Evaluating Micro Parsons Problems as Exam Questions

Zihan Wu*
University of Michigan
Ann Arbor, MI, USA
ziwu@umich.edu

David H. Smith IV*
University of Illinois
Urbana, IL, USA
dhsmith2@illinois.edu

ABSTRACT

Parsons problems are a type of programming activity that present learners with blocks of existing code and requiring them to arrange those blocks to form a program rather than write the code from scratch. Micro Parsons problems extend this concept by having students assemble segments of code to form a single line of code rather than an entire program. Recent investigations into micro Parsons problems have primarily focused on supporting learners leaving open the question of micro Parsons efficacy as an exam item and how students perceive it when preparing for exams.

To fill this gap, we included a variety of micro Parsons problems on four exams in an introductory programming course taught in Python. We use Item Response Theory to investigate the difficulty of the micro Parsons problems as well as the ability of the questions to differentiate between high and low ability students. We then compare these results to results for related questions where students are asked to write a single line of code from scratch. Finally, we conduct a thematic analysis of the survey responses to investigate how students' perceptions of micro Parsons both when practicing for exams and as they appear on exams.

CCS CONCEPTS

• **Social and professional topics** → **Computing education**.

KEYWORDS

Parsons Problems, CS1, Assessment, micro Parsons Problems

ACM Reference Format:
Zihan Wu and David H. Smith IV. 2024. Evaluating Micro Parsons Problems as Exam Questions. In *Proceedings of the 2024 Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2024), July 8–10, 2024, Milan, Italy.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3649217.3653583

1 INTRODUCTION

Learning to program is difficult for beginners. Researchers have been working to design pedagogical activities for learners that are engaging and welcoming for novices. To improve student engagement, present models for good practice, and provide immediate feedback, Parsons and Haden [21] designed Parsons problems as an alternative type of practice. Instead of asking learners to write code from scratch, Parsons problems provide mixed-up code blocks and require learners to identify the blocks needed in the correct solution and rearrange them into the correct order. As a type of completion problem, Parsons problems make programming practice less challenging for beginners by reducing the problem space.

In traditional Parsons problems, each code block contains at least one line of code. Micro Parsons problems extended this concept by focusing on a single line of code. It provides small code fragments for learners and asks them to form a code statement in one line. Prior work on micro Parsons problems has studied its effect on learners as practice questions, and found that it reduced the time needed for learners to complete the problems, encouraged more learners to complete optional practice tasks, and resulted in comparable learning gain as writing code from scratch [31].

For traditional Parsons problems, Denny et al. [6] evaluated their effectiveness as exam items, and found that students' performances on these problems are highly correlated with code writing tasks. As a fine-grained variation of Parsons problems, micro Parsons problems also have the potential to be used as exam items. However, existing literature on micro Parsons problems focuses on their benefits for learners, leaving an unaddressed question concerning their effectiveness as exam items.

This paper aims to bridge this gap by conducting an empirical investigation of using micro Parsons problems within the context of exams and exam preparation materials in an introductory programming course. We investigate the following research questions:

RQ1: What are the psychometric properties (e.g., item difficulty and discrimination) of micro Parsons problems and how do these compare to those of single line code writing problems?
RQ2: What are students perceptions and preferences when comparing micro Parsons problems to single line code writing problems for the purposes of studying and when they appear on summative assessments?

2 BACKGROUND

2.1 Parsons and Micro Parsons Problems

Parsons problems, as introduced by Parsons and Haden [21], are a programming activity typically used in introductory courses. They provide a scaffolded environment to construct programs, presenting students with blocks of code that they must rearrange to form a solution rather than having them write that code from scratch. This problem type has shown a variety of benefits for learners such as improving learning efficiency [9], improving engagement [10, 18, 21], reducing cognitive load [3, 12, 19], improving the motivation of students [16]. Researchers have designed many variants of Parsons problems and investigated students' perceptions [6, 17, 20].

*Both authors contributed equally to this research.

ITiCSE 2024, July 8–10, 2024, Milan, Italy

ACM ISBN 979-8-4007-0600-4/24/07
https://doi.org/10.1145/3649217.3653583

A recently introduced variant of Parsons problems is micro Parsons problems [29, 30]. Whereas in traditional Parsons problems students are asked to vertically arrange blocks that each contain at least one line of code, micro Parsons problems require students to arrange smaller code fragments horizontally to form a single line of code. The motivation for this variation is to transfer the positive impacts that have been found with respect to traditional Parsons problems to other domains where the construction of a single line of code or expression is the core learning objective (e.g., regex, SQL). Much like prior investigations into traditional Parsons problems, initial investigations into micro Parsons have also focused on the impacts of utilizing them as a tool for learning [31]. Given the recency of this variety of Parsons problem, additional investigations to fully explore the effectiveness of this problem type in both formative and summative contexts are warranted.

2.2 Parsons Problems as Exam Items

Though both the original purpose of Parsons problems and much of the research that has followed has focused on their use as a tool for learning [8], several studies have investigated their utility as an exam item. Denny et al. [6] provided the first of these investigations. They identified that students' performance in Parsons problems was highly correlated with performance on code writing tasks, suggesting both items measure a similar skill. Similarly, Poulsen et al. [23] found that proof writing questions presented in a format similar to Parsons Problems, termed "Proof Blocks", still provided ample information on students' proof writing abilities.

Denny et al. [6] make additional recommendations on the design of Parsons problems for exams suggesting that: (1) incorrect blocks of code (termed distractors) should be visually grouped with their correct alternatives to minimize cognitive load and (2) the inclusion of syntax that indicates structure (e.g., curly braces, indentation) can be used to provide hints on the correct response. With respect to the former design consideration involving the use of distractors, more recent work has indicated that the presence of distractors increasing the difficulty of the item compared to not having distractors. However, their presence was shown to have a minimal impact on Parsons problems quality as an exam item at the cost of students spending substantially more time on the problem [24–27]. Similar investigations informing the design and utility of micro Parsons as an exam item have yet to be performed.

2.3 Measurement Theory

Central to evaluating a question type as an exam item is selecting a tool set for performing that evaluation. Additionally, the goal of the assessment must be identified and in order to determine how its items can be improved to meet that goal. There are multiple ways to evaluate the type of questions. In the case of the study by Denny et al. [6], one of the goals was to identify if Parsons problems evaluated a skill similar to code writing. Other criteria outside of this work include the difficulty of the item, and the ability of the question to delineate between students who are adept at a given skill and those who are not. By improving these characteristics of the items that make up an exam we can improve the measurement ability of the exam as a whole [7].

A common approach for examining the characteristics of an item is Item Response Theory (IRT). IRT models the probability of a student of a given ability (θ) to respond to a problem with a variety of estimated parameters that characterize a given item. There are a variety of models in the IRT family, each characterized by the number of parameters it includes. For example, the 2 Parameter Logistic Model (2PL)

$$P_i(\theta) = \frac{1}{1 + e^{-(a_i \cdot (\theta - b_i))}}$$

includes parameters for estimating item discrimination (a) and difficulty (b) [4, 5]. Difficulty is characterized by the ability level at which a student has a 50% chance of responding correctly to a question and discrimination how sharply an item distinguishes between students at a that given difficulty level. The 3PL model adds a third parameter generally referred to as "pseudo-guessing" which estimates a lower asymptotic bound and is interpreted as the probability of a low-ability student guessing the correct solution [4]. The 4PL model adds a final fourth parameter for estimating an upper asymptotic bound which is often referred to as "carelessness" or "slip" and is interpreted as the probability of an otherwise high ability student responding incorrectly to the question [2].

3 METHODS

3.1 Course Context and Data Collection

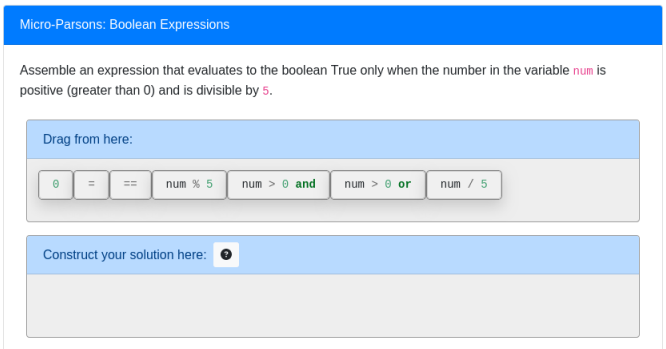
To evaluate our research questions, we designed a variety of micro Parsons problems and deployed them on exams in a large introductory Python course. The course aims at undergraduate students with non-tech majors such as business majors, and covers basic Python topics including Python Collections, loops, and basic classes and objects. Students are given four proctored exams throughout the semester along with a practice exam generator released a week before each exam. These exam generators allow students to continuously generate practice exams, which draw questions from a bank of questions defined by the instructor.

The number of students sitting each exam varied throughout the semester due to attrition, with 437 taking the first exam and 418 taking the final. All exams are computer-based and delivered via an open-source assessment platform called PrairieLearn [28]. This platform enables students to receive immediate feedback on the correctness of a submission and facilitates partial credit by allowing students to make multiple attempts on a question for a reduced number of points for each subsequent attempt.

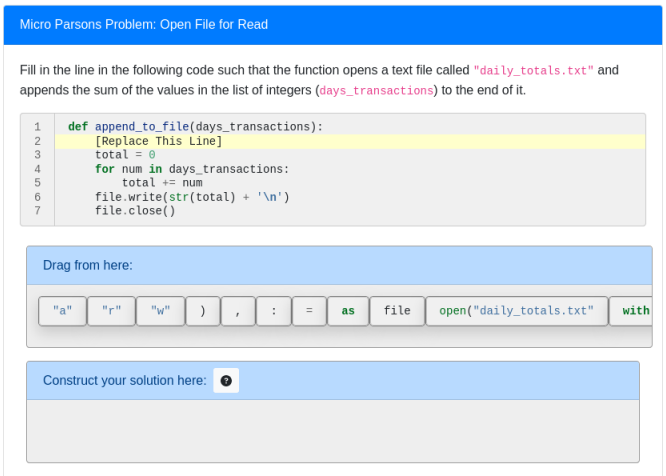
Beyond micro Parsons problems, these exams contain the following other question types: 1) code writing question, 2) traditional parsons problems, 3) single-line code writing problems (Figure 2), 3) code fixing problems, 4) short answer code comprehension questions, and 5) code tracing questions. Collection of this data, as well as all other data collected in this course, was approved by the institutional ethics review board at the institution where the study was conducted.

3.2 Problem Design

Two variations of micro Parsons problems were deployed in the course. In the first two exams the included problems asked students to arrange the given blocks to form a single, independent line of



(a) Standard micro Parsons - The first variation of micro Parsons problems as used on exams one and two.



(b) Fill-in-the-Blank micro Parsons - The second variation of Parsons problems as used on exams three and four.

Figure 1: The two variations of Parsons problems used in the course.

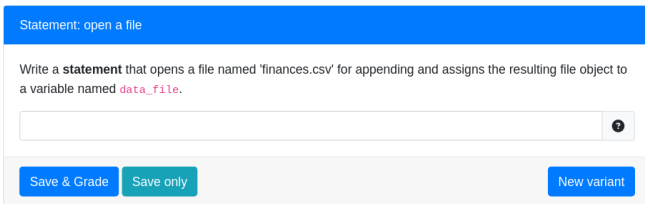


Figure 2: An example of a single-line programming question

code that accomplished a given task (Figure 1a). In the remaining two exams, students were given an existing segment of code with a single line of code removed. For these problems, students were asked to arrange blocks to "fill in the blank" such that the segment of code as a whole accomplished its given task (Figure 1b). In both cases, the micro Parsons problems included jumbled distractors reduce the possibility of guessing the correct solution. Additionally, students were given partial credit for incorrect attempts based on the number of blocks they positioned correctly.

3.3 Student Survey

In addressing **RQ2**, students were given an end-of-course survey on their perceptions of the micro Parsons problems and how they compared to single-line programming questions. First, students were asked to provide their general thoughts in comparing the two question types in short answer format. Following that, students were asked two multiple choice asking which question type they preferred where the options were: (1) Micro Parsons, (2) Single Line Code Writing, and (3) No Preference. The questions are as follows:

- **Q1:** Which question type do you prefer to appear on exams?
- **Q2:** Which question type do you prefer to appear on study materials (e.g., practice exams)?

In association with each multiple choice question was a short answer question which asked students to expand on their preference.

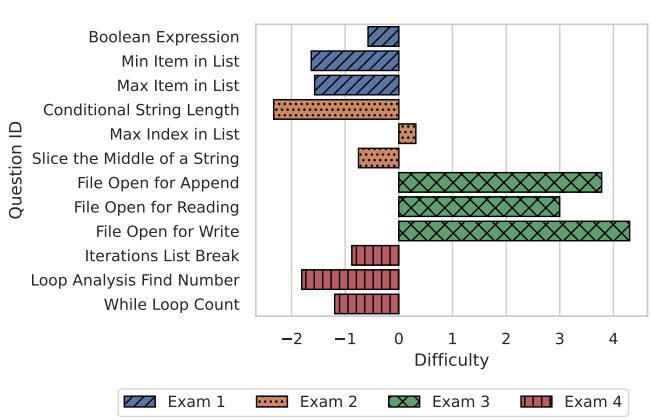
4 RQ1: DIFFICULTY AND DISCRIMINATION

To address **RQ1** the scores students received on questions were dichotomized based on the success of a student's first attempt. This is done to enable analysis with a 2PL IRT model, which requires the use of dichotomized scores. The reason for fitting the model based on the correctness of a student's first attempt rather than their best attempt is, given students can attempt a question multiple times, many questions were ultimately answered correctly by all or a large majority of students. This would prevent those questions from being included when fitting the model, as 2PL IRT does not support the inclusion of questions with one response category (i.e., all correct or all incorrect). We fit a 2PL IRT model for each exam to determine the item difficulty and item discrimination statistics for the items that appear on those exams. To aid in the interpretation of the item-discrimination statistics, figures use the thresholds presented by Baker [1] of: Very Low ($\leq$ 0.34), Low (0.35 – 0.64), Moderate (0.65 – 1.34), High (1.35 – 1.69), and Very High ($>$ 1.7).

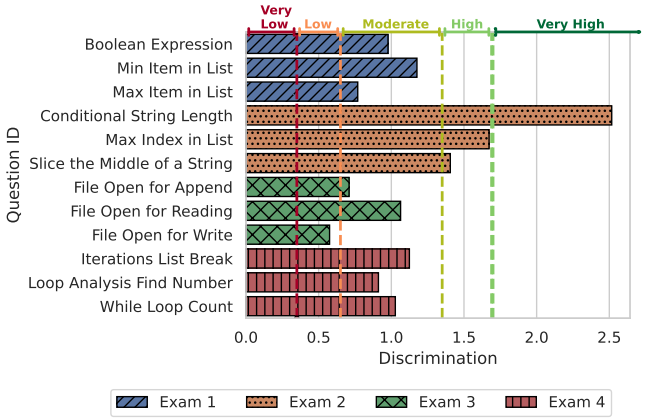
For the comparison between single-line code writing problems and micro Parsons problems, we selected a total of 17 single-line code writing questions that appeared alongside the micro Parsons problems on exams. These questions cover related topics (e.g., string slicing, boolean expressions, file opening) but are not identical in format to the micro Parsons. As such, their results are included to help contextualize the item statistics for micro Parsons – given they are a related item testing similar concepts – rather than provide a one-to-one comparison.

4.1 Results

Overall, the micro Parsons problems appeared to generally have low difficulty (Figure 3a) coefficients with moderate discrimination (Figure 3b). These results indicate that the problems, in general, appear to do a reasonable job of differentiating between students is the lower ability range. Comparing the distribution of difficulty



(a) Item difficulty statistics from the 2PL model for each of the micro Parsons included on exams. Coefficients are interpreted as the ability level at which a student has a 50% chance of answering correctly.



(b) Item discrimination statistics from 2PL model for each of the micro Parsons included on exams. The thresholds for item discrimination (e.g., how well an item distinguishes between students above and below a given ability level) from Baker [1] are presented as well.

Figure 3: The Item-Difficulty and Item-Discrimination statistics for the items used in this study.

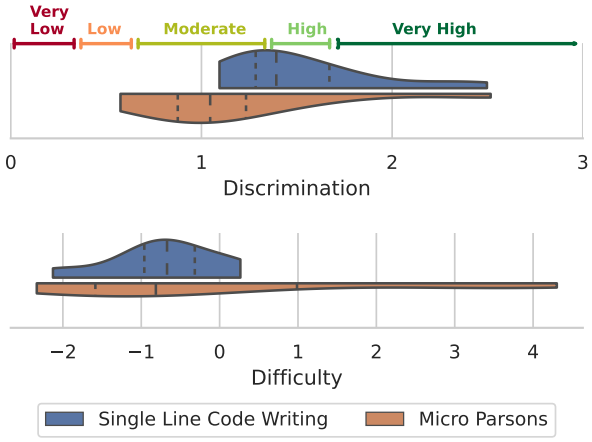


Figure 4: Comparison of the distributions of item-difficulty and item-discrimination statistics for micro Parsons and single-line code writing questions

and discrimination statistics for micro Parsons and single-line programming questions we find that both appear to have similar distributions of difficulty but single-line code writing problems have higher item-discrimination (Figure 4). This indicates that on average the item functions for students of lower ability and single-line code writing questions provide a more sharp delineation between students at that ability level.

In looking at Figure 3a, the one notable exception to this trend are the questions relating to opening a file which appeared on exam three. These questions had exceptionally high difficulty compared to micro Parsons which appeared on other exams. These questions differed from their counterparts in that the student was required to read the surrounding code and, based on the indentation, determine whether to open the file using only the `open()` function or using `with-as`. In looking at the incorrect submissions we find that 34% of all incorrect submissions attempted to use `with-as`. This highlights a potential connection exists between the distractors that are chosen in micro Parsons and the provided code for which the students are intended to fill in the blanks.

5 RQ2: STUDENT PERCEPTIONS AND PREFERENCES

In total, we collected survey responses from 390 students. Two researchers independently coded the first 50 responses to develop an initial code book. Upon iterating and finalizing the code book on the first 50 responses, the researchers independently coded the next 50 to establish an interrater agreement. The inter-rater agreement, Krippendorf's alpha, was $\alpha = 0.7$, which was sufficient for preliminary conclusion [15]. One of the researchers coded the remaining 290 responses according to the established code book.

5.1 Results

We collected anonymous responses from 390 students asking which question types they preferred, both in the context of exams and practice. When asked about the preferred type of question to appear on the exam, 163 (41%) students chose micro Parsons problems, 138 (35%) chose single-line code writing questions, and 89 (23%) students chose no preference. For practice, however, the most popular response was no preference (n = 160, 41%), while 134 (34%) students preferred single-line code writing questions, and 96 (25%) students preferred micro Parsons problems.

Several themes emerged from students' responses in explaining their choices and comparing the two types of problems.

5.1.1 Perceived Difficulty. When comparing the two types of problems, 83 students (21%) explicitly noted that they thought micro Parsons problems were easier. Among them, 13 students felt that micro Parsons problems **provided hints** compared to the single-line code writing question, especially when they did not know where

to start: *"micro parsons are good to give you a head start on the topic and help you out a bit, [single-line code writing] questions kind of mess you up if you don't know where to start." (P308)* Fifteen students felt that micro Parsons problems were easier in exams because they provided **partial credit** for incorrect responses.

Contrary to prior within-subjects think-aloud studies on Parsons problems where all participants find them easier than writing code from scratch [13], in our survey responses, 50 students (13%) felt that micro Parsons problems are more difficult than single-line code writing problems. *"I found that the micro Parsons were slightly more difficult as I had to try to think through the code with what was given instead of trying to think it through on my own." (P23)* In our study, only micro Parsons problems contained the fill-in-the-blank type of problems, which can also contribute to the perceived difficulty.

5.1.2 Assisting Learning. Most students reported that they found both types of problems beneficial for learning. Meanwhile, 36 students (9%) specifically pointed out that micro Parsons are helpful for learning, because they **struggle less** and **can practice specific parts of a single line**. They felt that micro Parsons problems **provide examples** of how the code lines should look like, **provide better feedback** on which part of the solution was incorrect, and **require less memorization**: "*it is more in line with how actual coding is: you do not have to memorize a lot of stuff, you should know the concept and the ways to look for the information.*" (P98) On the other hand, 32 (8%) students explicitly mentioned that the single-line code-writing questions helped them learn. They felt these questions involve **"actual coding"** and **shows what they actually know**. They felt that having less help could force them **memorize better** or apply more **high-level thinking**.

Interestingly, learners' comments on which variation they think better helps learning do not always align with their selected preferences for homework or exams. Many students explicitly said they preferred the practice exams to *"have the same format of questions" that would appear on the exam. (P159)* For the exam questions, students chose micro Parsons problems because they provide extra credits when the answers are incorrect.

5.1.3 Confusing Distractors. A total of 40 students (10%) mentioned that micro Parsons problems could be confusing at times. Specifically, 10 students offered complaints about the distractors in the micro Parsons problems. Some suggested removing all distractors, or providing a "use all blocks" prompt for those problems that do not contain distractors: *"Usually with the micro Parsons problems I tend to overthink my answer and end up getting the question wrong. All the extra blocks are confusing." (P215)* Most of the time, the distractors contain minor problems that novices tend to make, such as the use of square brackets instead of parentheses, or ignoring quotation marks around string literals, and minor logic errors (e.g., `and` vs `or`). Four students explicitly expressed their annoyance with these distractors: *"it is extremely annoying when micro parsons problems include wrong blocks because it only feels like they are trying to trip me up based on reading errors." (P292)*

6 DISCUSSION

From our results we highlight and expand on three takeaways: 1) the ability of micro Parsons problems to provide partial credit and simplified feedback; 2) commentary on how distractor selection can in some cases significantly impact item difficulty; 3) the expertise reversal effect and its potential impact on the design of micro Parsons; and 4) the relationship between the presence of distractors and confusion students encountered on micro Parsons problems. Each of these serves to inform the design and use of micro Parsons as an item used on exams and practice materials alike.

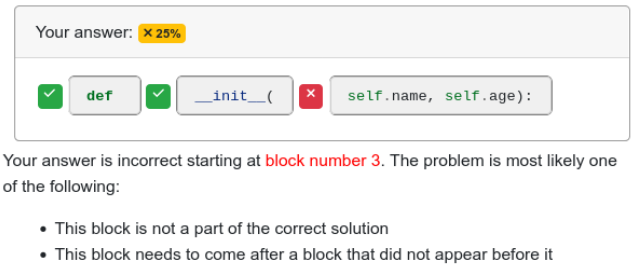


Figure 5: The feedback and partial credit mechanism used for our implementation of micro Parsons.

6.1 Feedback and Partial Credit with Micro Parsons Problems

As noted by students in the qualitative results, one of the key differences between micro Parsons problems and single-line code writing problems is the limited solution space enforced by requiring students to arrange predefined blocks. As noted by Denny et al. [6] – though in the context of traditional Parsons problems – this enables more straightforward scoring for instructors. However, a notable difference between the micro Parsons used in this study and those used in the investigations of Denny et al. [6] is our study uses an online platform to deliver the questions whereas their used a paper based exam. The ability to receive instantaneous feedback and make multiple attempts allows forwards these benefits onto students as well both during the exam and while practicing.

As noted by many students, the use of single-line code writing problems and micro Parsons alike was well received as many students viewed as questions as testing them on and allowing them to practice elements of larger programs in isolation. Though it may be the case that micro Parsons problems have less discriminating power relative to single-line code writing questions there are other considerations that instructors may have beyond these metrics. By reducing the error space and providing feedback that is easier to interpret for this class of programming problem it is likely that the students' experience is improved. This is particularly a consideration in the context of practice exams, such as the ones our students encountered, as it scaffolds their studying process.

6.2 Distractor Selection for Fill-in-the-Blank

As discussed in the quantitative results (Section 4), the fill-in-the-blank micro Parsons (see fig. 1b) had much higher item-difficulties any other micro Parsons problem or single line code writing question. In these questions, students were presented with all the blocks needed to open a file either using `with-as` or simply storing the result of `open()` in a variable. The intention was for students to infer from the code they were completing that `with-as` was not appropriate given it requires any manipulation of the file to be

done underneath it in an indented block. This problem highlights a unique connection that can be made between the distractors chosen and the context of the code the student is required to complete in micro Parsons problems for languages where whitespace is semantic.

6.3 Potential Expertise Reversal Effect

From the student survey, we discovered that some students perceived micro Parsons problems as more difficult than writing code from scratch. This finding differs from prior work that included within-subject studies for students to compare micro Parsons problems or regular Parsons problems with writing code from scratch [13, 31]. One potential explanation is the expertise reversal effect. It refers to the situation where some instructional methods designed for novices become inefficient or have negative consequences for advanced learners [14]. According to cognitive load theory, the instructional methods for novices usually contain extra information and guidance. However, expert learners already possess schema-based knowledge, which provides internal guidance. Expert learners need to resolve and reconcile the two different sets of guidance, which can introduce extra burdens in working memory.

As pointed out by some students in the qualitative analysis, constructing solutions only by given blocks can sometimes be even more difficult when they already have a plain-code answer in mind. The hints, in this case, the blocks, can become extra guidance that is not needed by expert learners and cause a burden for them. This is particularly a consideration when selecting distractor blocks as the inclusion of plausible alternatives – and thus more blocks – may increase the prevalence and impact of this effect. This may be considered a desirable difficulty in practice materials for illustrating alternative solutions. However, in summative assessments this added difficulty may stem from the question format rather than the content being tested. This finding highlights the careful consideration that should be taken when selecting the number of blocks, and distractors, that are included in a solution space.

6.4 Confusing Distractors

In our qualitative results, we discovered that some students found micro Parsons problems confusing. The major complaint is the distractors. Only a few students elaborated on their confusion, explaining that they felt that the distractors were too tricky. There are two interpretations behind students' perception of the trickiness of the distractors. First, students felt that sometimes it was too difficult to identify the correct blocks versus the distractors; It is also possible that the complaints reflected that they deem the difficulty unnecessary.

Although distractors received complaints from some students, prior work in the context of traditional Parsons problems suggests they may be useful for learning. Prior work on proof blocks, a type of Parsons problem for mathematical proofs has shown that learners who completed proof blocks with distractors performed better on the post-test than those who completed proof blocks without distractors, though the difference was not statistically significant [22]. In a multi-institutional multi-national study, learners who practiced with distractors were also found to have better performances on fixing code that contains similar errors in the distractors than those who did not [11]. Considering the importance of debugging and fixing code, especially with the development of generative AI tools, it is likely distractors will become increasingly important for learning.

Meanwhile, the visual presentation of the distractors can also affect students' perceptions. Similar to previous research on micro Parsons problems, we only used jumbled distractors in this study. Because micro Parsons problems require learners to rearrange code in one line, the design of interfaces for visual grouping distractors with their correct alternatives is not as straightforward. In our study, no student provided concrete examples of distractors that they found confusing. Future research should investigate the attributes that make distractors confusing for learners, and examine the effective ways to add distractors that can reduce confusion while maintaining item difficulty and item discrimination.

7 LIMITATIONS AND FUTURE WORK

There are several limitations to the results of our study that can affect its generalizability. First, only a small number of micro Parsons were used throughout the semester. More questions and question versions would need to be created to provide a more complete and accurate picture of how micro Parsons function as an exam item and how they compare to related text-entry code writing questions.

Second, a limitation of the quantitative analysis of this work is that the comparison of item-discrimination statistics between micro Parsons problems and single-line code writing problems was at a categorical level. That is, we did not create pairs of questions with identical solutions and compare them. As such, our results comparing the item discrimination and difficulty distributions are more suited to contextualize the results for micro Parsons using a related code writing exercise.

A final limitation is students had access to practice exams which contained both the single line code writing problem generators used on exams and micro Parsons that covered related topics to those on their exams. This likely had the impact of lowering the difficulty of the items compared to if fully novel questions were used on the exams.

8 CONCLUSION

This study employed micro Parsons problems as exam items across four exams, and adopted measurement theory to analyze its effectiveness. We calculated the item difficulty and item discrimination of micro Parsons problems as well as traditional single-line code writing problems across the semester. In this study, we found that micro Parsons questions are comparable in difficulty but slightly lower in discrimination compared to writing code from scratch. We also found insights that are worth future exploration from our student survey. Students displayed diverse preferences when it came to micro Parsons problems and single-line code questions in exam, largely based on their perceived potential to earn more points. We found a potential expertise reversal effect from student response that considered micro Parsons problems harder than writing code from scratch, which motivates further inquiries into the ideal design for micro Parsons problems used in exams.

REFERENCES

[1] Frank B Baker. 2001. *The basics of item response theory.* ERIC.

[2] Mark A Barton and Frederic M Lord. 1981. An upper asymptote for the three-parameter logistic item-response model. *ETS Research Report Series* 1981, 1 (1981), i–8.

[3] Jeff Bender, Bingpu Zhao, Alex Dziena, and Gail Kaiser. 2023. Integrating Parsons puzzles within Scratch enables efficient computational thinking learning. *Research and Practice in Technology Enhanced Learning* 18 (2023), 022–022.

[4] Allan Birnbaum. 1968. Some latent trait models and their use in inferring an examinee's ability. *Statistical theories of mental test scores* (1968).

[5] André F De Champlain. 2010. A primer on classical test theory and item response theory for assessments in medical education. *Medical education* 44, 1 (2010), 109–117.

[6] Paul Denny, Andrew Luxton-Reilly, and Beth Simon. 2008. Evaluating a new exam question: Parsons problems. In *Proceedings of the Fourth International Workshop on Computing Education Research* (Sydney, Australia) *(ICER '08)*. Association for Computing Machinery, New York, NY, USA, 113–124. https://doi.org/10.1145/1404520.1404532

[7] Robert L Ebel and David A Frisbie. 1972. *Essentials of educational measurement.*

[8] Barbara J. Ericson, Paul Denny, James Prather, Rodrigo Duran, Arto Hellas, Juho Leinonen, Craig S. Miller, Briana B. Morrison, Janice L. Pearce, and Susan H. Rodger. 2022. Parsons Problems and Beyond: Systematic Literature Review and Empirical Study Designs. In *Proceedings of the 2022 Working Group Reports on Innovation and Technology in Computer Science Education* (<conf-loc>, <city>Dublin</city>, <country>Ireland</country>, </conf-loc>) *(ITiCSE-WGR '22)*. Association for Computing Machinery, New York, NY, USA, 191–234. https://doi.org/10.1145/3571785.3574127

[9] Barbara J. Ericson, James D. Foley, and Jochen Rick. 2018. Evaluating the Efficiency and Effectiveness of Adaptive Parsons Problems. In *Proceedings of the 2018 ACM Conference on International Computing Education Research* (Espoo, Finland) *(ICER '18)*. Association for Computing Machinery, New York, NY, USA, 60–68. https://doi.org/10.1145/3230977.3231000

[10] Barbara J. Ericson, Mark J. Guzdial, and Briana B. Morrison. 2015. Analysis of Interactive Features Designed to Enhance Learning in an Ebook. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research* (Omaha, Nebraska, USA) *(ICER '15)*. Association for Computing Machinery, New York, NY, USA, 169–178. https://doi.org/10.1145/2787622.2787731

[11] Barbara J. Ericson, Janice L. Pearce, Susan H. Rodger, Andrew Csizmadia, Rita Garcia, Francisco J. Gutierrez, Konstantinos Liaskos, Aadarsh Padiyath, Michael James Scott, David H. Smith, Jayakrishnan M. Warriem, and Angela Zavaleta Bernuy. 2023. Multi-Institutional Multi-National Studies of Parsons Problems. In *Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education*. Association for Computing Machinery, New York, NY, USA, 57–107. https://doi.org/10.1145/3623762.3633498

[12] Geela Venise Firmalo Fabic, Antonija Mitrovic, and Kourosh Neshatian. 2019. Evaluation of Parsons problems with menu-based self-explanation prompts in a mobile python tutor. *International Journal of Artificial Intelligence in Education* 29 (2019), 507–535.

[13] Carl C. Haynes and Barbara J. Ericson. 2021. Problem-Solving Efficiency and Cognitive Load for Adaptive Parsons Problems vs. Writing the Equivalent Code. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (<conf-loc>, <city>Yokohama</city>, <country>Japan</country>, </conf-loc>) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 60, 15 pages. https://doi.org/10.1145/3411764.3445292

[14] Slava Kalyuga. 2009. The expertise reversal effect. In *Managing cognitive load in adaptive multimedia learning*. IGI Global, 58–80.

[15] Klaus Krippendorff. 2018. *Content analysis: An introduction to its methodology*. Sage publications.

[16] Amruth N. Kumar. 2017. The Effect of Providing Motivational Support in Parsons Puzzle Tutors. In *Artificial Intelligence in Education*, Elisabeth André, Ryan Baker, Xiangen Hu, Ma. Mercedes T. Rodrigo, and Benedict du Boulay (Eds.). Springer International Publishing, Cham, 528–531.

[17] Brooke Morin and Krista M Kecskemety. 2021. Collaborative Parsons Problems in a Remote-learning First-year Engineering Classroom. In *2021 ASEE Virtual Annual Conference Content Access.*

[18] Brooke C Morin, Krista M Kecskemety, Kathleen A Harper, and Paul Alan Clingan. 2020. Work in Progress: Parsons Problems as a Tool in the First-Year Engineering Classroom. In *2020 ASEE Virtual Annual Conference Content Access.*

[19] Briana B. Morrison, Lauren E. Margulieux, Barbara Ericson, and Mark Guzdial. 2016. Subgoals Help Students Solve Parsons Problems. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (Memphis, Tennessee, USA) *(SIGCSE '16)*. Association for Computing Machinery, New York, NY, USA, 42–47. https://doi.org/10.1145/2839509.2844617

[20] Solomon Sunday Oyelere, Friday Joseph Agbo, Ismaila Temitayo Sanusi, Abdullahi Abubakar Yunusa, and Kissinger Sunday. 2019. Impact of puzzle-based learning technique for programming education in Nigeria context. In *2019 IEEE 19th International Conference on Advanced Learning Technologies (ICALT)*, Vol. 2161. IEEE, 239–241.

[21] Dale Parsons and Patricia Haden. 2006. 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*. 157–163.

[22] Seth Poulsen, Hongxuan Chen, Yael Gertner, Benjamin Cosman, Matthew West, and Geoffrey L Herman. 2023. Measuring the Impact of Distractors on Student Learning Gains while Using Proof Blocks. *arXiv preprint arXiv:2311.00792* (2023).

[23] Seth Poulsen, Mahesh Viswanathan, Geoffrey L Herman, and Matthew West. 2022. Evaluating proof blocks problems as exam questions. *ACM Inroads* 13, 1 (2022), 41–51.

[24] David H. Smith, Max Fowler, and Craig Zilles. 2023. Investigating the Role and Impact of Distractors on Parsons Problems in CS1 Assessments. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1* (<conf-loc>, <city>Turku</city>, <country>Finland</country>, </conf-loc>) *(ITiCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 417–423. https://doi.org/10.1145/3587102.3588819

[25] David H. Smith and Craig Zilles. 2023. Discovering, Autogenerating, and Evaluating Distractors for Python Parsons Problems in CS1. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (<conf-loc>, <city>Toronto ON</city>, <country>Canada</country>, </conf-loc>) *(SIGCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 924–930. https://doi.org/10.1145/3545945.3569801

[26] David H. Smith, IV, Seth Poulsen, Max Fowler, and Craig Zilles. 2023. Comparing the Impacts of Visually Grouped and Jumbled Distractors on Parsons Problems in CS1 Assessments. In *Proceedings of the ACM Conference on Global Computing Education Vol 1* (<conf-loc>, <city>Hyderabad</city>, <country>India</country>, </conf-loc>) *(CompEd 2023)*. Association for Computing Machinery, New York, NY, USA, 154–160. https://doi.org/10.1145/3576882.3617927

[27] David H. Smith IV. 2023. Useful Distractions? Investigating the Utility of Distractors in Parsons Problems. In *Proceedings of the 2023 ACM Conference on International Computing Education Research - Volume 2* (Chicago, IL, USA) *(ICER '23)*. Association for Computing Machinery, New York, NY, USA, 62–63. https://doi.org/10.1145/3568812.3603463

[28] Matthew West, Geoffrey L Herman, and Craig Zilles. 2015. Prairielearn: Mastery-based online problem solving with adaptive scoring and recommendations driven by machine learning. In *2015 ASEE Annual Conference & Exposition*. 26–1238.

[29] Zihan Wu. 2023. Investigating the Effectiveness of Variations of Micro Parsons Problems. In *Proceedings of the 2023 ACM Conference on International Computing Education Research - Volume 2* (Chicago, IL, USA) *(ICER '23)*. Association for Computing Machinery, New York, NY, USA, 120–122. https://doi.org/10.1145/3568812.3603447

[30] Zihan Wu, Barbara Ericson, and Christopher Brooks. 2021. Regex Parsons: Using Horizontal Parsons Problems to Scaffold Learning Regex. In *Proceedings of the 21st Koli Calling International Conference on Computing Education Research* (Joensuu, Finland) *(Koli Calling '21)*. Association for Computing Machinery, New York, NY, USA, Article 31, 3 pages. https://doi.org/10.1145/3488042.3489968

[31] Zihan Wu, Barbara J. Ericson, and Christopher Brooks. 2023. Using Micro Parsons Problems to Scaffold the Learning of Regular Expressions. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1* (<conf-loc>, <city>Turku</city>, <country>Finland</country>, </conf-loc>) *(ITiCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 457–463. https://doi.org/10.1145/3587102.3588853