles: a fun and effective learning tool for first programming courses," in *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52*, 2006, pp. 157-163.

[2] P. Denny, A. Luxton-Reilly, and B. Simon, "Evaluating a new exam question: Parsons problems," in *Proceedings of the fourth international workshop on computing education research*, 2008, pp. 113-124.

[3] B. J. Ericson, L. E. Margulieux, and J. Rick, "Solving parsons problems versus fixing and writing code," in *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*, 2017, pp. 20-29.

[4] F. Paas, T. Van Gog, and J. Sweller, "Cognitive load theory: New conceptualizations, specifications, and integrated research perspectives," *Educational psychology review,* vol. 22, no. 2, pp. 115-121, 2010.

[5] B. B. Morrison, L. E. Margulieux, B. Ericson, and M. Guzdial, "Subgoals help students solve Parsons problems," in *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, 2016, pp. 42-47.

[6] J. Helminen, P. Ihantola, V. Karavirta, and S. Alaoutinen, "How do students solve parsons programming problems?--execution-based vs. line-based feedback," in *2013 Learning and Teaching in Computing and Engineering*, 2013: IEEE, pp. 55-61.

[7] P. Ihantola and V. Karavirta, "Two-dimensional parson's puzzles: The concept, tools, and first observations," *Journal of Information Technology Education,* vol. 10, no. 2, pp. 119-132, 2011.

[8] B. Morin, K. M. Kecskemety, K. A. Harper, and P. A. Clingan, "The inverted classroom in a first-year engineering course," *Proceedings of the 120th ASEE Annual Conference & Exposition*, vol. 23, p. 1, 2013.

[9] K. M. Kecskemety and B. Morin, "Student Perceptions of Inverted Classroom Benefits in a First-Year Engineering Course," *Proceedings of the 121st ASEE Annual Conference & Exposition*, vol. 24, p. 1, 2014.

[10] R. J. Freuler, M. S. Gates, J. A. Merrill, M. M. Lamont, and J. T. Demel, "An Anonymous Electronic Journal System – Program Assessment Tool and Monday Morning Quarterback," in *Proceedings of the 2002 American Society for Engineering Education Annual Conference*, June 2002.

[11] M. T. Chi, P. J. Feltovich, and R. Glaser, "Categorization and representation of physics problems by experts and novices," *Cognitive science,* vol. 5, no. 2, pp. 121-152, 1981.

[12] V. Karavirta, J. Helminen, and P. Ihantola, "A mobile learning application for parsons problems with automatic feedback," in *Proceedings of the 12th Koli Calling International Conference on Computing Education Research*, 2012, pp. 11-18.

[13] A. N. Kumar, "Helping Students Solve Parsons Puzzles Better," in *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*, 2019, pp. 65-70.