

The Journal of Computing Sciences in Colleges

**Papers of the 19th Annual CCSC
Southwestern Conference**

March 27th-28th, 2026
University of California, Riverside
Riverside, CA

Bin Peng, Associate Editor
Park University

Megan Thomas, Regional Editor
California State University, Stanislaus

Volume 41, Number 9

April 2026

The Journal of Computing Sciences in Colleges (ISSN 1937-4771 print, 1937-4763 digital) is published at least six times per year and constitutes the refereed papers of regional conferences sponsored by the Consortium for Computing Sciences in Colleges.

Copyright ©2026 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

Table of Contents

The Consortium for Computing Sciences in Colleges Board of Directors	5
CCSC National Partners	7
Welcome to the 2026 CCSC Southwestern Conference	8
Regional Committees — 2026 CCSC Southwestern Region	9
Exploring Students’ Self-Reported Collaboration on Pair Programming Assignments	10
<i>Sophia Krause-Levy, Wanda Young, Jenna Therese Tumlos and Jennifer K. Olsen, University of San Diego</i>	
Putting “Active” into Learning Theory of Computation	29
<i>Tammy VanDeGrift, University of Portland</i>	
Exploring Data Mining Algorithms and Ethics Via a News Journal	48
<i>Kevin Treu, Furman University</i>	
Student Scheduling Patterns and Associated Outcomes for Computer Science Courses in a High Utilization Testing Facility	59
<i>Kelly Downey and Paea LePendu, University of California, Riverside</i>	
Enhancing Engineering Design Thinking and Transferable Skills Through Personalized TinkerCAD Projects	73
<i>SJ Kim, Emma Regentova, and Venki Muthukumar, University of Nevada Las Vegas</i>	
Evaluating Large Language Models on Solved and Unsolved Problems in Graph Theory: Implications for Computing Education	83
<i>Adithya Kulkarni, Ball State University; Mohna Chakraborty, Jio Institute; Jay Bagga, Ball State University</i>	
Exploring the Impact of Slide vs. Poster Presentations on Student Engagement in a Data Analytics	101
<i>Julia Olivieri, University of the Pacific</i>	
Reviewers — 2026 CCSC Southwestern Conference	114

The Consortium for Computing Sciences in Colleges

Board of Directors

Following is a listing of the contact information for the members of the Board of Directors and the Officers of the Consortium for Computing Sciences in Colleges (along with the years of expiration of their terms), as well as members serving CCSC:

Bryan Dixon, President (2026), bcdixon@csuchico.edu, Computer Science Department, California State University Chico, Chico, CA 95929.

Shereen Khoja, Vice President/President-Elect (2026), shereen@pacificu.edu, Computer Science, Pacific University, Forest Grove, OR 97116.

Abbas Attarwala, Publications Chair (2027), aattarwala@csuchico.edu, Department of Computer Science, California State University Chico, Chico, CA 95929.

Ed Lindoo, Treasurer (2026), elindoo@regis.edu, Anderson College of Business and Computing, Regis University, Denver, CO 80221.

Haiyan Cheng, Membership Secretary (2028), hcheng@willamette.edu, Department of Computer Science, Willamette University, Salem, OR 97301.

Judy Mullins, Central Plains Representative (2026), mullinsj@umkc.edu, University of Missouri-Kansas City, Kansas City, MO (retired).

Michael Flinn, Eastern Representative (2026), mflinn@frostburg.edu, Department of Computer Science & Information Technologies, Frostburg State University, Frostburg, MD 21532.

Gabe Ferrer, Midsouth Representative (2028), ferrer@hendrix.edu, Department

of Computer Science, Hendrix College, Conway, AR 72032.

David Largent, Midwest Representative (2026), dllargent@bsu.edu, Department of Computer Science, Ball State University, Muncie, IN 47306.

Michael Gousie, Northeastern Representative (2028), gousie_mike@wheatoncollege.edu, Computer Science Department, Wheaton College, Norton, MA 02766.

Ben Tribelhorn, Northwestern Representative (2027), tribelhb@up.edu, School of Engineering, University of Portland, Portland, OR 97203.

Mohamed Lotfy, Rocky Mountain Representative (2028), MohamedL@uvu.edu, Information Systems & Technology Department, College of Engineering & Technology, Utah Valley University, Orem, UT 84058.

Mika Morgan, South Central Representative (2027), mika.morgan@msutexas.edu, Department of Computer Science, Midwestern State University, Wichita Falls, TX 76308.

Karen Works, Southeastern Representative (2027), keworks@pc.fsu.edu, Department of Computer Science, Florida State University Panama City, Panama City, FL 32405.

Michael Shindler, Southwestern Representative (2026), mikes@uci.edu, Computer Science Department, UC Irvine, Irvine, CA 92697.

Serving the CCSC: These members are serving in positions as indicated:

Bin Peng, Associate Editor, bin.peng@park.edu, Computer Science and Information Systems, Park University, Parkville, MO 64152.

Lucas Cordova, Associate Treasurer, lpcordova@willamette.edu, Department of Computer Science, Willamette University, Salem, OR 97301.

George Dimitoglou, Comptroller, dimitoglou@hood.edu, Department of Computer Science, Hood College, Frederick, MD 21701.

Megan Thomas, Membership System

Administrator, mthomas@cs.csustan.edu, Department of Computer Science, California State University Stanislaus, Turlock, CA 95382.

Karina Assiter, National Partners Chair, KarinaAssiter@landmark.edu, Landmark College, Putney, VT 05346.

Ed Lindoo, UPE Liaison, elindoo@regis.edu, Anderson College of Business and Computing, Regis University, Denver, CO 80221.

Deborah Hwang, Webmaster, hwangdjh@acm.org.

CCSC National Partners

The Consortium is very happy to have the following as National Partners. If you have the opportunity please thank them for their support of computing in teaching institutions. As National Partners they are invited to participate in our regional conferences. Visit with their representatives there.

Platinum Level Partner

College Board

Gold Level Partner

Rephactor

ACM2Y

Blossoms

MAP-CS: Mission-Aligned Programs in Computer Science

Welcome to the 2026 CCSC Southwestern Conference

Welcome to the 2026 Southeastern Regional Conference of the Consortium for Computing Sciences in Colleges. The CCSC:SW Regional Board welcomes you to University of California, Riverside (UCR). The conference is designed to promote a productive exchange of information with respect to computer science education. It is intended for faculty as well as administrators of academic computing facilities, and it is also intended to be welcoming to student participants in a variety of special activities. We hope that you will find something to challenge and engage you at the conference!

We received 13 papers this year of which 7 were accepted to be presented at the conference and included in the proceedings – an acceptance rate of 53%. We have an exciting activity designed specifically for students – a poster session, with prizes for the top submissions.

We especially would like to thank the faculty, staff, and students of UCR for their help in organizing and publicizing this conference. Many thanks also to the CCSC Board, the CCSC:SW Regional Board, and to a wonderful Conference Committee. We also need to send our deepest appreciation to our partners, sponsors, and vendors.

We hope you enjoy the conference and your visit to UCR.

Marism Salloum and Allan Knight
University of California, Riverside
Co-Chairs, CCSC:SW 2026

2026 CCSC Southwestern Conference Committee

Conference Co-Chairs

Mariam Salloum University of California, Riverside
Allan Knight University of California, Riverside

Papers Chair

Megan Thomas California State University, Stanislaus

Speakers Chair

Michael Shindler University of California, Irvine

Posters Chair

Ashley Pang University of California, Riverside

Panels/Tutorials Chair

Olivera Grujic California State University, Stanislaus

Lightning Talk Chair

Sophia Krause-Levy University of San Diego

Partner’s Chair

Michael Shindler University of California, Irvine

Nifty Tools & Assignments Chair

Joshua Gross California State University, Monterey Bay

Regional Board — 2026 CCSC Southwestern Region

Regional Chair

Joshua Gross (acting) California State University, Monterey Bay

Regional Representative

Michael Shindler (2023) University of California, Irvine

Treasurer/Registrar

Soohyun Liao (2025) University of California Dan Diego

Past Conference Chairs

Sophia Krause-Levy University of San Diego

Sat Garcia University of San Diego

Webmaster

Megan Thomas (2021) California State University, Stanislaus

Exploring Students' Self-Reported Collaboration on Pair Programming Assignments*

*Sophia Krause-Levy, Wanda Young,
Jenna Therese Tumlos and Jennifer K. Olsen
Computer Science Department
University of San Diego
San Diego, CA 92110*

{skrauselevy, wyoung, jtumlos, jenniferolsen}@sandiego.edu

Abstract

Pair programming has been increasingly adopted in undergraduate computing courses with research showing benefits related to grades, assignment quality, and retention. However, it is not clear from previous work if these benefits are due to the specific pair program paradigm or if other factors may be responsible. To better understand the impact that pair programming may have on learning, in this work we explore how students report working together when given a pair programming assignment to complete outside of the classroom and if they actually are following the pair programming process.

We surveyed 43 undergraduate students at a small liberal arts institution on their collaboration practices during pair programming assignments, including if they worked together and if the division of work was perceived as equal. We found that 72% of the students reported dividing work on assignments, and the majority of students did not seem to be following the pair programming process. Although the majority still reported completing around half the assignment. Students did not divide the work in one standard way, but divisions included by section of the

*Copyright ©2026 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

assignment, when they had time, who had the skills, or if they had a preference. These results suggest that students are not engaging in the pair programming process as prescribed and warrants further investigation into what about the process is leading to higher student outcomes.

1 Introduction

Pair programming is the practice of two students working at a single computer to develop and test programs and follows a specific structure [41]. More recently the term pair programming has been used to refer to any cooperative set up in which there is more than one programmer responsible for the completion of the code. For example, in side-by-side programming, the programmers work separately on separate computers to complete different parts of the assignment but are able to work together as needed [29, 32]. In this paper, we are focused on the practice of students working together collaboratively, which is when "partners do the work 'together'" [11]. In this case, we are using the definition of pair programming as it was introduced in eXtreme Programming and referenced above [1]. We focus on this type of programming as it is commonly used at institutions including ours. Furthermore, the collaborative learning process requires students to articulate their mental model when sharing ideas and to assess and update their mental model based on partner contributions [7, 15], which has been shown to be beneficial to learning [20].

Pair programming has been widely adopted in undergraduate computing over the last two decades, driven by evidence of higher pass rates [24, 25, 26, 28] and improved retention in the major [23, 24, 28]. However, prior research has typically focused on curriculum-level outcomes rather than the nuances of student engagement. This distinction is crucial, as the reported benefits may stem from factors other than the collaborative process itself. For instance, increased social interaction can bolster a sense of belonging and, subsequently, retention independent of collaborative learning mechanics [27, 31, 36]. Conversely, the difficulty in measuring individual contributions may mask social loafing [12], allowing students to achieve higher grades with reduced learning. Consequently, it remains an open question whether the benefits of unsupervised pair programming arise from the prescribed methodology or adjacent variables. In this paper, we take the first steps to address this gap by analyzing student self-reports on their engagement strategies for pair programming assignments completed outside the classroom.

While there is a body of literature showing the impact of pair programming on student grades and retention in computing, to our knowledge there has not been a study trying to understand if students are actually following the pair programming process when they are outside of the classroom. Increased course

pass rates and increased course retention rates have been found across both studies in which programming is performed in the classroom [2, 3, 4, 6, 18, 25, 28, 30] and when programming assignments have been assigned outside of class [9, 13, 14, 22, 23, 24, 37, 38, 40]. We focus on the assignments outside of the classroom because when students are working in the classroom, there can be more oversight on their process to make sure that they are correctly engaging in pair programming that cannot occur outside of the classroom.

In this study, we surveyed 43 students in computing courses across the program at a small liberal arts institution asking them to report on their process during pair programming assignments. We found that in the majority of the cases, students are not using the pair programming process and are in fact dividing up the work, suggesting this direction as a fruitful line of inquiry.

2 Background

Pair programming is a collaborative learning paradigm that scripts the collaboration process through the use of roles. During pair programming, the students take turns as the driver (who writes the code) and the navigator (who helps and looks for errors) [41]. The pair programming process is borrowed from industry from a methodology known as eXtreme Programming [1] where instead of students, two experts work together.

Studies have found that pair programming in undergraduate courses increases student success. Pair programming has been shown to increase students' scores on programming assignments and lab assignments, in some cases called program quality, but really a measure of performance [3, 13, 22, 23, 24, 25], while others find no difference [28]. Other studies looked at the program quality itself (written code), with mixed results on the effect of pair programming [9, 14, 22, 24]. Additionally, it has been found that assignments with pair programming have higher student submission rates [14] and attendance is higher for labs where pair programming is used [30].

In terms of course outcomes, pair programming has been shown to increase course completion and pass rates, retention in CS-related majors and retention to subsequent computing courses [3, 4, 23, 24, 28, 40]. Studies that focus on additional course outcomes have mixed results such as for final exam scores [22, 23, 24, 28, 30, 25], with the majority of cases finding no statistically significant difference between students who do and do not pair program [22, 23, 24, 28, 30, 42] and a few finding pair programming increases exam and post-test scores [17, 19, 26]. Some studies suggest that pair programming may lead to better course grade outcomes and final exam scores for students who struggle more with the material or had lower SAT scores [3, 30, 33].

What is not clear from these studies on student success and course out-

comes is whether students are actually engaging (or to what extent) in the pair programming process when instructed to. If students are not engaging in the process as prescribed, then further work is needed to understand what about assigning pair programming is beneficial.

Finally, there have been many studies that research students' perceptions of the pair programming process including whether they enjoy pair programming and whether it improves their confidence in the programs they produce [4, 9, 14, 23, 24, 40] and their satisfaction with their programming experience/created program [6, 14, 24]. In all cases, the majority of students have reported enjoying the pair programming experience [6, 9, 13, 21, 23, 24, 25, 26, 28, 33, 40]. Students cited many benefits of pair programming such as being able to verbalize their thoughts, having fewer bugs by catching each other's errors, perceiving the process as being more effective, allowing them to explore more ideas related to completing the assignment, and being good for helping meet people in their courses [6, 9, 33, 37]. The main issues cited with pair programming were having to resolve conflicts between partners [21], scheduling time to work with their partners [21, 37], and being paired with a student who has less experience [9]. If students are not able to overcome the issues cited, they may not engage in the pair programming process and may choose to complete their assignments in a different manner.

To our knowledge, no research has been conducted on understanding how students choose to complete their assignments in courses that require pair programming outside of the classroom. Our work is focused on understanding if students are actually following the pair programming process as this may have implications towards where the benefits of assigning pair programming lie and how we can better support the process.

3 Study Design

3.1 Research Question

The aim of this study is to investigate the pair programming process as it takes place outside of the classroom as reported by students. Specifically, we raise the following research questions:

RQ-1: When students are assigned a pair programming assignment, how do students self-report working together on the assignment?

RQ-2: How much do students self-report contributing to the assignment?

Investigating these questions is an important step to understanding what about pair programming makes is impactful.

3.2 Population

Our population of students includes undergraduate students enrolled in computer science courses at a small liberal arts institution in the United States. We recruited students who were either a computer science major/minor or were currently enrolled in a computer science class during Spring 2024. We asked them to reflect on their last experience pair programming in a course, which could be their current course or during a previous semester if they were not currently in a course that had pair programming. Courses that require or allow pair programming have a statement in their syllabus explaining the process. Instructors usually explain this process in class and often have it on assignments. Given that students are reflecting on their last experience, we cannot 100% verify instructor language or depth of explanations. Below is the statement that is used in the common course materials in our CS1 class and often replicated in higher-level classes:

“It is very rare that a real world software project will be done alone: keeping with that spirit, most PSAs in this course will require pair programming. Pair programming does not mean, “You do this assignment and I’ll do the next one.” Pair programming means actively working together on a program in the same room, using the same computer. In pair programming one person is the driver while the other is the navigator. The driver operates the keyboard and mouse while the navigator checks what the driver is doing and offers input about what to do next. These two roles should switch periodically. Studies have shown that students engaged in pair programming do better in subsequent programming courses. Here is a somewhat cheesy video (<https://www.youtube.com/watch?v=rGU12uqRhE>) that provides an overview on pair programming, including some tips for making it work effectively for you.

Please remember that it is your responsibility to understand all of the code that you submit. I will pair you with another student; the list of pairings will be available on Canvas. You should contact your partner immediately to introduce yourself and try to coordinate your schedules and find times to work together. If you do not hear back from your partner or if they tell you they are planning on dropping the course, please contact the instructor immediately. If I don’t hear from you, I will assume you are staying with your initial partner.”

3.3 Data Collection and Cleaning

This study was approved by our institutional review board. Following our approved human-subjects protocol, we administered a survey at the end of the 2024 spring semester asking students about their experiences collaborating on programming assignments in courses that use pair programming. The survey was anonymous and participants did not receive any incentives to participate

beyond the potential benefit of improving knowledge in CS education.

We employed two methods to recruit students to participate in the study. We sent out a call to participate via email to the computer science mailing list, a mailing list that includes all undergraduate students who have declared a B.S. or B.A. in computer science or who have declared a minor. At the time the survey was sent, there were 255 current majors and minors on the list. In addition, we reached out to all instructors teaching computer science courses in the spring semester and asked them to share the recruitment text with the students. In total 13 instructors responded and said they would announce the survey in their courses. In total 338 students were enrolled in computing courses during the time.

We received 50 responses, giving us approximately a 15% response rate. Of those responses, seven listed the course as either our software engineering course or senior project, neither of which require pair programming. These responses are excluded from the analysis. Therefore, our study population includes 43 responses. No students from the introductory computing course responded to the survey. Respondents were enrolled in a variety of courses including CS2, advanced data structures, networking, algorithms, digital hardware, human-centered systems, and automata.

3.4 Survey Content

Our survey consisted of three parts: (1) information on students' collaboration on projects using pair programming, (2) questions on their last experience working with a partner, and (3) student background and demographic data (e.g. anticipated graduation year, age, race/ethnicity, gender). This paper analyzes a subset of the survey questions related to division of labor. Below are the relevant questions we asked about their last experience working with a partner using pair programming:

1. Did you divide up work on the project?
2. What percentage did you do? (Please answer with a whole number between 0-100)
3. How did you divide up the work?
4. Any additional comments on your experiences collaborating on projects?

3.5 Participant Characteristics

All participants were students enrolled in undergraduate computing courses at a small liberal arts institution. See Table 1 for details on the participant

Table 1: Number of participants (N) and percentage of participants grouped by participant characteristics. Note in some cases the numbers do not add up to 100% as some students chose not to answer.

Group	Subgroup	N	%
Gender	Female	19	44.2%
	Male	22	51.2%
BLNPI Status	BLNPI	12	27.9%
	Non-BLNPI	28	65.1%
FirstGen Status	FirstGen	12	27.9%
	Non-FirstGen	29	67.4%
Transfer Status	Transfer	10	23.3%
	Non-Transfer	31	72.1%
Age	18-20	15	34.9%
	21-23	21	48.8%
	24+	5	11.6%
Graduation Year	2024	15	34.9%
	2025	19	44.2%
	2026	6	14.0%
	2027	2	4.65%

characteristics. For races/ethnicities that are typically underrepresented in computing we grouped students based on whether or not they identified as Black, Latino/a, Native American or Pacific Islander (BLNPI) or not (non-BLNPI).

The characteristics of the participants who responded to the survey aligned with overall department statistics with the exception of gender. In this case, students who identified as female (44.2%) responded to the survey at a higher rate than are in the department as majors and minors (25.5%).

3.6 Qualitative Analysis

To understand how students divided the work on the programming assignments, we asked students the free response question “How did you divide up the work?”. To analyze this question, the second and third authors on this paper performed a qualitative analysis using open-coding. There were five rounds of the open-coding phase of the analysis, resulting in a list of codes that we

believe accurately capture the divisions of labor. The codes were improved through an iterative process where the authors coded the responses independently, followed by comparing the coding and discussing any differences until agreement was met. This process was repeated for the next round. A hierarchy of codes was created including three high-level codes and five sub-codes. The 40 responses to this question were coded with high-level codes and 32 responses for sub-codes. Three students did not respond to this question. In the final phase of categorization, Cohen's Kappa was used to calculate the inter-rater reliability. Cohen's Kappa for the final round was 90.6% for the high-level codes and 83.0% for the sub-codes.

4 Results

4.1 RQ-1: Assignment Process

In order to understand how students self-report working together on an assigned pair programming assignment, we first looked at their response to the yes or no question: "Did you divide up work on the project?", to which all 43 participants responded. The majority of students, 31 (72.1%) of the responses, reported dividing up the work.

In order to further understand how students worked together on the pair programming assignments, we performed a qualitative analysis using open-coding on the free response question "How did you divide up the work?". Below we present the final coding scheme for the analysis. We do not provide the associated counts of participant responses because students may not have included all of the methods that they used to divide the work in their responses.

4.1.1 High-Level Categories.

No Division. Students reported working together at the same time for the entirety of the assignment, not on their own time. In some cases, this included the definition of pair programming.

- *"Regarding my partner for algorithms, we basically do everything together, talk through every solution, one person coding and the other pulling up resources. We both do 100% of the work I would say."*
- *"We didn't divide up the work, we worked on each part together."*
- *"We looked at it together and worked on fixing and discussing out-loud."*

Partial Division. Students reported dividing up the work to complete separately some of the time, but also working together.

- *“We start the projects together and she usually finishes them and lets me know they’re done.”*
- *“Both worked separately to figure out methods to complete the project then came together to finish the rest of the project with our ideas together.”*

Full Division. Students reported dividing the work between partners to work separately. They reported a full division of work.

- *“I assigned myself certain sections of a project that could be done separately, and my partner worked on the other sections.”*

4.1.2 Sub-Categories.

Responses coded as full or partial division were analyzed further to identify how work was divided.

Tasks by Section. Students reported dividing the work by sections of the assignment for each partner to complete.

- *“Every other question was mine to answer.”*
- *“By sections of the project.”*
- *“We divided up the functions that each of us would implement.”*

Time. Students reported dividing the work by their availability to work on the project.

- *“Our schedules were different so we divided up work and did what we could based on when we had the time to do so.”*
- *“Whatever there was to do after we last worked together for however long they had the time do finish.”*

Skills. Students reported dividing the work by their and their partner’s skills/strong suits.

- *“We based on what we thought our strengths on the project were allowing us to get it done faster and be more efficient.”*
- *“It was more a matter of whoever understood the logic for the assignment better ended up doing more of the work.”*
- *“Based on time/availability, expertise.”*
- *“Based on interests and strong suits.”*

Preference. Students reported dividing the work by based on their and their partner's interests and likes/dislikes.

- *“Whatever we wanted to do and what needed to be done.”*

Cannot Parse. In some cases the description for the division of labor was too vague to interpret or no method of division of labor was provided.

- *“Switched back and forth.”*
- *“Did whatever I could, tried to get partner to do the rest.”*
- *“Whoever could figure out the working algorithm first.”*

There were seven discrepancies between the researcher's consideration of no division and the student's response to “Did you divide up work on the project?”. For the seven cases, there was an even divide between if the student marked a no but the researchers considered their description as a division and the students that marked a yes and the researchers considered their description as working together.

In three cases, students who worked together synchronously considered this to be a division of labor. For example, *“We sat together with both of computers and used live share so we could talk through the problems and both code.”* In three other cases, if a student contributed more to the project, they marked this as no division. For example, *“I did most of the work.”* Finally, there was one answer which contradicted itself, marking the division of labor column with a no but the answer being: *“Allocating tasks based on strengths.”*

Given that for the majority of students there was alignment between these two questions and for the students where there was not the discrepancies were even in both directions, we do not believe this changes our results.

4.2 RQ-2: Assignment Contribution

In order to understand how much students view contributing to the project, we looked at responses to the following question, “What percentage did you do?” Student responses ranged from 10-99%, with a mean of 55.9% and standard deviation of 14.9% (see Figure 1). Most of the students self-reported doing an even or almost even amount of work with their partner. Students that reported doing an even amount of the work were both students that reported dividing the work and those that reported working together.

The two students who self-reported the minimum and maximum of 10% and 90% believed they did very little or almost all of the work respectively. Here are their statements on how the work was divided:

- *“I did most of the work”*

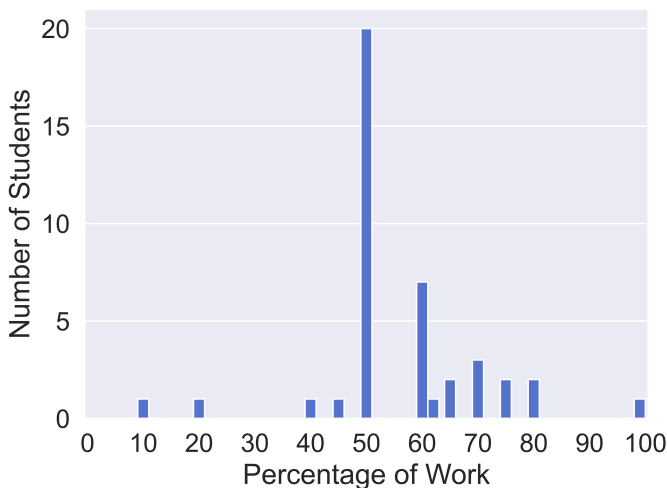


Figure 1: Distribution of reported percentage of work.

- “We start the projects together and she usually finishes them and lets me know they’re done.”

Perhaps surprising, the student who reported doing 20% of the work reported following the pair-programming paradigm: “In a group of 2, we usually switch our roles as the “driver” or the “navigator”. The process of switching roles may based on the completion of specific functions or tasks.” This will be discussed further in Section 5.

5 Discussion

5.1 Students May Not Be Pair Programming

We find that over 70% of the students who filled out the survey divided up the work on their pair programming assignments rather than following the pair programming best practices or even collaborating. However, students do still seem to feel that the work is being split evenly given that the majority of students reported completing 45-65% of the work on the assignment.

Given that most students appear to divide their work and do not collaborate, it raises the question of what processes lead to the positive impacts on retention and completion of the course seen in previous studies [23, 24, 25, 26, 28]. It could be that the positive impact we see on retention and course completion comes from allowing students to engage in cooperative learning and

complete work as a team rather than coming from the use of pair programming itself. On the other hand, there is the possibility that in previous studies students did engage in the pair programming process, and it was the engagement in this process that led to the positive impacts. We did not collect retention or course completion data so we are unsure of the relationship the processes our students engaged in may have had with these measures. Although our results cannot answer this question, by finding that students do not often follow pair programming best practices, we show that further investigation is needed to understand what mechanisms may be leading to these beneficial outcomes.

In addition to students just dividing the work, what may be more concerning are their heuristics for the division of the work, including time available and preference and skills. Several quotes that fall into this category include: *"We based on what we thought our strengths on the project were allowing us to get it done faster and be more efficient."*, *"Allocating tasks based on strengths"*, and *"It was more a matter of whoever understood the logic for the assignment better ended up doing more of the work."* The goal of most programming assignments is to give students practice with the concepts that they are learning in class. When students divide the work just based on when they have time to work on it or by preferences or skills where the student who has a higher mastery of the skill works on that section, then they are not getting practice in the areas where they need it most.

These results align with previous research that has found mixed results on the impact of pair programming on student learning outcomes such as course grades and final exam scores [22, 23, 24, 25, 28, 30]. It may be that allowing students to pair program, but where they do not actually pair program and may divide the work in a manner that is not conducive to learning, can lead to higher scores on their programming assignments or lab grades and, therefore, is allowing them to pass the course without an increase in learning benefits. If we are able to improve the level at which students actually follow the pair programming process outside of the classroom, could we improve outcomes even more?

Finally, students self-reported percentages of the work they did raises the question if it is clear what an effective pair programming paradigm entails. Interestingly, one student who stated they followed the pair programming process put 20% as their reported percentage of work. There are multiple ways to interpret this. Was the student a passive navigator and not contributing when not typing? Did the student and their partner not switch roles often enough? Did the student, irrespective of roles, not feel like they were contributing the conceptual ideas that led to the problem being solved? Although researchers have investigated how the collaborative process relates to learning [5, 34], there is still an open question on what effective instructional guidance may entail [16].

5.2 Additional Responses

Students had the option to write extra thoughts through an optional free response question, “Any additional comments?” which resulted in 14 responses. We looked at these comments to see if there were any insights into why students may have chosen not to pair program. Given the limited information provided, in this section we discuss common notions of the comments. Of these responses, many highlight the primary influence their partner has on their experience pair programming. For example, a student wrote: *“I think group projects are great but only if you have a good partner that’s at the same level as you who is willing to put in the work. Bad partners are no fun and it’s not always guaranteed that you’ll have a good partner.”* This comment calls attention to the effectiveness and enjoyment from having a “good partner”, which leads to the question of what exactly makes a “good partner”.

Similarly, many comments discuss how knowing their partner beforehand has resulted in a more positive pair programming experience. This can indicate familiarity has an impact on a pair’s collaboration, but further research is required to examine this. Other comments noted a preference to work independently rather than collaborate on projects. One student noted, *“Collaboration genuinely slows me down and makes my projects worse...”* This and other comments provide insight to negative experiences of pair programming, which reasons for why students prefer to work alone can be studied in the future. A small minority of comments listed that time scheduling complexities and the class/project itself contributed to the experience as noted in *“sometimes its hard to find times to work together”*.

These comments may explain part of the reason students are choosing not to pair program on their assignments. Prior work suggests that these are common issues students report during pair programming [9, 21, 37]. Given that these are common issues, it raises the question of when pair programming may be beneficial to use within a class and if it is best to restrict its use to scheduled class and lab sessions.

5.3 Limitations and Risks to Validity

Our results are based on a single small liberal arts institution which may limit the generalizability of our findings. However, our curriculum follows the ACM Curriculum guidelines [8], which is followed by many institutions.

One threat to validity is that only a sub-population of students filled out the survey. This is particularly true of students in our CS1 class where we received no responses. This lack of response may be due to many students in that class in the spring semester not being intended majors and not finding any benefit to answering the survey. It may also be due to the students often

being earlier in their college career and being unsure of the solicitation. It may be that we would find a higher level of pair programming if all students responded. In future studies, if we are able to make the survey as part of the class feedback, we would expect to get more participation. Students may also have answered based on what they thought we wanted them to say but we do not believe this to be the case given that the majority of students report not following pair programming best practices. Furthermore, we had an over representation of women in our sample, which may have skewed the results.

Additionally, we can only report on students' perceptions of their experience, not the experience itself. The reported division of labor is likely accurate, but the reported amount of work completed by the students may be inaccurate. For example, a student who followed pair programming estimated their contribution as 20%. Students may have interpreted the question as the amount of time typing, the conceptual contributions made, or perhaps the amount of code they were present for when it was written. In future iterations we plan on updating the question to tease apart these possibilities. Unfortunately, it is not possible to report on anything beyond student perception without in-person monitoring which would likely change the dynamic of the pair.

5.4 Implications

These results are concerning, as students do not seem to be engaging in the pair programming process as intended. If collaboration is a key mechanism to the success of pair programming, then when students divide the work, they do not engage in interactive learning, which allows students to develop and revise their mental models and deepen their learning [7, 15]. However, effective collaboration does not occur spontaneously [35], and we should not expect students to naturally engage in the process without receiving collaboration support to be successful [39]. Furthermore, given that students do not all start with the same expectations about collaboration, adaptive collaborative learning support may be needed to provide students with the correct support at the correct time [10]. Before building out support for the pair programming process, it is important to identify what from the paradigm is beneficial to students and how best to support it.

In the future we plan on running our survey at more institutions to see if our results replicate and interview students to gain more insight into why they are not following the pair programming process.

6 Conclusion

We conducted a preliminary survey to understand how undergraduate students across all four years at a small liberal arts institution are collaborating on pair

programming assignments outside of the classroom. We find that the majority of students (72%) are dividing up the work to complete on their own and very few students are trying to engage in the pair programming process. Given the positive outcomes prior work has found for pair programming, this begs the question: is pair programming outside of the classroom leading to these positive student outcomes or is it simply allowing students to collaborate and work on teams? We would expect that inside of the classroom students would be more likely to follow pair programming best practices because instructors can provide immediate feedback and monitor the process more closely. More work needs to be done to understand how students are collaborating outside of the classroom, how we can improve the practices they choose to use, and if pair programming should be limited to inside the classroom.

References

- [1] Kent Beck. *Extreme programming explained: embrace change*. Addison-Wesley, 2000.
- [2] Nicholas A. Bowman et al. “How Prior Programming Experience Affects Students’ Pair Programming Experiences and Outcomes”. In: *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*. ITiCSE ’19. 2019, pp. 170–175.
- [3] Grant Braught, L. Martin Eby, and Tim Wahls. “The Effects of Pair-Programming on Individual Programming Skill”. In: *Proceedings of the 39th SIGCSE technical symposium on Computer science education*. 2008, pp. 200–204.
- [4] Grant Braught, Tim Wahls, and L. Marlin Eby. “The Case for Pair Programming in the Computer Science Classroom”. In: *ACM Transactions on Computing Education* 11.1 (2011).
- [5] Mehmet Celepkolu and Kristy Elizabeth Boyer. “The Importance of Producing Shared Code Through Pair Programming”. In: *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. SIGCSE ’18. 2018, pp. 765–770.
- [6] Mehmet Celepkolu and Kristy Elizabeth Boyer. “Thematic Analysis of Students’ Reflections on Pair Programming in cs1”. In: *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. SIGCSE ’18. 2018, pp. 771–776.
- [7] Michelene T.H. Chi and Ruth Wylie. “The ICAP Framework: Linking Cognitive Engagement to Active Learning Outcomes”. In: *Educational psychologist* 49.4 (2014), pp. 219–243.

- [8] ACM/IEEE-CS Joint Task Force on Computing Curricula. *Computer Science Curricula 2013*. 2020. URL: <https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2020.pdf>.
- [9] Timothy H. DeClue. “Pair Programming and Pair Trading: Effects on Learning and Motivation in a CS2 Course”. In: *Journal of Computing Sciences in Colleges* 18.5 (2003), pp. 49–56.
- [10] Pierre Dillenbourg. “Over-Scripting CSCL: The Risks of Blending Collaborative Learning with Instructional Design.” In: *Three worlds of CSCL. Can we support CSCL?* (2002), pp. 61–91.
- [11] Pierre Dillenbourg. “What do you mean by collaborative learning?” In: *Collaborative-learning: Cognitive and computational approaches*. (1999), pp. 1–19.
- [12] Marylene Gagne and Miron Zuckerman. “Performance and learning goal orientations as moderators of social loafing and social facilitation”. In: *Small Group Research* 30.5 (1999), pp. 524–541.
- [13] Edward F. Gehringer. “A Pair-Programming Experiment in a Non-Programming Course”. In: *Companion of the 18th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications. OOPSLA '03*. 2003, pp. 187–190.
- [14] Brian Hanks et al. “Program Quality with Pair Programming in CS1”. In: *Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education. ITiCSE '04*. 2004, pp. 176–180.
- [15] Robert Hausmann, Michelene Chi, and Marguerite Roy. “Learning from Collaborative Problem Solving: An Analysis of Three Hypothesized Mechanisms”. In: 2004, pp. 547–552.
- [16] Anja Hawlitschek, Sarah Berndt, and Sandra Schulz. “Empirical research on pair programming in higher education: a literature review”. In: *Computer science education* 33.3 (2023), pp. 400–428.
- [17] Tie Hui Hui and Irfan Naufal Umar. “Pair Programming and LSs in Computing Education: Its Impact on Students’ Performances.” In: *Online Submission* (2011), pp. 613–626.
- [18] Lindsay Jarratt et al. “A Large-Scale Experimental Study of Gender and Pair Composition in Pair Programming”. In: *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education. ITiCSE '19*. 2019, pp. 176–181.

- [19] Subbaraya Kuppuswami and Kalimuthu Vivekanandan. “The Effects of Pair Programming on Learning Efficiency in Short Programming Assignments”. In: *Informatics in Education* 3.2 (2004), pp. 251–266.
- [20] Yiping Lou, Philip C Abrami, and Sylvia d’Apollonia. “Small group and individual learning with technology: A meta-analysis”. In: *Review of educational research* 71.3 (2001), pp. 449–521.
- [21] Ian McChesney. “Three Years of Student Pair Programming: Action Research Insights and Outcomes”. In: *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. SIGCSE ’16. 2016, pp. 84–89.
- [22] Charlie McDowell, Brian Hanks, and Linda Werner. “Experimenting with Pair Programming in the Classroom”. In: *Proceedings of the 8th Annual Conference on Innovation and Technology in Computer Science Education*. ITiCSE ’03. 2003, pp. 60–64.
- [23] Charlie McDowell et al. “Pair Programming Improves Student Retention, Confidence, and Program Quality”. In: *Commun. ACM* 49.8 (2006), pp. 90–95.
- [24] Charlie McDowell et al. “The Impact of Pair Programming on Student Performance, Perception and Persistence”. In: *25th International Conference on Software Engineering, 2003. Proceedings*. IEEE. 2003, pp. 602–607.
- [25] Emilia Mendes, Lubna Al-Fakhri, and Andrew Luxton-Reilly. “A Replicated Experiment of Pair-Programming in a 2nd-year Software Development and Design Computer Science Course”. In: *Proceedings of the 11th Annual Conference on Innovation and Technology in Computer Science Education*. ITiCSE ’06. 2006, pp. 108–112.
- [26] Emilia Mendes, Lubna Basil Al-Fakhri, and Andrew Luxton-Reilly. “Investigating Pair-Programming in a 2nd-year Software Development and Design Computer Science Course”. In: *SIGCSE Bull.* 37.3 (2005), pp. 296–300.
- [27] Sukanya Kannan Moudgalya et al. “Measuring students’ sense of belonging in introductory CS courses”. In: *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. 2021, pp. 445–451.
- [28] Nachiappan Nagappan et al. “Improving the CS1 Experience with Pair Programming”. In: *Proceedings of the 34th Technical Symposium on Computer Science Education*. SIGCSE ’03. 2003, pp. 359–362.

- [29] Jerzy R. Nawrocki et al. “Pair Programming vs. Side-by-Side Programming”. In: *Software Process Improvement*. Ed. by Ita Richardson, Pekka Abrahamsson, and Richard Messnarz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 28–38.
- [30] Clem O'Donnell et al. “Evaluating Pair-Programming for Non-Computer Science Major Students”. In: *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*. SIGCSE '15. 2015, pp. 569–574.
- [31] Patrick O'Keeffe. “A sense of belonging: Improving student retention”. In: *College student journal* 47.4 (2013), pp. 605–613.
- [32] Lutz Prechelt, Ulrich Stärk, and Stephan Salinger. “7 types of cooperation episodes in Side-by-Side programming”. In: (2008).
- [33] Alex D. Radermacher and Gursimran S. Walia. “Investigating the Effective Implementation of Pair Programming: An Empirical Investigation”. In: *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*. SIGCSE '11. 2011, pp. 655–660.
- [34] Fernando J Rodriguez, Kimberly Michelle Price, and Kristy Elizabeth Boyer. “Exploring the pair programming process: Characteristics of effective collaboration”. In: *Proceedings of the 2017 acm sigcse technical symposium on computer science education*. 2017, pp. 507–512.
- [35] Nikol Rummel and Hans Spada. “Learning to Collaborate: An Instructional Approach to Promoting Collaborative Problem Solving in Computer-Mediated Settings”. In: *The Journal of the Learning Sciences* 14.2 (2005), pp. 201–241.
- [36] Linda J Sax et al. “Sense of belonging in computing: The role of introductory courses for women and underrepresented minority students”. In: *Social Sciences* 7.8 (2018), p. 122.
- [37] Beth Simon and Brian Hanks. “First Year Students' Impressions of Pair Programming in CS1”. In: *Proceedings of the Third International Workshop on Computing Education Research*. ICER '07. 2007, pp. 73–86.
- [38] Tammy VanDeGrift. “Coupling Pair Programming and Writing: Learning about Students' Perceptions and Processes”. In: *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*. SIGCSE '04. 2004, pp. 2–6.
- [39] Armin Weinberger et al. “Epistemic and Social Scripts In Computer-Supported Collaborative Learning”. In: *Instructional Science* 33 (2005), pp. 1–30.
- [40] Linda L. Werner, Brian Hanks, and Charlie McDowell. “Pair-Programming Helps Female Computer Science Students”. In: *Journal on Educational Resources in Computing (JERIC)* 4.1 (2004), 4–es.

- [41] Laurie Williams et al. “Building Pair Programming Knowledge Through a Family of Experiments”. In: *2003 International Symposium on Empirical Software Engineering, 2003. ISESE 2003. Proceedings.* IEEE. 2003, pp. 143–152.
- [42] Laurie Williams et al. “Pair Programming in an Introductory Computer Science Course: Initial Results and Recommendations”. In: *OOPSLA Educator’s Symposium, Seattle, WA.* 2002.

Putting “Active” into Learning Theory of Computation*

Tammy VanDeGrift
Computer Science, Shiley School of Engineering
University of Portland
Portland, OR, 97203
vandegri@up.edu

Abstract

Active learning techniques engage students in methods other than listening and watching a lecture. Active learning in computer science can take many forms: group problem-solving, project-based learning, peer instruction, and interactive lab exercises. This paper presents several active learning activities in an undergraduate-level Theory of Computation course, where students engage with the material through physical movement around the classroom or use their hands to explore concepts. While the activities are written specifically for Theory of Computation, they are general enough for use in other computing courses. The main result of this report is that students rated the in-class activities highest among the course materials that supported their learning. The instructor observed that student engagement was high throughout the duration of activities. Because the activities were done in groups, another positive outcome is that students met more classmates.

1 Introduction and Related Work

Studies have shown that undergraduate students, especially in science, technology, mathematics, and engineering, are more successful learners when instructional time uses active learning. Undergraduate students enrolled in physics

*Copyright ©2026 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

courses with active learning techniques performed better on assessments; however, students self-reported that they learned less in classes that used active learning [6]. Hao and others studied introductory computer science student performance in a lecture-based course and an active-learning course and found that students in the active-learning course did better, especially on the last exam in the course [13]. A comprehensive meta-review of over 200 studies of STEM classes that used active learning versus more traditional lectures showed that active learning contributed to better performance on tests and concept inventories, while also lowering failure rates [9].

The computer science education community has embraced active learning across the curriculum. For example, Gehringer and Miller invited students to design active learning activities for CS1 and CS2 courses [10]. Grinder and others designed an Information Search and Retrieval course that used gamification, regular feedback, and collaborative group work [12]. Dehbozorgi offered a framework for patterns to help instructor apply best practices in active and collaborative learning [5].

This paper focuses on using active learning in an upper-division undergraduate Theory of Computation course, contributing to the set of instructional materials and tools to support student learning. Papers by Frede [8] and Sigman [19] argue that learning theory of computation is challenging for students, given the mathematical and formal analyses that students need to develop. Bqa and others describe a sequence of content for a Theory of Computation course to support students' understanding of language classes [3]. An example of active learning in a Theory of Computation course is using peer review of student-written papers [7]. Another example asks students to simulate a Turing machine outdoors with chalk on a sidewalk [11].

One approach to engage students in design is through simulation tools. Computer science faculty and researchers have developed software to support construction of automata, such as the Java Computability Toolkit [17], JFLAP [18, 26, 1], and a Prolog toolkit for formal languages [27]. Interactive visualizations and auto-graded exercises can also help students learn to design automata [15]. While the course described in this paper uses JFLAP for constructing automata, the focus of this paper is on technology-free activities.

This paper presents a collection of activities that gets students to move or to use their hands to manipulate something physical. While movement-based learning, also known as kinesthetic learning [23], is often used in primary school, the instructional approach is used less often in higher education. Even still, college students can benefit from incorporating movement while they learn. Students in an upper-division engineering course report that kinesthetic learning is fun, helps with processing course concepts, helps with recall during exams, and keeps their energy high [23]. CS Unplugged activities use movement to

give young people exposure to computational thinking [28]. Kinesthetic learning has been used in computer science classes: 1) students act as cons cells, and 2) students act as recursive function calls [2]. Following the introduction, the educational context is presented. Section 3 describes the activities and lessons, so other instructors can adapt them, and the next section presents an evaluation of the activities.

2 Educational Context

The activities took place in an undergraduate Theory of Computation (ToC) course at a small, private, comprehensive university on the west coast of the USA, offering a major and minor in Computer Science. This experience report focuses on the ToC course offered in Fall 2023. Section A had 16 students and section B had 20 students.

The ToC course is required for the CS major and is typically taken by third-year and fourth-year undergraduate students. The prerequisites for ToC are Discrete Math and Data Structures. The ToC course includes automata (deterministic finite automata, non-deterministic finite automata, pushdown automata, Turing Machines), regular expressions, grammars, formal proofs, the pumping lemmas, undecidability, complexity class P, class NP, and NP-complete languages. ToC is a prerequisite course for the Compiler Design course, which is also required for the CS major.

3 “Active” Learning Activities

This section describes a series of kinesthetic learning activities to support students learning concepts in Theory of Computation. Each description includes the purpose, content, materials, and the accompanying lesson. In some cases, students walk to different parts of the classroom. In others, students use their hands to manipulate cards or game pieces.

3.1 Blobs and Lines: Students Meet Each Other

Purpose: This is an ice-breaker for students to meet their classmates and set the stage that the learning environment will ask students to do more than sit, write, and listen during the semester.

Content: There is no specific ToC content for this activity.

Materials: Categorical questions for blobs and numerical/comparable questions for lineups need to be created by the instructor. It is recommended that at least four questions of each type are prepared so students can have several quick conversations. Desks may need to be moved to the side of the room.

Lesson: [35 minutes] The instructor created the following questions for blobs and lines for ToC:

- Lineup: Think about where you were one week ago and estimate the distance to this classroom. Form a single line of people from the smallest to largest distance.
- Blob: Make blobs according to your favorite color.
- Lineup: Form a line for who is most ready to least ready to start the semester.
- Blob: Make blobs according to favorite social media platform.
- Lineup: Form a line from smallest to biggest based on how many other people you already know in this class.
- Blob: Make blobs according to how you got to campus today (ex: walk, bike, skateboard, bus, car).
- Lineup: Form a line so students are in alphabetical order by preferred name.
- Blob: Make blobs according to favorite data structure.

After each lineup and blob, give students a couple minutes to introduce themselves to their blob or lineup neighbors. It is recommended that the instructor participate as well, inserting themselves into the lines and blobs, so they can also have quick conversations with students. Each lineup or blob question takes three to five minutes for students to move around, ask questions, and introduce themselves to nearby students. If time allows, ask students to suggest a blob or lineup question. If students have mobility challenges, encourage students to form the line or blob around students that need to be stationary. This activity works well in classes that have up to 25 students. With more than 25 students, sub-groups of about 20 students can form independent lines and sets of blobs.

3.2 Jigsaw: Students Build and Explain DFAs

Purpose: This activity uses the jigsaw method [4]. Students get to work in two different groups as part of the activity. In the first grouping, students work together to create a DFA for a language. Groups are then re-formed such that the second grouping is comprised of students who created solutions for different languages. Each student provides an overview of the DFA designed from their first group.

Content: This activity focuses on regular languages and, in particular, creating deterministic finite automata. This activity could be used for any type of problem, however.

Materials: The instructor should prepare one regular language per four to five students in the class. For example, in a class of 20 students, the instructor

should select five regular languages and write each on a sheet of paper. Bring blank paper, so students can draw their DFAs. Bring playing cards (one to two decks) to randomly assign the second grouping.

Lesson: [25 minutes] Ask students to form groups of four to five students, such that the total number of groups corresponds to the number of regular languages prepared by the instructor. Assign each group a different language and have them collectively design a DFA to accept the language. Every member is responsible for understanding their DFA solution and for writing it on their own piece of paper. After about eight to ten minutes of working together (or longer if students need more time), assign each student a number within their group (1, 2, 3, 4...). The instructor typically does this with playing cards or two sets of playing cards. Prior to class, the instructor makes sets of cards (1 to 5) to do the second grouping. Have the 1s, 2s, 3s, etc. move to different parts of the classroom and re-form teams. Ask the new groups to share their DFAs by showing it to their group and asking what language they think it accepts. After some guesses (hopefully, one is correct), the student shares how to process some example strings through the DFA and how the DFA was constructed to accept the language. Ring a bell every two minutes, so students know when to move to the next person in the group. Here are the languages the instructor used:

- $\{w \mid w \text{ ends in } 00 \text{ over alphabet } \{0, 1\}\}$
- $\{w \mid w \text{ does not contain substrings } ab \text{ nor } ba \text{ over alphabet } \{a, b\}\}$
- $\{w \mid w \text{ has length at most } 4 \text{ over alphabet } \{0, 1\}\}$
- $\{w \mid w \text{ is not the empty string over alphabet } \{a, b\}\}$

3.3 Poster Walk: Students Summarize Regular Languages

Purpose: The main purpose of this activity is to encourage students to review regular languages prior to taking an exam by creating group posters and doing a gallery walk [21]. Instead of the instructor reviewing the topics, this gives students a chance to converse and create a shared document about definitions, notation, transformations, and machines. In prior years, the instructor would ask students to bring questions to the class session allocated for exam review. Often times, students had no questions, so the instructor went over topics. With the posters, students had to review and discuss the topics, which generally led to questions while they were making the posters or after they reviewed other groups' posters.

Content: At this point in the semester, students have practiced working with DFAs, NFAs, regular expressions, regular operators, and proving languages are regular.

Materials: If possible, the instructor should bring large poster-sized sheets

of paper and markers of various colors. If that is not feasible, students could create posters on different parts of a whiteboard/chalkboard in the classroom.

Lesson: [25 minutes] Assign students into groups of three to four students. Give the group ten to fifteen minutes to create a poster to demonstrate what they know about regular languages. They can use other resources, the textbook, and the course website to review information. Remind the students that content matters more than artistry. Sometimes, students get nervous about putting pen to paper without a rough draft, but remind them that content is what is important in this activity. After groups have created posters, have students walk around to view other posters to see what other groups included. This might take another ten minutes, depending on how many posters are made. Hopefully, the set of posters covers most of the material for the unit. This is a great way for the instructor to see if there is critical material missing prior to the exam. Have a class discussion about what they saw on the posters that jogged their memory to see if they want to review more topics/techniques prior to the exam. The instructor can give students a chance to vote on best poster by attaching a post-it note on what they deem as best poster. The poster designers with the most votes can earn extra credit, candy, or points toward the exam (instructor choice). Figure 1 shows three student-created posters.

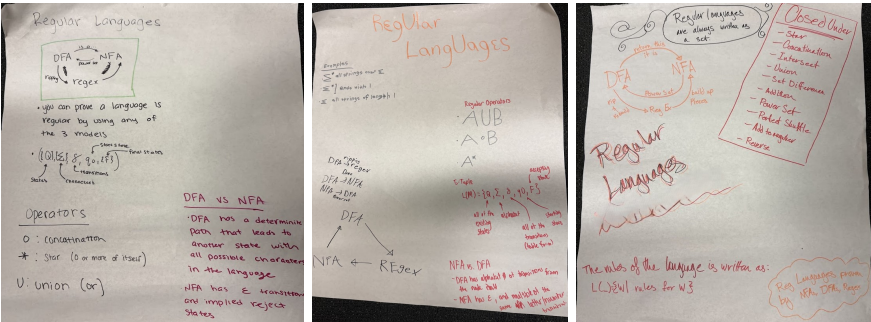


Figure 1: Examples of posters about regular languages

3.4 Building Strings: Students Create Grammars

Purpose: The instructor of ToC has observed that students are challenged to think recursively when constructing grammars for context-free languages. In the past, students have over-complicated their grammar construction. Also, giving students more practice with recursion helps with their skills as computer scientists. This keeps students thinking of base cases (shortest strings) for rules

and recursive cases (extending strings) for rules.

Content: This activity focuses on creating grammars for context-free languages.

Materials: The instructor should prepare card-stock cards or post-it notes. One each card or post-it, write a 0 or a 1. Put ten cards with 0 and ten cards with 1 in an envelope. Prepare one envelope for each activity group in the class.

Lesson: [20 minutes] Have each group take the cards out of the envelope. The instructor can write the language (see below) on the board or project it from a computer. For the language, ask the groups to form the shortest string in the language. (No card represents the empty string.) Ask the groups to create a rule from grammar start variable S to the shortest string or to the set of shortest strings. Then, ask the groups to lengthen the shortest string to the next shortest string in the language. Ask them to think about how to create a recursive rule that fits the extension pattern/invariant. Have the groups complete the grammar for the language. The instructor can check in with groups or ask a random group to share their answer with the class. Working through each language takes about five minutes. Here are the languages used in the activity:

- A: $\{0^n 1^n \mid n > 0\}$
- B: $\{w \mid w \text{ starts and ends with the same character (actually regular, but can still write the grammar)}\}$
- C: $\{w \mid \text{length of } w \text{ is odd (actually regular, but still still write the grammar)}\}$
- D: $\{w \mid \text{length of } w \text{ is odd and exact middle character is } 0\}$

Specific Example: Figures 2 and 3 shows the cards for language D above. The instructor asked students to build the shortest string for D. The shortest string 0 is shown in Figure 2. Then the instructor asked the student groups to write that as a grammar rule, so students wrote $S \rightarrow 0$. The instructor asked students to use that shortest string to build the next shortest strings(s) that are in language D. Students figured out to add one character to the left and one character to the right of the original zero. Those cards are shown in Figure 3. The instructor asked the students to add a rule to the grammar that represents that process. (Assume a string is in language D. How does one lengthen that string and create the next shortest string(s) in the language?) Students then wrote rules, and most wrote $S \rightarrow \Sigma S \Sigma$. The full grammar would then be $S \rightarrow 0 \mid \Sigma S \Sigma$. Having students manipulate cards helps them map the lengthening of strings to grammar rule(s). Before doing this activity with cards, the instructor noticed misconceptions about how to use recursion in grammar rules. In earlier iterations of the course, students wrote incorrect

rules for language D like $S \rightarrow 0|T0T; T \rightarrow \Sigma T|\epsilon$, which allows for inconsistent substring extensions before after the 0.



Figure 2: Shortest string for language D - length of w is odd and exact middle character is 0

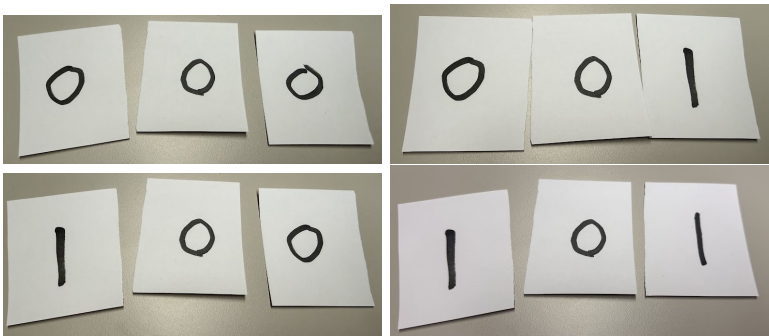


Figure 3: Next shortest strings for language D - length of w is odd and exact middle character is 0

3.5 Floor Tape: Students Simulate TM

Purpose: The purpose of this activity is to give students a memorable way to simulate a Turing machine, by reading the tape, writing to the tape (if overwritten), and moving left or right across the tape. It is inspired by the activity described in [11], but can be done indoors.

Content: This activity introduces students to the movement of deterministic Turing machines and reading/writing to tape cells.

Materials: The instructor should prepare example deterministic Turing machine state diagrams that can be projected to the classroom or provided as handouts. If the instructor wants to showcase that a Turing machine may not halt, they can prepare a “busy beaver” Turing machine diagram. The instructor should bring several sheets of blank paper and a marker.

Lesson: [30 minutes] Show the deterministic Turing machine state diagram to the students. The instructor who developed this activity used the Turing

machine for the language for $w#w$ where w consists of 0's and 1's. Ask a student to volunteer to be the head of the Turing machine. Give that student a marker and a stack of paper in case they need to overwrite what is on the cell. Ask another student for a sample input string and have them write it on the tape cell papers on the floor (one symbol per piece of paper). Have the Turing machine head student start at the first character of the input string. Ask them to execute one step of the Turing Machine. For the $w#w$ Turing machine, this likely overwrites the 0 or 1 with an x. Figure 4 shows the papers on the floor during different stages of the activity. Have the "head" student write x on a new sheet of paper and replace the original sheet on the floor to represent that the cell data is overwritten. The student will then step one cell to the right to mimic the movement of the Turing machine. It is advisable that the instructor (or a third student volunteer) keep track of the current state within the Turing machine by writing the current state on the board. If the Turing machine started in q_0 and is now in q_1 after one step, then write q_1 on the board. Have the Turing machine "head" student continue the simulation, step by step until the machine halts or it is evident that the machine will not halt. The instructor may want to run the simulation a few times with different volunteers (a string that is accepted, a string that is rejected, a string with invalid characters). A single simulation can take up to five minutes, since it is best to go slow, so the whole class knows the current state and the current content of the tape. The class can help direct the student walking the tape as they process the transitions in the Turing machine state diagram. Note that the number of papers on the floor is finite, so the instructor should remind students that the tape continues indefinitely on both ends (or just add papers to the left and right ends of the tape as needed). Also, the instructor may want to remind students that the simulation is for a single-tape, deterministic Turing machine. A non-deterministic machine would need to replicate versions of the tape contents at each non-deterministic branch, which may be difficult to simulate on the floor in the classroom.

3.6 Playing Games: Students Make Connections

Purpose: Learning should also be fun. This activity lets students explore three games related to Turing machines. The first game is called Turing Tumble, which can be purchased as a physical apparatus [20] or can be played via an online simulator [24]. Turing Tumble uses logic-like widgets on a vertical peg board to move input (red and blue balls) at the top of the peg board to a string of output (red and blue balls) at the bottom of the peg board. The second game is called Code Master [16], which consists of puzzles of getting from one state to another on a map and using code-like instructions. The third game is Game of Life [14], which is a simulation of generations of cells on a

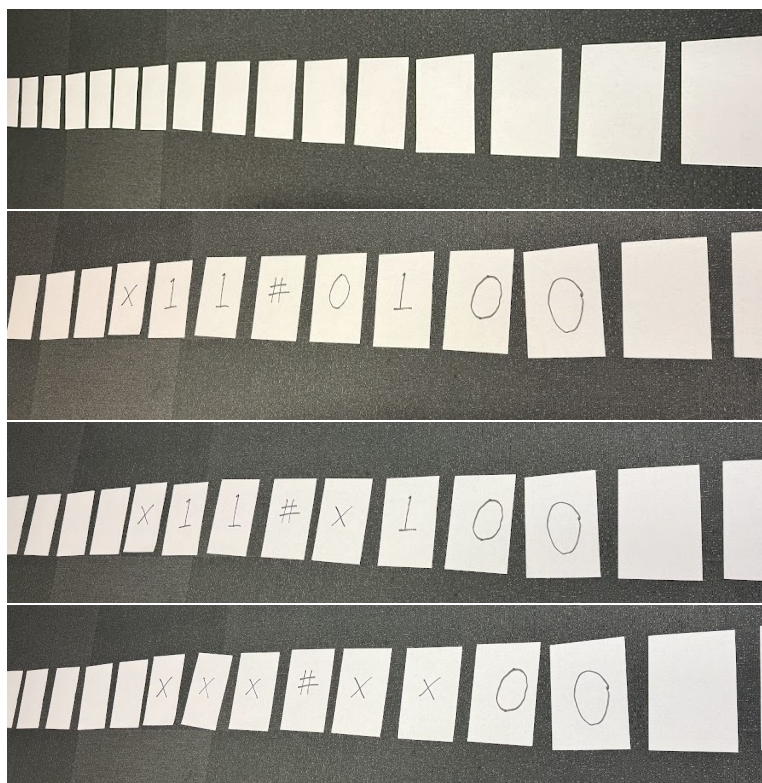


Figure 4: Turing machine tape contents at different stages of the activity: a. blank cells before adding input string 011#0100 to the tape; b. tape after first transition of reading zero, writing x, and moving right; c. tape after reading 0, writing x, and moving left; d. tape just before machine rejects.

two-dimensional grid.

Content: This activity gives students opportunities to think more generally about computation and Turing machine equivalence.

Materials: Code Master is a physical game, so the instructor would need to purchase a copy of it. Turing Tumble can be purchased as a physical game or it can be played via the simulator [24]. The Conway's Game of Life simulator can be found at [14]. If there are no computers in the classroom, the instructor should ask students to bring laptops to access the games.

Lesson: [50 minutes] Turing Tumble and Code Master are puzzle games with instruction booklets and sample solutions. Have students work together in groups to solve a couple of the puzzles for each game. To keep things simple,

the instructor of the ToC course chose certain puzzles for students to attempt. Each of these games takes about 15 minutes. Have students play Game of Life, starting the simulation with various patterns of live cells to see if they can get a pattern that repeats over time and a pattern where the cells eventually all die. Students can search for examples on the Web to find starter patterns for Game of Life to see how those simulations proceed. Experimenting with Game of Life takes about 15 minutes. Figure 5 (right) shows students working together on a puzzle for the Turing Tumble game. Figure 5 (left) shows students working together on a puzzle for the Code Master game. The instructor can do a class discussion about connections they see between Turing Machines and the games. Alternatively, the instructor can ask students to complete a reflection to make connections between at least one of the games and Turing Machines [25].



Figure 5: Left: Code Master Game Board; Right: Turing Tumble Game Board

3.7 Four Corners: Students Vote for Class

Purpose: This is a review game at the end of the semester. Students vote on the language class by moving to one of the four corners of the classroom [22]. The instructor can quickly see how the students perceive each language and if they can devise the proofs or proof ideas to defend their vote.

Content: This activity reviews computational classes: regular, context-free and not regular, decidable and not context-free, and undecidable.

Materials: The instructor should bring four pieces of paper and tape. Each sheet of paper should include one of the following classes: 1) Regular, 2) Context-Free and Not Regular, 3) Decidable and Not Context-Free, 4) Undecidable. Tape each sheet to the wall at each corner of the classroom. Also, bring blank sheets of paper and have a few at each corner of the room. The

instructor should also make a list of languages (a few from each of the four groups) to use in the activity. Each language can be projected on its own slide or provided to students as a list on a handout.

Lesson: [40+ minutes] The instructor may want to draw the concentric rings of regular languages, context-free languages, and decidable languages on the board to remind students of subset relationships (see Figure 6).



Figure 6: Language classes for four corners activity

Show students the language class labels at each corner of the room. Show the first language description to the students. Have students vote for the correct class by going to the appropriate corner. Once at their corner, that blob of students should collaborate to prove their choice is correct. If they choose Regular, they can build a DFA, NFA, or a regular expression. If they choose Context-Free and Not Regular, they can build a PDA or grammar and use the pumping lemma for regular languages to prove the language is not regular. If they choose Decidable and not Context-Free, they can build a Turing machine (algorithm or state machine) and use the pumping lemma for context-free languages to prove it is not context-free. If they choose Undecidable, they can build a reduction proof or use Rice's Theorem. Students may go to a different corner if they determine their proof techniques or machines are not working for that language and for their chosen class. If time is limited, students can discuss their justification approach instead of formulating machines and proofs. Here are some languages (all with alphabet $\{a, b\}$):

- $\{w \mid w \text{ has more a's than b's in any order}\}$
- $\{\langle M \rangle \mid M \text{ is a Turing machine description and } M \text{ accepts 'baa'}\}$
- $\{w \mid w \text{ has the substring 'baa'}\}$
- $\{a^n b^n a^n \mid n \geq 0\}$
- $\{w \mid w \text{ reads the same forward as it does in reverse}\}$
- $\{w \mid w \text{ starts and ends with the same character}\}$
- $\{\langle M, w \rangle \mid M \text{ is a Turing machine description and } M \text{ writes 'x' over 'a' when running on } w\}$
- $\{\langle M \rangle \mid M \text{ is a Turing machine description and } M \text{ has at least 10 states}\}$
- $\{w \mid w \text{ has length at least 10}\}$

4 Evaluation

The guiding question for this experience report is: 1) Did the in-class activities support student engagement and learning?.

The data for question 1 includes students' survey feedback and instructor observations from the Theory of Computation course. The study was approved by the University's Institutional Review Board.

There were 36 students enrolled in Theory of Computation in Fall 2023. Students were given a few extra credit points (worth less than 1% of the overall grade) if they completed the optional end-of-semester survey. Of the 36 students, 25 completed the optional survey.

4.1 Supporting Student Learning

The instructor observed that students were engaged in all seven activities described above. Students remained on task throughout the activities. Note that the poster walk and the four corners activities were designed to help students review course material prior to an exam, so they had some additional incentive to participate, learn from their peers, and ask questions. The physical movement around the room kept student energy on the higher side and reduced the power dynamic of instructor-at-the-front and students sitting.

On the end-of-semester survey, students rated how each the course materials supported their learning of ToC (1=not, 2=minimally, 3=somewhat, 4=mostly, 5=very much). There were seven in-class activities described in this paper, but the end-of-semester survey did not ask specifically about each of these activities, other than the Turing games activity. Table 1 shows the average scores (1 to 5) for each of the course activities. The in-class activities were rated the highest of all course learning aids. The Turing machine games activity was ranked on the lower-end of the activities. The instructor suspects that the Turing machine games activity did not have a direct skill or proof technique

Table 1: Average student scores for course materials supporting their learning (1=not, 5=very much)

Coursework	Average
In-class activities	4.6
Synchronous class time	4.4
Exams and studying for exams	4.4
Homework problem sets	4.3
Problem-solving groups	4.2
Programming project	4.1
Lecture videos (viewed before class)	4.0
Exam videos (problem explanation by video)	3.9
Turing machine games activity	3.8
Regular expression lab in python	3.5
Artwork	3.3
Friday quizzes	3.2

associated with it, so students may not have seen that activity as essential for their learning for a formal assessment, such as an exam.

Before students were shown the rating question on the survey, two open-ended questions asked what was most helpful and least helpful to their learning and why. Here are a few sample comments about the in-class activities:

- “I really enjoyed the board game lab that we did. Helped me visual a lot of the concepts.”
- “The in-class activities were helpful because they were just other examples that gave us practice and could get feedback or help right then and there.”
- “I would also say that the in-class activities would also be super useful (I even remember the TM activity that we did in class) - because it was fun, it actually made it super enjoyable to watch and learn from! Overall, I think activities that involve with interaction with other people is something that makes it more helpful for me to learn.”
- “The in-class activities were the most helpful for me. It provided me an opportunity to ask questions about things that confused me, and to recognize things I didn’t really understand (especially when going through problems with my table group).”

None of the 25 students mentioned the in-class activities as being the least helpful in the free response question. The textbook, videos, and quizzes were named as least helpful. Overall, the data suggests that the students found the in-class activities helpful for learning.

5 Conclusion

This experience report provides in-class activities to promote student movement in the classroom. Even though they were used in a single computer science course, the techniques can apply to other courses. For example, the poster walk can be used in any course to review material. In fact, the instructor used the poster activity in Computational Biology the next semester. Four corners can be used for any multiple-choice type question or other categorization where students might disagree. It works well when there is no definitive answer. The instructor has used four corners in Data Structures to have students choose one of four data structures for various tasks, even if there is no clear “best” choice. Blobs and lines can be used as an ice-breaker in any course. Students reported that the in-class activities helped their learning; in fact, it was the top-rated ToC course activity. The instructor asked how many classmates each person knew at the beginning and end of the semester. The range of known peers at the beginning of the semester was 0 to 12 with an average of 4.97. At the end of the semester, the range of known peers was 3 to 18 with an average of 10.00. One great outcome of using group learning activities was getting to interact with more of their peers.

References

- [1] Charles D. Allison. “Practical computation theory”. In: *J. Comput. Sci. Coll.* 23.1 (Oct. 2007), pp. 141–146. ISSN: 1937-4771.
- [2] Andrew Begel, Daniel D. Garcia, and Steven A. Wolfman. “Kinesthetic learning in the classroom”. In: *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*. SIGCSE '04. Norfolk, Virginia, USA: Association for Computing Machinery, 2004, pp. 183–184. ISBN: 1581137982. DOI: 10.1145/971300.971367. URL: <https://doi.org/10.1145/971300.971367>.
- [3] Rabranea Bqa, Tanzeela Shakeel, and Yaser Daanial khan. “A Pedagogical Approach towards Theory of Computation”. In: *Proceedings of the 2019 8th International Conference on Educational and Information Technology*. ICEIT 2019. Cambridge, United Kingdom: Association for Computing Machinery, 2019, pp. 192–197. ISBN: 9781450362672. DOI: 10.1145/3318396.3318430. URL: <https://doi.org/10.1145/3318396.3318430>.

- [4] Travis S. Crone and Mary C. Portillo. “Jigsaw Variations and Attitudes About Learning and the Self in Cognitive Psychology”. In: *Teaching of Psychology* 40.3 (2013), pp. 246–251. DOI: 10.1177/0098628313487451. eprint: <https://doi.org/10.1177/0098628313487451>. URL: <https://doi.org/10.1177/0098628313487451>.
- [5] Nasrin Dehbozorgi. “Active Learning Design Patterns for CS Education”. In: *Proceedings of the 2017 ACM Conference on International Computing Education Research*. ICER ’17. Tacoma, Washington, USA: Association for Computing Machinery, 2017, pp. 291–292. ISBN: 9781450349680. DOI: 10.1145/3105726.3105741. URL: <https://doi.org/10.1145/3105726.3105741>.
- [6] Louis Deslauriers et al. “Measuring actual learning versus feeling of learning in response to being actively engaged in the classroom”. In: *Proceedings of the National Academy of Sciences* 116.39 (2019), pp. 19251–19257. DOI: 10.1073/pnas.1821936116. eprint: <https://www.pnas.org/doi/pdf/10.1073/pnas.1821936116>. URL: <https://www.pnas.org/doi/abs/10.1073/pnas.1821936116>.
- [7] Ryan E. Dougherty. “Experiences Using Research Processes in an Undergraduate Theory of Computing Course”. In: *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*. SIGCSE 2024. Portland, OR, USA: Association for Computing Machinery, 2024, pp. 310–316. ISBN: 9798400704239. DOI: 10.1145/3626252.3630849. URL: <https://doi.org/10.1145/3626252.3630849>.
- [8] Christiane Frede and Maria Knobelsdorf. “Exploring how Students Perform in a Theory of Computation Course using Final Exam and Homework Assignments Data”. In: *Proceedings of the 2018 ACM Conference on International Computing Education Research*. ICER ’18. Espoo, Finland: Association for Computing Machinery, 2018, pp. 241–249. ISBN: 9781450356282. DOI: 10.1145/3230977.3230996. URL: <https://doi.org/10.1145/3230977.3230996>.
- [9] Scott Freeman et al. “Active learning increases student performance in science, engineering, and mathematics”. In: *Proceedings of the National Academy of Sciences* 111.23 (2014), pp. 8410–8415. DOI: 10.1073/pnas.1319030111. eprint: <https://www.pnas.org/doi/pdf/10.1073/pnas.1319030111>. URL: <https://www.pnas.org/doi/abs/10.1073/pnas.1319030111>.
- [10] Edward F. Gehringer and Carolyn S. Miller. “Student-generated active-learning exercises”. In: *Proceedings of the 40th ACM Technical Symposium on Computer Science Education*. SIGCSE ’09. Chattanooga, TN, USA: Association for Computing Machinery, 2009, pp. 81–85. ISBN:

9781605581835. DOI: 10.1145/1508865.1508897. URL: <https://doi.org/10.1145/1508865.1508897>.
- [11] Michael Goldweber. “Two kinesthetic learning activities: turing machines and basic computer organization”. In: *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education*. ITiCSE ’11. Darmstadt, Germany: Association for Computing Machinery, 2011, p. 335. ISBN: 9781450306973. DOI: 10.1145/1999747.1999847. URL: <https://doi.org/10.1145/1999747.1999847>.
 - [12] Michael T. Grinder et al. “Loving to learn theory: active learning modules for the theory of computing”. In: *SIGCSE Bull.* 34.1 (Feb. 2002), pp. 371–375. ISSN: 0097-8418. DOI: 10.1145/563517.563488. URL: <https://doi.org/10.1145/563517.563488>.
 - [13] Qiang Hao et al. “Effects of Active Learning Environments and Instructional Methods in Computer Science Education”. In: *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. SIGCSE ’18. Baltimore, Maryland, USA: Association for Computing Machinery, 2018, pp. 934–939. ISBN: 9781450351034. DOI: 10.1145/3159450.3159451. URL: <https://doi.org/10.1145/3159450.3159451>.
 - [14] Edwin Martin. *Game of Life*. 2024. URL: <https://playgameoflife.com/>.
 - [15] Mostafa Mohammed, Clifford A. Shaffer, and Susan H. Rodger. “Teaching Formal Languages with Visualizations and Auto-Graded Exercises”. In: *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. SIGCSE ’21. Virtual Event, USA: Association for Computing Machinery, 2021, pp. 569–575. ISBN: 9781450380621. DOI: 10.1145/3408877.3432398. URL: <https://doi.org/10.1145/3408877.3432398>.
 - [16] Ravensburger. *Code Master*. 2024. URL: <https://www.ravensburger.us/products/thinkfun/single-player-logic-games/code-master-44001950/index.html>.
 - [17] Matthew B. Robinson et al. “A Java-based tool for reasoning about models of computation through simulating finite automata and Turing machines”. In: *The Proceedings of the Thirtieth SIGCSE Technical Symposium on Computer Science Education*. SIGCSE ’99. New Orleans, Louisiana, USA: Association for Computing Machinery, 1999, pp. 105–109. ISBN: 1581130856. DOI: 10.1145/299649.299704. URL: <https://doi.org/10.1145/299649.299704>.

- [18] Susan H. Rodger, Jinghui Lim, and Stephen Reading. “Increasing interaction and support in the formal languages and automata theory course”. In: *Proceedings of the 12th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*. ITiCSE '07. Dundee, Scotland: Association for Computing Machinery, 2007, pp. 58–62. ISBN: 9781595936103. DOI: 10.1145/1268784.1268803. URL: <https://doi.org/10.1145/1268784.1268803>.
- [19] Scott Sigman. “Engaging students in formal language theory and theory of computation”. In: *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*. SIGCSE '07. Covington, Kentucky, USA: Association for Computing Machinery, 2007, pp. 450–453. ISBN: 1595933611. DOI: 10.1145/1227310.1227463. URL: <https://doi.org/10.1145/1227310.1227463>.
- [20] Upper Story. *Turing Tumble*. 2024. URL: <https://upperstory.com/en/turingtumble/>.
- [21] Collaborative for Teaching and Learning. *Gallery Walk*. 2024. URL: <https://ctlonline.org/gallery-walk/>.
- [22] The teacher toolkit. *Four Corners*. 2024. URL: <https://www.theteachertoolkit.com/index.php/tool/four-corners#:~:text=Students%20express%20their%20opinion%20or%20thinking%2C%20and%20decision%2Dmaking..>
- [23] Joe Tranquillo. “Kinesthetic Learning In The Classroom”. In: *2008 Annual Conference & Exposition*. 10.18260/1-2-3389. <https://peer.asee.org/3389>. Pittsburgh, Pennsylvania: ASEE Conferences, June 2008.
- [24] Richard Twilton. *Tumble Together*. 2024. URL: <https://tumble-together.herokuapp.com/>.
- [25] Tammy VanDeGrift. “Connecting Education and Fun: Using Turing Machine Games in Theory of Computation”. In: *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 2*. SIGCSETS 2025. Pittsburgh, PA, USA: Association for Computing Machinery, 2025, pp. 1645–1646. ISBN: 9798400705328. DOI: 10.1145/3641555.3705151. URL: <https://doi.org/10.1145/3641555.3705151>.
- [26] Rakesh M. Verma. “A visual and interactive automata theory course emphasizing breadth of automata”. In: *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*. ITiCSE '05. Caparica, Portugal: Association for Computing Machinery, 2005, pp. 325–329. ISBN: 1595930248. DOI: 10.1145/1067445.1067535. URL: <https://doi.org/10.1145/1067445.1067535>.

- [27] Michel Wermelinger and Artur Miguel Dias. “A prolog toolkit for formal languages and automata”. In: *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*. ITiCSE '05. Caparica, Portugal: Association for Computing Machinery, 2005, pp. 330–334. ISBN: 1595930248. DOI: 10.1145/1067445.1067536. URL: <https://doi.org/10.1145/1067445.1067536>.
- [28] Ursula Wolz, Michael Milazzo, and Meredith Stone. “Kinesthetic learning of computing via "off-beat" activities”. In: *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education*. ITiCSE '11. Darmstadt, Germany: Association for Computing Machinery, 2011, pp. 68–72. ISBN: 9781450306973. DOI: 10.1145/1999747.1999769. URL: <https://doi.org/10.1145/1999747.1999769>.

Exploring Data Mining Algorithms and Ethics Via a News Journal*

Kevin Treu
Department of Computer Science
Furman University
Greenville, SC, USA
kevin.treu@furman.edu

Abstract

This paper describes an ongoing experiment with engaging students in current applications of data mining while they are studying the subject in an introductory class. This semester-long Data Mining News Journal activity is integrated into the course curriculum and has been effective in encouraging students to engage with and reflect upon the real world applications of the algorithms they are learning. Experience has shown that this reflection enhances not only technical understanding of the underlying algorithms, but also awareness and appreciation of several ethical issues studied in class. Students are given autonomy to discover their own contemporary examples of data mining in the news, and to relate their selected examples to the class material in multiple ways. With each iteration of the Journal, the course instructor identifies a theme and facilitates a class discussion driven by student reflections. Preliminary results suggest that student self-motivation for their own learning and career pathway objectives are enhanced.

Keywords: data mining, machine learning, news journal, student autonomy, metacognition

*Copyright ©2026 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

1 Introduction

The undergraduate Computer Science curriculum has experienced a rapid growth of new courses and curricular initiatives relating to data analytics in recent years. At Furman University this growth includes a very highly enrolled interdisciplinary minor in data analytics, along with courses at both the introductory and advanced levels. The benefits of studying data analytics are evident to students not only of Computer Science, but from a wide variety of disciplines. A solid background in data mining tools and algorithms enhances critical thinking and problem solving skills in programs of study from Biology to Economics to Philosophy. Such a foundation has also proven to be valuable to long term career pathways. Consequently, not only are such courses and programs very popular, but filled with potential for tailoring the learning experience to the specific interests and objectives of students.

At the same time, pedagogy research in teaching and learning has identified numerous advantages of utilizing metacognitive activities in the classroom. These reflective tasks have the objective of improving “the awareness or understanding of one’s own cognitive processes or acquired knowledge” [5]. Such advantages include identification of particular affinities or connections that students have with the course content, and a higher degree of student autonomy in the learning process. These in turn result in increased engagement, comprehension, and self-motivation [2]. As with the popularity of data analytics, this trend is multidisciplinary, with metacognitive exercises having been employed successfully in a variety of fields [9, 13, 1].

There is a clear opportunity, it seems, to help the burgeoning number of students of data analytics from all fields enhance their experience and reach their learning and career objectives by employing such reflective practices. Moreover, the ubiquitous nature of data and predictive analytics in society means that ample resources are available for such exercises. In Furman’s *Introduction to Data Mining*, a course designed to be accessible to students of all majors, a semester-long Data Mining News Journal assignment has been a successful implementation of metacognition. Specifically, it has given students a degree of control over their learning by being able to research data mining applications that align with their interests, has provided a meaningful context for course concepts such as algorithms and ethics, has generated meaningful discussion with diverse perspectives represented, and generally has been effective in enhancing engagement and helping students appreciate the value of what they are learning. Furthermore, the activity has provided the instructor/author with insight as to current topics of interest to the students which provide inspiration for new mini-lecture material.

2 Course Background

The course *Introduction to Data Mining* at Furman is 11 years old, having been offered for the first time in the spring semester of 2015. It has several characteristics that make it relatively unique. Most importantly, it is designed for an audience of undergraduate students with any disciplinary background or major. Thus it is imperative that it be accessible to students with a minimum amount of computing experience. The only pre-requisite is CS1, which ensures a basic level of experience with algorithmic thinking and Python coding.

To enhance accessibility, however, it is not a programming class. Various data mining algorithms are studied and practiced with pencil and paper, and implemented via the Weka software application [11]. This allows focus to be maintained on the details of the algorithms and avoids the pitfalls that can come from struggles with programming. This follows the implementation of Dr. David Sullivan’s non-major introductory CS course at Boston University [10], to whom the author owes a great debt.

At Furman, this course satisfies the *Human Behavior* (HB) general education requirement. It shares this designation primarily with courses in the social sciences. As such, it strongly emphasizes empirical methods specifically used to describe and understand the behavior of both individuals and groups, and to predict that behavior as effectively as possible. These methods take the form of data mining algorithms. This GER status provides an additional incentive for students to enroll in the class, and thus also contributes to the multidisciplinary nature and varied experience levels of the audience. At Furman University, a small liberal arts institution with approximately 2,700 undergraduate students, this course typically enrolls 24-28 students per section, with three sections typically offered each academic year.

The course material is organized around the following topics:

- Finding, assembling, and preparing data for the mining process
- Understanding basic data mining strategies and algorithms and learning how to apply them appropriately
- Recognizing how knowledge of important concepts can be represented and communicated effectively via different types of models and visualizations
- Evaluating the accuracy and effectiveness of models resulting from data mining
- Applying the results of an accurate model to real-world problems
- Consideration of the ethical issues raised by data mining technology

For purposes of this report, the focus is on algorithms and ethical issues, and maximizing comprehension and critical thinking involving those topics. Algorithms are organized by whether they implement supervised or unsupervised

learning strategies. Students thus study algorithms for supervised classification and numerical estimation, and algorithms for unsupervised association learning and clustering. Details of the specific algorithms can be found in the textbook *Data Mining: Practical Machine Learning Tools and Techniques* [12].

Ethical issues of data mining are organized into the following broad categories [4]:

- Unfair discrimination
- Reinforcing human bias
- Lack of transparency
- Privacy
- Consent and power

Examples, case studies, and assignments used in the class are regularly assessed in light of these potential ethical concerns via discussion and reflection. They therefore are a significant focus.

3 Data Mining News Journal

Understanding *how* various data mining algorithms work and *why* they work in appropriate situations are of primary learning outcomes of the class, of course. With an audience of students having diverse academic backgrounds and objectives, this can be quite a challenge. Computer Science majors work and study alongside students who do not plan to take any further courses in CS. Learning styles and affinity for specific topics can vary greatly. Something similar is also true of the ethics topics. Though not a technical endeavor, students clearly differ in their passion regarding these different concerns. The Data Mining News Journal activity provides a means for engaging all students at their own areas of interest, with reflective prompts designed to enhance and deepen comprehension.

3.1 Expectations of Students

The meta- (or “higher state of”) cognition sought with this activity is a student’s appreciation of how the methods that they are learning in class are being applied to meaningful effect in the real world, and how the ethical implications align with the categories that they have also studied. Ideally, this prompts heightened interest in, and attention to the material that is being studied in a more conventional manner. An additional objective is to encourage students to maintain a habit of regularly consulting authoritative periodicals to stay abreast of developments in their field that have an impact on society – to become lifelong scholars, in other words.

The News Journal provides a way to relate the technical details of data mining and machine learning algorithms to real world applications involving human behavior. The activity emphasizes how such algorithms can have a profound impact in terms of not only *predicting* what humans will do, but gaining a deeper *understanding* of their behavior.

The activity is a standing weekly assignment to find one article or current event concerning data mining/predictive analytics. The key criteria for the selected stories are *currency*, *authority*, and *relevance*. That is, they must describe current research or application of data mining, should come from a respected, vetted source, and be accessible and of interest to students of data mining. In accordance with the GER, all selections have to be related to some aspect of human behavior.

In service to student autonomy, a wide spectrum of sources is permitted, though students are required to vary them. Peer-reviewed academic publications are not required. Indeed, for the sake of accessibility other types of sources are encouraged. These include national and even local newspapers, as well as periodicals such as *Time*, *Newsweek*, *Science*, and *Scientific American*. Students are encouraged to visit web sites devoted to data mining and the broader topic of data journalism, such as FlowingData and KDNuggets for news stories, and the Kaggle competitions page for examples of data mining problems being solved currently [3, 7, 6]. Podcasts devoted to data analytics are also encouraged.

Given the wealth of sources for current events in data mining, students are not permitted to repeat the same source in consecutive weeks. Effort beyond going back to a trusted data journalism site and using the first article listed is expected, as discernment and critical thinking are an important part of the project.

In terms of logistics, a new entry to each student's journal is due at 2pm every Thursday afternoon. Entries are posted to the *News Journal* forum on the course Moodle page, and must include (1) a link to the selected story, (2) a 100-150 word summary/explanation of the story, and (3) a 100-150 word reflection. Specific news stories may not be repeated, so it is important that students read the submissions of others.

While the News Journal is integrated throughout the semester alongside the study of specific algorithms, the discussion themes are intentionally emergent rather than predetermined. The course textbook provides comprehensive coverage of data mining algorithms and techniques, but does not address contemporary applications or ethical implications in meaningful detail. The News Journal fills this gap by grounding abstract algorithms in current real-world contexts and surfacing ethical issues as they arise in practice. After reviewing each week's submissions, the instructor identifies patterns and con-

nections that have organically appeared in student selections. This responsive approach ensures themes are grounded in authentic student interests rather than imposed topics. However, certain themes recur frequently enough across semesters to become regular touchpoints—for example, early themes typically focus on beneficial applications before shifting attention to ethical concerns and the responsibilities that accompany such powerful technologies.

Reflections are the first critical component of the metacognitive process. Each reflection is expected to answer one or more of these questions:

- How does the story affirm or amplify something we have learned in class? Or refute something?
- What impact is the application having, or might it have, on society? What does it stand to teach – or predict – about human behavior?
- Are there any interesting innovations to describe, or possibly even flaws that you find questionable?
- In what other problem domains might the application be fruitfully applied?
- What makes it interesting to you?

These reflections require students to monitor their own understanding by identifying which concepts they grasp well enough to connect to real-world applications. Furthermore, each reflection should have the following qualities:

- It should be in your own words. That is, you should select a story that engages and interests you, so that you are able to describe its contents without using phrases or complex terminology copied from the story itself.
- It should go beyond the “what?” question and also discuss the “how?” As in: “How is the data mining accomplished?” in your story. What does the dataset look like (attributes, instances, class attribute)? What methodology (data collection, data cleaning, machine learning technique, evaluation, etc.) is used? Relate the story directly to things we are learning in class.

Class discussion is the second critical metacognitive component. In class on the Friday after a News Journal deadline, the discussion is organized around key themes that emerge from the group of submissions, collectively. A component of the grade for the assignment is active student participation in this discussion. Particularly interesting or well reported stories are sometimes highlighted by a short presentation in class. Similarly, selected stories may be used to help make a larger point in support of a theme that emerges.

3.2 Expectations of Instructor

A surprisingly beneficial – and also intimidating – aspect of this activity is what is expected of the instructor in terms of assessment and facilitation of the weekly discussion. After the deadline on Thursday at 2pm, all News Journal submissions must be read and evaluated according to the assignment criteria. More importantly, a weekly theme is identified.

Certain themes can be broad and easily crafted to fit almost any collection of submissions. The author typically begins the semester with “Positive benefits of data mining” and “With power comes responsibility” as the first two themes. The former focuses on benefits to individuals and society as a whole from effective data analytics in the news. The latter then turns the students’ attention to ethical issues raised by their journal selections, the first time this is covered in a formal way in the class. Mini-lectures guide the discussion, with current News Journal articles selected to make key points.

Subsequent themes can sometimes be repeated from semester to semester, but often have to be improvised prior to the class meeting. Identifying coherent themes can be challenging when student submissions span diverse domains. In such cases, the instructor may choose a broader organizing principle (such as “unexpected applications”) or focus on what the mining processes have in common even when the application domains differ. Over time, experience with recurring topics has made theme identification more efficient. Some examples of themes guided by student discoveries have contributed meaningfully to the author’s teaching materials and are given in the next section.

4 Discussion

A decade of experience implementing this assignment has allowed several benefits to be observed, as well as identified potential pitfalls.

4.1 Themes

As noted previously, when students follow their affinities for both the course material and their major field of study, the instructor is able to discern what they are passionate about. Viewed collectively, this allows themes to emerge. Indeed, determination of a weekly theme for mini-lecture and discussion is a primary objective of the instructor, typically requiring some improvisation.

In addition to the broad themes of positive impact and responsible use of data mining already mentioned, the News Journal makes it easy to reflect on several other commonly occurring themes in a natural way. These include, for example:

- Personalization – how data mining can impact individual lives

- Correlation is not causation
- A little goes a long way – even small improvement in predictive accuracy can have a significant impact
- Sports analytics
- Health prediction
- Predicting/preventing crime or predicting recidivism/making sentencing decisions
- Data mining impacting hobbies/recreation

Examples of themes such as these are ubiquitous enough that they can be effectively used for reflection during each iteration of the class. This has been very effective in reinforcing algorithm details and ethical issues previously raised in class.

Other interesting themes arise less frequently, but are worth exploring when they do. Examples include:

- Data mining in admissions and hiring
- Flight risk – predicting retention of students or employees
- All data is predictive – effective mining of datasets not compiled for that purpose
- Sometimes it is better not to know – situations in which what is learned/predicted from data mining is harmful or destructive

Another category of themes involves observations of specific algorithms from class being implemented in real world examples. For example, we cover the basic 1R algorithm for determining classification rules in datasets where one attribute is sufficient for accurate prediction. Occasionally news stories will highlight such datasets. Other algorithms such as Naïve Bayes and k-Means are occasionally mentioned explicitly in current stories as well.

The most rewarding examples are those that have led to permanent modules in the class – including lecture material, lab activities, and assignments. These are:

- Sentiment analysis
- Social media data mining
- Text mining
- Ensemble learning (metalearning) algorithms

In each of these cases, experience with the News Journal has demonstrated levels of student interest and frequency of mention high enough to motivate the author to develop additional supporting materials.

4.2 Enhanced Engagement

The Data Mining News Journal has been effective in drawing a connection between the big, exciting things that are happening with data analytics “out there” in the “real world” and what the students are learning “in here” in the university “bubble”. This connection encompasses not only the technical details of algorithms for analytics, but the big ethical questions that are raised – and that typically make new advances newsworthy. It also motivates students to become invested in the class itself.

The autonomous nature of the exercise also contributes to this self-motivation. The key is helping students discern their own passions and encouraging them to pursue them in a way that engages directly with the course material. [9] Students often ask how course material is specifically relevant or beneficial to their personal and professional pathways, and a clear answer is evident when something they have found that is of particular interest to them is directly related to the class. When students have a sense of ownership in their own learning process, such intrinsic motivation is often a consequence. [2]

4.3 Enhanced Learning

The publicly posted News Journal summaries and reflections represent students explaining to the others their data mining example and how it connects to the class. They also have the opportunity (optionally) to do this in person in the follow up class discussion. When questions about new results are asked and answered, and subsequently explained to others, real learning occurs most effectively. [8]

Moreover, the encouragement to find connections between class topics and current results in the field enhances the learning that is taking place. This occurs through increased engagement primarily, with the additional aspect that the instructor is able to preview the summaries and reflections prior to class discussion. This enables an assessment of their true understanding and progress, and also of what they find both interesting and confusing. This information is invaluable in the planning of mini-lectures and discussion facilitation.

4.4 Issues

Generally, the Data Mining News Journal is recognized for its benefits by the students. However, there are persistent issues to consider and address if possible. Procrastination is an issue, of course. In this case, students who wait until the last minute sometimes fail to read over the reflections of their classmates and repeated stories can occur.

Students sometimes have to be discouraged from adhering to their affinities too closely – that is, by repeating stories from the same application area too often. Sports analytics is a common example of this.

Similarly, there can be a tendency to return each week to the same source for the next article. It is not uncommon, for example, for certain resources to have to be banned for a week.

A clear distinction between *hidden* and *shallow* knowledge is made in the class – with the former being the outcome of data mining and the latter achievable via a database query. The News Journal has been an effective means of revealing (and correcting) student misunderstanding of this distinction. An important lesson is that articles about data are not necessarily about data *mining*.

5 Conclusion

The study of data analytics at an introductory undergraduate level can potentially devolve into a certain form of rote memorization of key algorithms, processes, and ethical principles. Encouraging students to identify and follow their own affinities and learn about their own learning via metacognitive activities helps to make the study instead an exploration of concepts, theories, and connections in which the students are invested and engaged.

A topic as contemporary, multidisciplinary, and in demand as data analytics appears to be a perfect candidate for the application of metacognitive practices that have shown to be effective in other domains. A Data Mining News Journal is one implementation of such a practice that has shown promise in facilitating the teaching and learning objectives of the class.

References

- [1] Kay Colthorpe et al. “Prompting undergraduate students’ metacognition of learning: implementing ‘meta-learning’ assessment tasks in the biomedical sciences”. In: *Assessment & Evaluation in Higher Education* 43.2 (2018), pp. 272–285.
- [2] Edward L. Deci and Richard M. Ryan. “The “what” and “why” of goal pursuits: Human needs and the self-determination of behavior”. In: *Psychological inquiry* 11.4 (2000), pp. 227–268.
- [3] *flowingdata.com*. Accessed: 2025-09-13. 2025. URL: [https : / / flowingdata.com/](https://flowingdata.com/).

- [4] Futurelearn.com. *Ethical questions in data science*. Accessed: 2026-01-24. 2021. URL: <https://www.futurelearn.com/info/courses/key-topics-in-digital-transformation/0/steps/261259>.
- [5] Lina Gassner. *Developing metacognitive awareness – a modified model of a PBL-tutorial*. Accessed: 2024-07-11. 2009. URL: <http://hdl.handle.net/2043/108880>.
- [6] Kaggle.com Competition page. Accessed: 2026-01-07. 2026. URL: <http://www.kaggle.com/competitions>.
- [7] KDNuggets.com. Accessed: 2026-01-07. 2026. URL: <http://kdnuggets.com>.
- [8] Stephanie McGuire, Saundra Yancy McGuire, and Thomas Angelo. *Teach students how to learn: Strategies you can incorporate into any course to improve student metacognition, study skills, and motivation*. Routledge, 2015.
- [9] Buket Oztas. “Metacognitive Exercises in the Political Science Classroom: Reflections and Research on Student Investment in Learning”. In: *The Palgrave Handbook of Teaching and Research in Political Science*. Cham: Springer International Publishing, 2023, pp. 35–46.
- [10] David G. Sullivan. “A data-centric introduction to computer science for non-majors”. In: *SIGCSE '13: Proceeding of the 44th ACM technical symposium on Computer science education*. 2013, pp. 71–76.
- [11] Weka Wiki. Accessed: 2024-07-17. 2024. URL: <https://waikato.github.io/weka-wiki/>.
- [12] Ian H. Witten et al. *Data Mining: Practical Machine Learning Tools and Techniques*. 4th. Waltham, MA: Morgan Kaufmann, 2017.
- [13] Anat Zohar and Sarit Barzilai. “A review of research on metacognition in science education: Current and future directions”. In: *Studies in Science education* 49.2 (2013), pp. 121–169.

Student Scheduling Patterns and Associated Outcomes for Computer Science Courses in a High Utilization Testing Facility *

Kelly Downey and Paea LePendu
Computer Science and Engineering
University of California, Riverside
Riverside, CA 92521
{kldowney, paealp}@ucr.edu

Abstract

Rampant cheating and enrollment growth have driven the establishment of testing centers at large universities, like University of California, Riverside, where new assessment strategies are being explored. Since creating ours in 2023, use has grown from 8 to 27 courses and 5,550 to 8,015 exams. Yet, there remains a lack of research on these types of testing centers. To address this, we observed four different types of computer science courses, both undergraduate and graduate level. We found that scheduling patterns form a universal and distinctive pattern. Students tend to schedule exams as late as possible with the greatest amount of students scheduling late in the middle of the quarter. Yet, nearly every student who receives a grade of C or lower is a late test taker.

Initially, we hoped to gather data to help optimize seating capacity, and we have by adjusting hours according to peak usage patterns. However, these results provide new insights pedagogically as well. We can dispel widespread concerns about collusion or assumptions that late test takers have an upper hand, something faculty worry about with flexible

*Copyright ©2026 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

testing. We can also identify at-risk students and provide early intervention. Ultimately, our goal is to share insight about testing centers at large universities and to support goals like mastery learning, flexible testing, and multiple chance exams given ever expanding enrollments.

1 Introduction

Research shows frequent and flexible testing modalities enhance student learning outcomes, which has led to the adoption of in-person computer-based testing facilities (CBTF). Research indicates that smaller, low-stakes exams given at more frequent intervals promote better learning and provide flexible options for a wider range of students, such as commuters [1][4]. However, traditional exams administered in high-density classrooms present significant logistical and security challenges. The physical layout of a populated classroom creates numerous visual obstacles, making it difficult for proctors to maintain clear sightlines and increasing the risk of undetected in-person cheating [9][12]. These problems compound when courses shift toward a frequent, low-stakes testing model; the administrative burden scales linearly, requiring instructors and teaching assistants to devote excessive time to manual proctoring and grading [8][15]. Furthermore, high-frequency traditional testing exacerbates scheduling conflicts for students and results in the unsustainable consumption of paper and other physical resources [11].



Figure 1: In-person computer-based testing facility

A viable solution to optimize the advantages of frequent and flexible testing is auto-graded digital exams[14] proctored in a controlled, CBTF. These facilities, as discussed by Ziles [10] and Downey[6] and pictured in Figure 1, provide trained proctors, adaptable scheduling options, managed devices, and the option to use traditional paper-and-pencil exams. With increasing enrollment numbers, the primary benefit driving increased adoption of digital assessments is the ease of auto-grading. Furthermore, the reduction of costs and ecological impact associated with paper exams also represent a significant benefit[16].

To maintain fairness and uphold the integrity of digital exams, proctoring is essential. The importance of proctoring any exam (digital or paper) has been widely recognized[2][5][7]. However, administering proctored digital exams comes with unique challenges. Online proctoring, where students are allowed to use their own device and choose the location to take the exam, raises concerns about student privacy and the potential for inequity[3]. On the other hand, in-person proctoring during class time, where students use their own personal laptops, has its own challenges: availability of sufficient power outlets, appropriate sized desks, and accommodations for those unable to attend class [1].

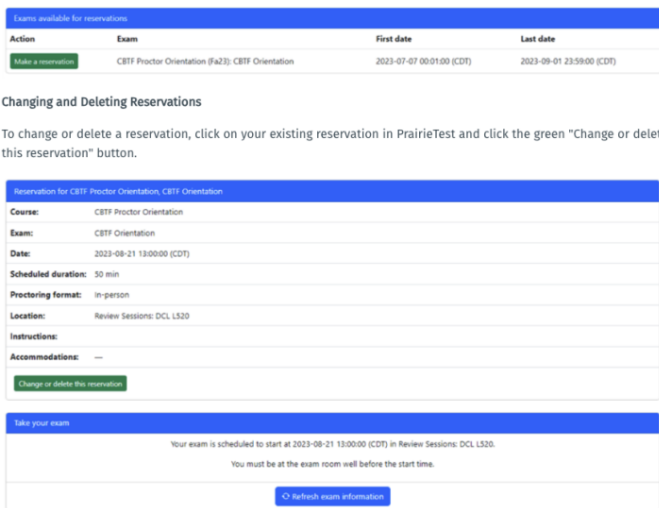


Figure 2: Student view of test reservation system

To facilitate proctoring to a dedicated test center, instructors define an exam window by setting a start and end date [14]. This allows students the flexibility to select an exam time that best suits their schedule within that period. Additionally, a registration period could also be set to ensure that

reservations are made in a timely manner. Ideally, the testing center would use an appointment ticketing system enabling students to book their preferred time. An example of such a system can be seen in Figure 2.

As with any shared resource, time and space are limited. Each facility will have a fixed number of testing spaces available each hour on days it is open and staffed. This means that students not only have to navigate the testing schedule and center’s operating hours but also the availability of seats. To make the best use of the resources available, administrators should optimize the allocation of operating hours to when students are expected to make reservations.

Previous work has examined scheduling habits to determine preferred timeslots and course performance correlation. Zilles et al[13] found that most students will schedule later in the test window. However, they did not study related outcomes. Chen et al [2] found that average student performance drops the later they take an exam, but were unsure if the pattern would be the same at other institutions.

Our work confirms these results at a large, R1, Hispanic Serving Institution and goes one step further to show the patterns and outcomes hold true in four different types of CS courses. In this paper, we will present the scheduling habits for students observed by our test center for CS courses of different levels. We looked at introductory courses, a mid-level course, and a graduate course. Our goal is to share knowledge and results about testing centers so instructors and administrators gain insight into the pedagogical needs of students while still addressing practical issues like staffing hours and cheating prevention.

2 Scheduling Behavior of Students

Our first goal was to study the scheduling habits of students who took an exam with a flexible scheduling window in our university’s CBTF. We used the logs generated when students made reservations in Prairetest (from the makers of the Prairie Learn) [10] to infer scheduling habits for four CS courses. We selected these courses because they each offered multiple exams during the term, used a multi-day flexible testing window, and represented a variety of different types of courses, both undergraduate and graduate. Across the four different types of courses, we extracted 3,395 reservations.

For each exam in each course, we grouped students into “t-minus” groups where “t” represents the day the exam is due (the last day) and “minus” is the number of days prior. So, “t-2” means two days before that last day. Students who did not take the exam during the exam window were placed in the N/A group. Students who are in groups t and t-1 are late test takers while students in group t-3 and t-4 are early test takers

2.1 Scheduling Behavior of Students in a CS1 Level Course

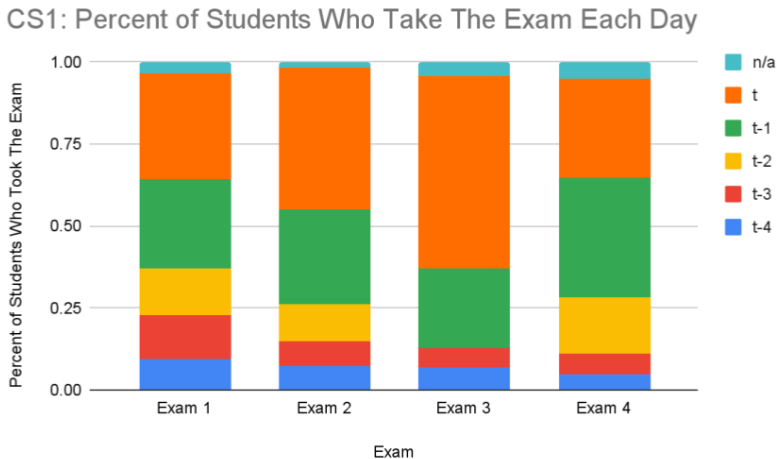


Figure 3: The scheduling habits of students in a CS1 level class

Figure 3 illustrates the distribution of students across four exams in a CS1 level course administered throughout the term. A significant majority of students, over 60%, belong to the late test-taking group. Notably, group *t* (represented in orange), which represents the final day of the exam, accounts for an average of 41% of students. This highlights the challenge of accommodating all students in larger courses within a single day, emphasizing the necessity for effective scheduling and planning for seat allocations.

One possible confounding factor was holidays. The data in Figure 1 overlaps with two holidays occurring that quarter, which may have influenced the scheduling choices of some students. For example, zero students took Exam 3 in group *t-2* because the facility was closed on that day to observe the holiday. Exam 4 was holiday-adjacent, which may affect decisions, but there were no closures. With common-sense adjustments, it does not seem like the overall patterns were disrupted by much: students worked around them $\pm$ one day. Late test takers did not suddenly become early test takers.

The pronounced dip in early *t*-groups like in later exams, like *t-3* in Exam 3, indicates that the number of late test takers tends to increase for each consecutive exam until it swings back later, following a “soup ladle” pattern. This soup ladle pattern or dip for group *t* in the second part of the quarter was universally present in all classes observed, not just CS 1. This means that

as the course progresses later in the quarter (presumably introducing more challenging material), students are waiting longer to take the exam.

In terms of forecasting seating distribution requirements, this strong, universal soup ladle pattern clearly means that staff should plan for more seats on days when exams are due and increasingly fewer on days prior.

2.2 Scheduling Behavior of Students in a CS2 and Later CS Course

The soup ladle reservation pattern holds for two other courses: a CS2 level course and a Later CS course. See Figure 4 and Figure 5.

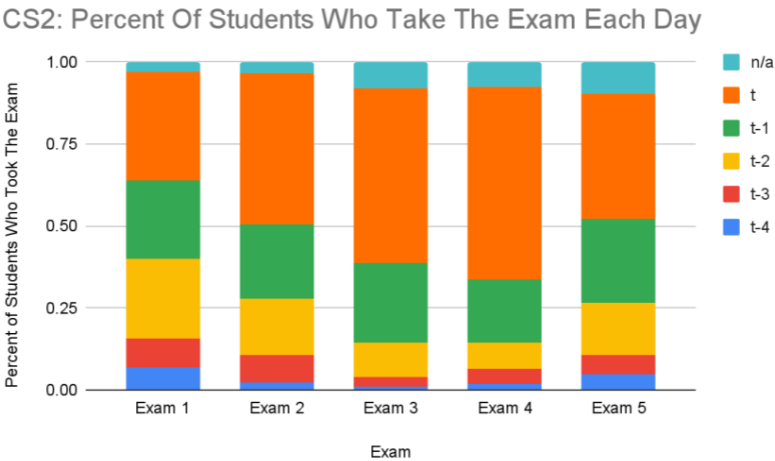


Figure 4: The scheduling habits of students in a CS2 level class

The CS2 level course immediately follows the CS1 level course. Unlike CS1, nearly all students in the CS2 course would have previously experienced using the test center. This experience has not affected their scheduling behaviors. We still see the increasing number of late test takers in consecutive exams with the soup-ladle pattern at the end.

The Later CS course answers the question of whether the pattern changes with students with more academic experience beyond the introductory courses with a resounding no. The Later CS course is usually taken by students in their second year and is not part of the introductory series. From Figure 5, we note the same behavior with late test takers (t or t-1) comprising over 60% of the class. On average, 45% of students are in group t. Again, we also note the strong soup ladle pattern.

Later CS: Percent of Students Who Take The Exam Each Day

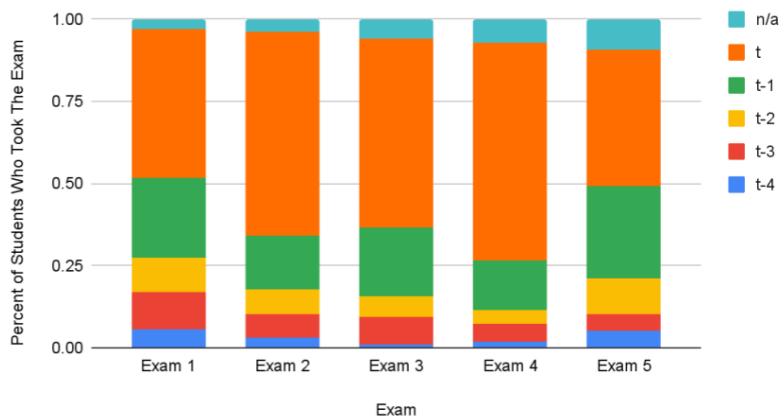


Figure 5: The scheduling habits of students in a later CS class

Of note, these two courses happened to be synchronized, meaning their exams were held in the same weeks. Also, unlike with CS1, they did not coincide with holidays. In fact, they were on different weeks from CS1 altogether. This confirms that holidays seem to have marginal, if any, effects on scheduling patterns.

2.3 Scheduling Behavior of Students in a Grad Level Course with a Shorter Exam Window

The three classes presented above had a testing window of five days and were all undergraduate level. Figure 5 shows the scheduling pattern for a grad-level CS course with a two-day testing window. We see similar patterns.

In the graduate level course, students were allowed to schedule their exam for a 50 minute slot during the eight hours the test center was open on exam day one or day two. We assigned four testing groups by dividing each day in half, four hours in each t group. So, the earliest test takers, $t-3$, took the exam on day one within the first half, and $t-2$ in the second half. On the second day, $t-1$ took the exam in the first half, and t in the second half.

Figure 6 shows that, even with a shorter test window, most students (again $>60\%$) choose to take the exam late and the largest percentage waits until the last possible window. Notably, this had little to do with when students logged in to register for their slot. In the graduate course, all students registered at

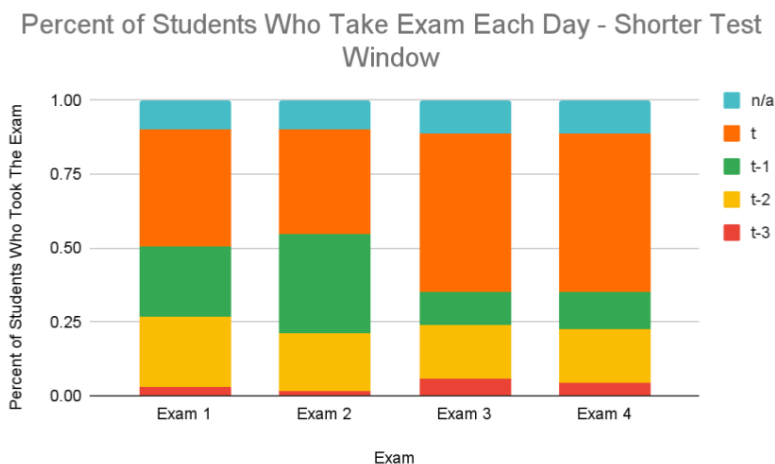


Figure 6: The scheduling habits of students in a grad- level CS course with a two day testing window

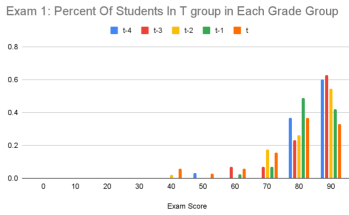
least one day before the window even began (the professor was diligent about checking and reminding students to register early).

3 Exam Scores and Scheduling Behavior

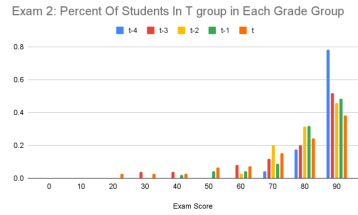
When we associate exam scores with t-group scheduling behaviors for each exam, we notice a pattern that suggests late exam takers are doing worse. Later, we also examine the effects on overall course grade. We only examined grades for undergraduate courses since students in the graduate course mostly got A's.

Students were placed into one of ten grade groups (normalized percent deciles) based on the score they achieved on a given exam. The same students were assigned a t-group as described earlier: t and t-1 are late exam takers, while t-3, and t-4 are early exam takers. Figure 7 plots the percent of students in each t-group that are in each grade group for Exam 1 through Exam 4 in the CS1 level course. Similarly, Figure 8 plots the same data for Exams 1 through Exam 5 in the CS2 level course.

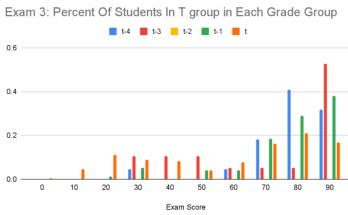
The plots show that the bulk of students with lower grades (0-30) are from groups t, t-1 (the late test takers) and t-2. The plots also show that the early test takers, t-4 and t-3, often had the best scores, in the 80s and 90s. In other words, averaged per exam, the early test takers are getting the A's and B's,



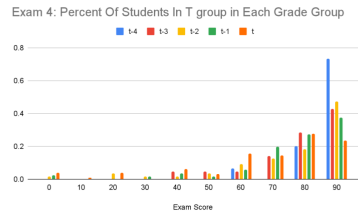
(a) CS1 Exam 1



(b) CS1 Exam 2

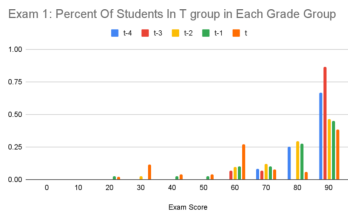


(c) CS1 Exam 3

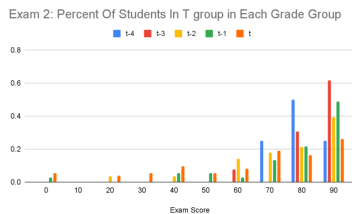


(d) CS1 Exam 4

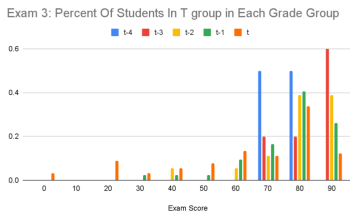
Figure 7: Percent of students in each T group in each grade group for CS1 exams



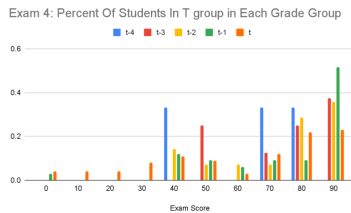
(a) CS2 Exam 1



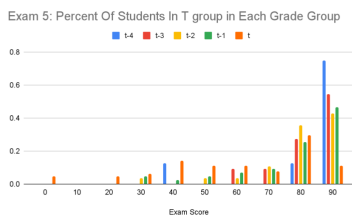
(b) CS2 Exam 2



(c) CS2 Exam 3



(d) CS2 Exam 4



(e) CS2 Exam 5

Figure 8: Percent of students in each T group in each grade group for CS2 exams

while the late test takers are getting the D's and F's.

4 Course Grade and Scheduling Behavior

As with the average per exam, the overall course grade is also correlated with the exam scheduling behavior. Early test takers get better grades, while late test takers get worse grades.

To better understand the overall results, we assigned a point value based on scheduling to each student and created a graph that compares these points to their course grade. Each t-group received a point value of 0 to 5, with the earliest group earning the highest score of 5. Specifically, group n/a receives 0 points, group t gets 1 point, t-1 gets 2 points, and so on. Students accumulate scheduling points by adding the points for all exams they scheduled for that course. For example, if a student consistently schedules in group t (which is worth one point each time), they would accumulate five points over five exams. In contrast, a student who regularly schedules early in group t-4 (worth five points each) would accumulate 25 points.

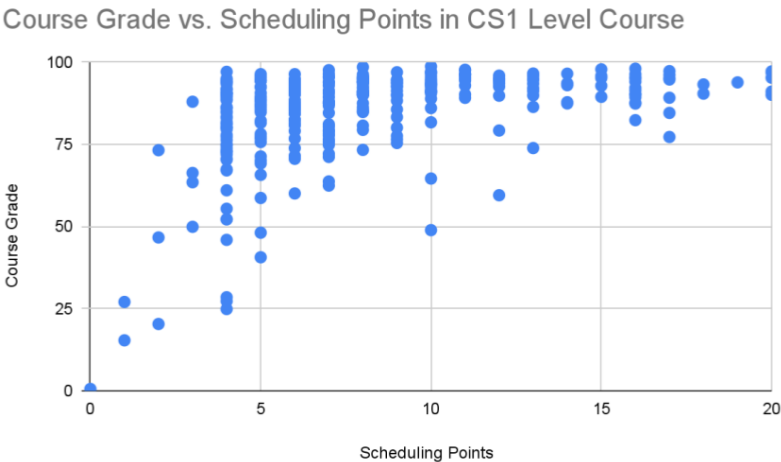


Figure 9: CS1 students scheduling points versus their overall course score

Figure 9 plots the scheduling points value versus the students' final overall course grade for the CS1 course. It illustrates that students with higher scheduling points are significantly less likely to receive a non-passing grade. In fact, not a single student who had >15 scheduling points received a course grade $<75\%$ (less than C). Although students who have lower point totals can

also have a passing score, lower point totals form the bulk of non-passing scores. That is, late test takers are more likely to not pass the class.

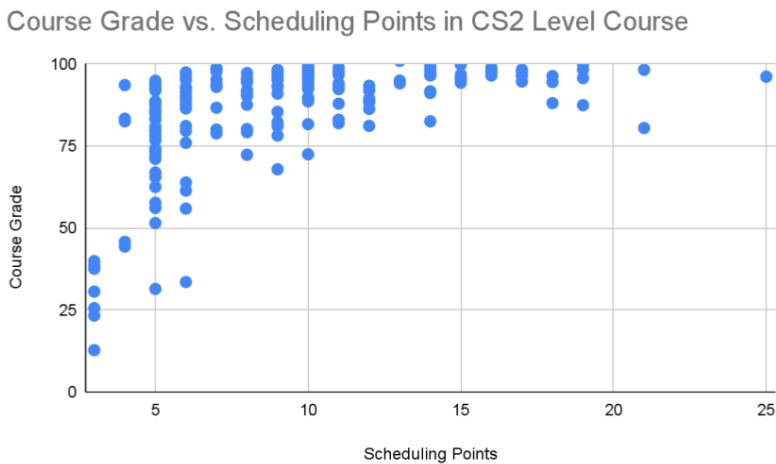


Figure 10: CS2 students scheduling points versus their overall course score

Figure 10 shows similar results for the CS2 level course. Again, the graduate course was not analyzed in this way given its highly skewed grade distribution.

5 Conclusion and Future Work

We see that students tend to schedule late in the test window with even more late test-taking behavior observed in the middle of the term than the beginning and end. The worst performers are habitually late test takers. This runs contrary to some fears that late test takers will do better because they will collude and get hints in advance.

These findings both reinforce and extend prior research conducted at other types of institutions. Our institution is an R1, Hispanic-Serving Institution (HSI) with a predominantly low-income, first-generation, and largely commuter student population. As a next step, we plan to investigate the factors that influence students' scheduling decisions, such as available study time, course conflicts, commuting demands, and family responsibilities.

The use of testing facilities is expected to grow as enrollment increases and academic integrity cases rise. For example, the **college system** is considering the creation of a cross-institutional testing network for many of the reasons outlined here.

Further research is needed not only to improve practical considerations such as scheduling logistics, but also to re-center assessment as a meaningful learning opportunity. A well-run testing center can support more innovative instructional approaches—such as multiple testing attempts, a common strategy in mastery learning [4], and targeted interventions during the testing window that turn assessment into an active learning experience.

References

- [1] Joni K Adkins and Diana R Linville. “Testing Frequency in an Introductory Computer Programming Course.” In: *Information Systems Education Journal* 15.3 (2017), pp. 22–28.
- [2] Helaine M Alessio et al. “Examining the effect of proctoring on online test scores.” In: *Online Learning* 21.1 (2017), pp. 146–161.
- [3] David G Balash et al. “Examining the examiners: Students’ privacy and security perceptions of online proctoring services”. In: *Seventeenth symposium on usable privacy and security (SOUPS 2021)*. 2021, pp. 633–652.
- [4] Robert L Bangert-Drowns, James A Kulik, and Chen-Lin C Kulik. “Effects of frequent classroom testing”. In: *The journal of educational research* 85.2 (1991), pp. 89–99.
- [5] Lee William Daffin Jr and Ashley A Jones. “Comparing Student Performance on Proctored and Non-Proctored Exams in Online Psychology Courses.” In: *Online Learning* 22.1 (2018), pp. 131–145.
- [6] Kelly Downey et al. “One Solution to Addressing Assessment Logistical Problems: An Experience Setting Up and Operating an In-person Testing Center”. In: *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*. 2024, pp. 317–323.
- [7] Alan Fask, Fred Englander, and Zhaobo Wang. “Do online exams facilitate cheating? An experiment designed to separate possible cheating from the effect of the online test taking environment”. In: *Journal of Academic Ethics* 12.2 (2014), pp. 101–112.
- [8] Robert V. Lindsey et al. “Improving students’ long-term knowledge retention through personalized review”. In: *Psychological Science* 25.3 (2014), pp. 639–647. DOI: 10.1177/0956797613504302.

- [9] Alejandro Marcus et al. “Addressing secure assessments for Internet based distance learning still an irresolvable issue”. In: *Proceedings of the 9th Latin-American Congress of Educational Computing (RIBIE)*. Discusses the limitations of human proctors in high-density classrooms, including fatigue and visual obstructions. Caracas, Venezuela, 2008.
- [10] PrairieTest. *PrairieTest: An exam proctoring system*. Accessed: 2025-06-26. 2025. URL: <https://us.prairietest.com>.
- [11] Mohammad A. Sarrayrih and Mohammad Ilyas. “Challenges of Online Exam, Performances and Problems for Online University Exam”. In: *IJCSI International Journal of Computer Science Issues* 10.1 (2013), pp. 165–172. URL: <https://www.ijcsi.org/articles/Challenges-of-Online-Exam-Performances-and-problems-for-Online-University-Exam.pdf>.
- [12] Donna Stuber-McEwen, Phillip Wiseley, and Susan Hoggatt. “Point, Click, and Cheat: Frequency and Type of Academic Dishonesty in the Virtual Classroom”. In: *Online Journal of Distance Learning Administration* 12.3 (2009). URL: <https://eric.ed.gov/?id=EJ864302>.
- [13] Craig Zilles, Matthew West, and David Mussulman. “Student behavior in selecting an exam time in a computer-based testing facility”. In: *2016 ASEE Annual Conference & Exposition*. 2016.
- [14] Craig Zilles et al. “Computerized testing: A vision and initial experiences”. In: *2015 ASEE Annual Conference & Exposition*. 2015, pp. 26–387.
- [15] Craig Zilles et al. “Student and Instructor Experiences with a Computer-Based Testing Facility”. In: *Proceedings of the 10th International Conference on Education and New Learning Technologies (EDULEARN)*. Often cited for the logistical benefits of UIUC-style testing facilities. 2018. DOI: 10.21125/edulearn.2018.1525.
- [16] Craig Zilles et al. “Student and instructor experiences with a computer-based testing facility”. In: *EDULEARN18 Proceedings*. IATED. 2018, pp. 4441–4450.

Enhancing Engineering Design Thinking and Transferable Skills Through Personalized TinkerCAD Projects*

SJ Kim¹ and Emma Regentova² and Venki Muthukumar²

¹Applied and Interdisciplinary Engineering

²Electrical and Computer Engineering

University of Nevada Las Vegas

Las Vegas, NV 89154

{sj.kim,emma.regentova,venki.muth}@unlv.edu

Abstract

This paper presents an empirical study investigating the effectiveness of TinkerCAD in teaching engineering concepts and design thinking. Thirty-one middle schoolers participated in a two-hour session that included instruction on CAD basics and independent design work. The study examined three key outcomes: understanding of the engineering design process, ability to create 3D models, and development of transferable skills for future projects. Statistical analysis using one-sample t-tests revealed positive effects across all three measures. Participants demonstrated strong understanding of the engineering process ($m = 4.2$, $p = 0.015$), effective comprehension of 3D modeling techniques ($m = 3.81$, $p = 0.0006$), and confidence in applying learned skills to future projects ($m = 3.97$, $p = 0.0001$). The study has practical implications for educators seeking evidence-based approaches to STEM education and suggests potential for broader integration of CAD software in engineering-related curricula.

*Copyright ©2026 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

1 Introduction

Engineering requires a thorough understanding of scientific concepts, problem-solving skills, and creativity. It is essential to introduce engineering concepts to students at a young age to foster their interest in the field.

TinkerCAD is web-based CAD software to design, create, and simulate engineering projects and there is a growing body of research that supports the use of TinkerCAD in the classroom. The use of TinkerCAD in teaching engineering concepts to K-12 has been an area of interest among educators and researchers[15, 8, 2, 13, 12, 1, 4, 10, 3]. Many studies have investigated the benefits of integrating CAD into engineering education that includes concepts. A team of teachers considered CAD as the platform as effective, accessible, and supportive for teaching coding, electronics, and design-based STEM activities [5]. The study found that the use of TinkerCAD was effective in promoting the development of problem-solving skills, creativity, and an interest in the field of engineering. The study also found that the use of TinkerCAD was more effective in teaching the engineering process than traditional teaching methods. Another study conducted by[11] investigated the use of Scratch and TinkerCAD in teaching engineering concepts to middle school students. The study found that both Scratch and TinkerCAD were effective in promoting the development of problem-solving skills, creativity, and an interest in the field of engineering. The study also found that the use of Scratch and TinkerCAD helped students to understand the engineering design process and to apply their knowledge to real-world situations. A study conducted by investigated the use of TinkerCAD in teaching engineering concepts to middle school students from a cognitive perspective. The study found that the use of TinkerCAD helped students to develop spatial visualization skills, which are essential in engineering[15, 6, 14, 9, 10, 3]. The study also found that the use of TinkerCAD helped students to understand the engineering design process and to apply their knowledge to real-world situations.

All these approaches to develop problem-solving skills, fostering their creativity, and cultivating an interest in the field of STEM[5, 7]. However, students were not given the opportunity to learn about engineering processes through activities that emphasized their personality and interests. Designing a name badge is a visual representation of one's name. It consists of the design of the badge shape, text, color and font of the name, and additional design elements. Besides reflecting someone's personality, interests, and identity, a name badge design can also depict how it was made. By designing a name badge with TinkerCAD, middle schoolers would be able to express themselves while gaining the engineering process.

2 Research Questions

A set of three research questions were formed on whether CAD helps middle schoolers understand the engineering process while creating a 3D shape and help create other future 3D models. The following are the three research questions with their hypotheses.

Table 1: Research Questions and Corresponding Hypotheses

Research Question (RQ)	Hypothesis
RQ1: Does using CAD software help middle school students understand the engineering process while designing their own name badge?	Middle school students who use CAD software will demonstrate a stronger understanding of the engineering process.
RQ2: Does using CAD software help middle school students create more accurate and detailed 3D models?	Middle school students who use CAD software will produce more accurate and detailed 3D virtual models.
RQ3: Does using CAD software help middle school students develop transferable skills applicable to future 3D modeling tasks?	Middle school students who use CAD software will develop transferable skills that can be applied to future 3D design projects.

3 Methodology

3.1 Participants

Middle school students enrolled in grades 6 through 8 in Clark County were invited to participate in this study. Participants were recruited as part of the NSF-funded AISL project. Participation was voluntary, and parental consent was obtained for all participants prior to the study. The demographic distribution included both male and female students, ensuring diverse representation across the three grade levels and balanced gender representation, regardless of prior interest in STEM.

3.2 Procedure

The study was conducted over a two-hour session structured in two distinct phases. In the first phase, participants attended a one-hour instructional lecture introducing TinkerCAD’s interface and functionality. This lecture cov-

ered fundamental features including basic shape manipulation, grouping and ungrouping objects, rotating and scaling elements, and using the workplane effectively. The instructor demonstrated essential techniques through live examples and provided hands-on practice opportunities for students to familiarize themselves with the software environment.

In the second phase, participants engaged in an independent design activity for one hour. Each student was tasked with creating a personalized name badge using TinkerCAD. The name badge project required students to apply the techniques learned in the lecture while expressing their individual creativity and personality. Students had autonomy in choosing the badge shape, text style, color scheme, and decorative elements. Throughout the design process, students were encouraged to experiment with different features and iterate on their designs.

3.3 Research Design

This study employed both qualitative and quantitative research design using a post-intervention survey methodology. The independent variables in this study included: (1) exposure to TinkerCAD instruction, and (2) engagement in the name badge design activity. The dependent variables consisted of participants' self-reported perceptions measured across three dimensions: 1) understanding of the engineering design process, 2) confidence in creating 3D virtual models, and 3) perceived ability to transfer learned skills to future projects.

3.4 Data Collection

Data was collected through a structured survey administered immediately following the completion of the name badge design activity. The survey instrument consisted of Likert-scale items designed to measure participants' perceptions of their learning experience. Participants rated their agreement with statements related to the three research questions on a scale from 1 (strongly disagree) to 5 (strongly agree). The survey was designed to capture participants' self-assessment of their understanding and confidence in applying the concepts and skills learned during the session.

3.5 Data Analysis

Quantitative data analysis was conducted using one-sample t-tests. This statistical approach was selected to determine whether participants' mean responses were significantly different from the neutral midpoint, indicating a positive or negative effect of the intervention. Three separate one-sample t-tests were performed, one for each research question. Statistical significance was determined

using an alpha level of 0.05. For each test, the t-statistics, degrees of freedom, p-value, mean score, and standard deviation were calculated and reported.

4 Results and Discussion

A total of 31 middle school students participated in this study, engaging enthusiastically in learning TinkerCAD and designing personalized 3D name badges. Figure 1 displays representative examples of participants' final name badge designs, demonstrating their creative application of the software's capabilities. To evaluate the effectiveness of the TinkerCAD intervention, three one-sample t-tests were conducted with hypothetical means to assess participants' perceptions across three key learning dimensions.



Figure 1: Results of Participants' Name Badge Design.

4.1 Statistical Results

The first analysis examined whether the TinkerCAD session enhanced participants' understanding of the engineering design process. Results revealed statistically significant positive effects ($t = 2.5589$, $df = 30$, $p = 0.015$, $M = 4.2$, $SD = 0.24$). With a mean score of 4.2, participants demonstrated strong agreement that the activity helped them think about engineering processes. The relatively small standard deviation (0.24) indicates consistent positive responses across participants.

The second analysis assessed participants' confidence in creating 3D virtual models following the instruction. This dimension showed even stronger statistical significance ($t = 3.8484$, $df = 30$, $p = 0.0006$, $M = 3.81$, $SD = 0.210$). The mean score of 3.81 indicates that participants felt confident in their newly

acquired 3D modeling skills, with the low p-value (0.0006) providing robust evidence of the intervention's effectiveness.

The third analysis examined participants perceived ability to apply learned skills to future projects. This dimension yielded the strongest statistical results ($t = 4.9908$, $df = 30$, $p = 0.0001$, $M = 3.97$, $SD = 0.194$). The mean score of 3.97 and highly significant p-value (0.0001) suggest that participants felt well-equipped to transfer their TinkerCAD skills to other contexts, demonstrating the activity's potential for fostering transferable learning.

4.2 Pedagogical Implications for Higher Education

The findings from this study offer several critical insights for college-level computing and engineering education.

Scaffolding Effect for College Readiness. The strong positive results suggest that early exposure to CAD tools like TinkerCAD can effectively prepare students for more advanced design software encountered in college engineering and computer science programs. College professors should recognize that incoming students with K-12 CAD experience may demonstrate stronger spatial reasoning abilities and design thinking skills. This creates opportunities for differentiated instruction in introductory courses, where professors can leverage students' prior experience while supporting those without such backgrounds.

Low Barrier to Entry for Learners. The study's success with middle school students, achieved in just two hours of instruction, demonstrates TinkerCAD's accessibility for novice learners. College instructors teaching non-majors or students from diverse academic backgrounds can apply this finding by incorporating TinkerCAD in introductory courses before transitioning to professional-grade CAD software. This approach can reduce frustration and improve retention in STEM disciplines where spatial visualization is critical.

Project-Based Learning Effectiveness. The name badge design task engaged students' personal identity and creativity while teaching technical skills. College professors could adopt similar strategies by designing projects that balance technical learning objectives with student autonomy and personal relevance. The statistical evidence suggests that personally meaningful projects enhance students' confidence in applying skills to new contexts.

Rapid Skill Acquisition for Prototyping. The ability of middle school students to grasp 3D modeling concepts in a condensed timeframe has implications for college courses incorporating maker education and rapid prototyping. Engineering and design programs could implement similar short-cycle tutorials to quickly bring students to functional competency, reserving more class time for complex applications and iterative design processes.

5 Conclusions and Future Works

This study empirically demonstrated that a two-hour TinkerCAD intervention significantly enhanced middle school students' understanding of engineering design processes ($M = 4.2$, $p = 0.015$), confidence in creating 3D virtual models ($M = 3.81$, $p = 0.0006$), and perceived ability to transfer skills to future projects ($M = 3.97$, $p = 0.0001$). All three measures exceeded the neutral midpoint with strong statistical significance, providing robust evidence that accessible CAD tools can effectively introduce foundational engineering and design concepts to novice learners in a condensed timeframe.

The personalized name badge design activity proved particularly effective as a pedagogical strategy, combining technical skill development with creative expression and identity formation. This approach aligns with research on project-based learning and suggests that personally meaningful tasks enhance both engagement and skill transfer. The consistency of positive results across all participants ($SD < 0.25$ for all measures) indicates that the intervention was broadly effective regardless of individual differences in prior experience or academic background.

This study makes several contributions to computing and engineering education research. The study bridges K-12 and higher education perspectives by explicitly connecting middle school learning outcomes to college-level pedagogical concerns. This cross-level analysis is particularly valuable for CCSC conference attendees who may engage in curriculum development, teacher preparation, or educational research spanning multiple educational levels. Another contribution is that the focus on transfer of learning addresses a critical challenge in computing education, ensuring that students can apply learned skills in novel contexts. The strong transfer results suggest that the combination of accessible tools, structured instruction, and open-ended creative tasks can effectively promote transferable learning finding with implications for course design across educational levels.

Future research should track participants over time to assess whether the positive perceptions documented in this study translate into sustained engagement with CAD tools, enrollment in STEM courses, or pursuit of computing and engineering careers. Longitudinal data would strengthen claims about the long-term impact of early CAD exposure on college readiness and STEM participation. Another direction for future work is the development and validation of more comprehensive assessment instruments beyond self-report measures. Performance-based assessments of spatial reasoning, design quality evaluations, and objective measures of skill transfer would complement perceptual data and strengthen claims about learning outcomes.

6 Limitations

This study has several limitations that should be considered when interpreting the results and their generalizability. The study involved 31 middle school students only from Clark County, Nevada. While this sample size was sufficient for detecting statistically significant effects, a larger and more demographically diverse sample would strengthen claims about the intervention's effectiveness across varied populations. The study relied entirely on self-reported perceptions of learning rather than objective performance measures. While students' confidence and self-efficacy are valuable outcomes, they may not perfectly correlate with actual skill acquisition or retention. Participants may have overestimated their abilities due to the Dunning-Kruger effect or social desirability bias. Future research should incorporate performance-based assessments, such as evaluating the technical quality of students' designs or testing their ability to complete novel CAD tasks. The two-hour duration of the intervention, while sufficient to demonstrate immediate positive effects, does not provide evidence of long-term learning or skill retention. Students' enthusiasm and positive perceptions immediately following the activity may not persist over time. Longitudinal studies tracking participants' continued engagement with CAD tools and spatial reasoning development would address this limitation. This study employed a post-intervention survey design without a control group receiving alternative instruction or no instruction. While the one-sample t-tests compared results to a neutral midpoint rather than a pre-intervention baseline, a randomized controlled trial would provide stronger causal evidence. Future research should include control conditions to isolate the specific effects of TinkerCAD instruction from general engagement or novelty effects. Participants completed only one type of project (name badge design), which may limit the generalizability of findings to other CAD applications or engineering tasks. Different design challenges requiring varied spatial reasoning skills or engineering concepts might yield different results. Research examining multiple project types would provide more comprehensive evidence of TinkerCAD's educational effectiveness. The quality and consistency of instruction may have influenced outcomes. While the same instructor delivered all sessions, individual teaching style, expertise level, and rapport with students could have affected results in ways not measured by this study. Multi-site studies with different instructors would help separate intervention effects from instructor-specific factors.

7 Acknowledgement

This study was supported by the National Science Foundation under Grant No. 2115331. The authors sincerely appreciate the students and educators

who participated in this study for their time and valuable contributions.

References

- [1] B.L. Asbendri et al. “Exploring the Impact of Tinkercad-Assisted Learning on Student Performance in Industrial Electronics Subject”. In: *Journal of Hypermedia & Technology-Enhanced Learning* 2.2 (2024), pp. 134–148.
- [2] A. Barbosa, I. Vale, and D. Alvarenga. “The use of Tinkercad and 3D printing in interdisciplinary STEAM education: A focus on engineering design”. In: *STEM Education* 4.2 (2024), pp. 222–246.
- [3] Srinjita Bhaduri et al. “3DnST: A Framework Towards Understanding Children’s Interaction with Tinkercad and Enhancing Spatial Thinking Skills”. In: *Proceedings of the 20th Annual ACM Interaction Design and Children Conference*. IDC ’21. Athens, Greece: Association for Computing Machinery, 2021, pp. 257–267.
- [4] Y. Chen, L. Cao, and Y. Zhang. “Teachers as makers: How K-12 teachers design 3D making lessons for classroom teaching”. In: *Education and Information Technologies* 28.8 (2023), pp. 6947–6975.
- [5] G. Deniz and S. Eryilmaz. “Information Technology Teachers’ Views on the Use of Tinkercad”. In: *Adıyaman Üniversitesi Eğitim Bilimleri Dergisi* 11.1 (2021), pp. 1–17.
- [6] H. E. Dere. “The effects of using web-based 3D design applications on spatial visualization and mental rotation abilities of middle school students”. Unpublished doctoral dissertation. Ankara University, 2017.
- [7] R. Erdogan et al. “Examination of the usability of Tinkercad application in educational robotics teaching by eye tracking technique”. In: *Smart Learning Environments* 10.1 (2023), p. 27.
- [8] S. Eryilmaz and G. Deniz. “Effect of Tinkercad on students’ computational thinking skills and perceptions: A case of Ankara Province”. In: *Turkish Online Journal of Educational Technology* 20.1 (2021), pp. 25–38.
- [9] S. Fowler et al. “Technology enhanced learning environments and the potential for enhancing spatial reasoning: A mixed methods study”. In: *Mathematics Education Research Journal* 33.1 (2021), pp. 145–168.
- [10] C. Julia and J.O. Antolì. “Enhancing spatial ability and mechanical reasoning through a STEM course”. In: *International Journal of Technology and Design Education* 28.4 (2018), pp. 957–983.

- [11] E. Maloney et al. “The Scratch programming language and environment”. In: *ACM Transactions on Computing Education* 20.4 (2010), p. 16.
- [12] J.-A. Marin-Marin, P. A. Garcia-Tudela, and P. Duo-Terron. “Computational thinking and programming with Arduino in education: A systematic review for secondary education”. In: *Heliyon* 10.8 (2024), e29177.
- [13] N. Standish et al. “The Effects of an Engineering Design Module on Student Learning in a Middle School Science Classroom”. In: *International Journal of Learning, Teaching and Educational Research* 15.6 (2016), pp. 59–72.
- [14] D. H. Uttal and C. A. Cohen. “Spatial thinking and STEM education: When, why, and how?” In: *Psychology of Learning and Motivation* 57 (2012), pp. 147–181.
- [15] J. Yue and D. M. Chen. “Does CAD improve spatial visualization ability?” In: *ASEE Annual Conference & Exposition Proceedings* (2001).

Evaluating Large Language Models on Solved and Unsolved Problems in Graph Theory: Implications for Computing Education*

Adithya Kulkarni¹, Mohna Chakraborty², and Jay Bagga¹

¹Department of Computer Science

Ball State University

Muncie, IN 47306

{adithya.kulkarni, jbagga}@bsu.edu

²School of Artificial Intelligence and Data Science

Jio Institute

Navi Mumbai, India

mohna.chakraborty@jioinstitute.edu.in

Abstract

Large Language Models are increasingly used by students to explore advanced material in computer science, including graph theory. As these tools become integrated into undergraduate and graduate coursework, it is important to understand how reliably they support mathematically rigorous thinking. This study examines the performance of a LLM on two related graph theoretic problems: a solved problem concerning the gracefulness of line graphs and an open problem for which no solution is currently known. We use an eight stage evaluation protocol that reflects authentic mathematical inquiry, including interpretation, exploration, strategy formation, and proof construction.

*Copyright ©2026 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

The model performed strongly on the solved problem, producing correct definitions, identifying relevant structures, recalling appropriate results without hallucination, and constructing a valid proof confirmed by a graph theory expert. For the open problem, the model generated coherent interpretations and plausible exploratory strategies but did not advance toward a solution. It did not fabricate results and instead acknowledged uncertainty, which is consistent with the explicit prompting instructions that directed the model to avoid inventing theorems or unsupported claims.

These findings indicate that LLMs can support exploration of established material but remain limited in tasks requiring novel mathematical insight or critical structural reasoning. For computing education, this distinction highlights the importance of guiding students to use LLMs for conceptual exploration while relying on independent verification and rigorous argumentation for formal problem solving.

1 Introduction

Large Language Models (LLMs) are increasingly used by students in computer science courses to clarify concepts, explore ideas, and assist with problem solving. Prior work documents both the potential educational benefits of these tools and the risks associated with overreliance on model generated reasoning [27, 25, 9, 29]. In courses involving mathematically rigorous material, such as graph theory, understanding how LLMs reason is essential for responsible integration into computing education.

Graph theory provides a useful setting for examining these issues. It is a foundational subject in many computing curricula and supports applications in networks, data science, and machine learning. Within graph theory, line graphs and graceful labeling are well studied topics that remain challenging even at advanced levels [3, 4, 8]. Students increasingly consult LLMs when engaging with such material, which motivates a careful evaluation of model behavior in this domain.

This study focuses on two closely related problems in graph theory that differ in logical direction and in their status within the literature. The first problem, which is solved, asks whether the line graph of a nongraceful graph must be graceful:

Problem 1: If G is nongraceful, is $L(G)$ graceful?

The second problem, which remains open, asks whether the converse implication holds:

Problem 2: If $L(G)$ is graceful, is G graceful?

While these statements appear superficially similar, they pose fundamentally different challenges. Problem 1 has known counterexamples and established solutions in the literature, allowing us to evaluate whether a model can correctly interpret definitions and reproduce valid reasoning. In contrast, no general solution is known for Problem 2, even for restricted classes of graphs, making it a suitable test of model behavior in the presence of genuine mathematical uncertainty.

We evaluate the model using an eight stage protocol¹ that reflects the typical phases of mathematical inquiry. The model is asked to interpret each problem, generate exploratory ideas, identify related areas, recall relevant results, propose strategies, construct proofs, critique its own reasoning, and revise its arguments. This structure enables a detailed analysis of both exploratory reasoning and formal deduction.

The contributions of this study are threefold. First, we provide an empirical characterization of LLM behavior on both a solved and an unsolved problem in graph theory. Second, we show that the model performs reliably on tasks grounded in established theory, producing accurate interpretations and a correct proof for the solved problem that was confirmed by an expert. Third, we show that for the open problem, the model generates coherent exploratory reasoning but does not produce new mathematical insight or advance the state of knowledge.

Overall, this work clarifies how LLMs can support conceptual exploration in advanced computing courses while also highlighting their limitations in tasks that require verification, abstraction, and original reasoning.

2 Background and Related Work

This section reviews prior work relevant to three aspects of this study: graph theory and line graphs, the use of LLMs in educational contexts, and research on hallucination and reasoning behavior in LLMs. Together, these areas provide the context for evaluating how an LLM behaves when engaging with both solved and unsolved problems in graph theory.

2.1 Graph Theory, Line Graphs, and Graceful Labeling

Line graph theory is a well established area of graph theory with foundational results dating to the mid twentieth century and continued development in recent decades. Research in this area documents both classical properties and modern generalizations of line graphs, reflecting their broad applicability

¹The prompts and LLM outputs can be found at https://github.com/kulkarniadithya/Dual_Role_of_LLM/

and sustained theoretical interest [3, 4]. Graceful labeling is another central topic in graph theory, characterized by a large and evolving body of literature. Gallian provides a comprehensive and continuously updated survey of graph labeling results, including graceful graphs and related constructions [8]. Both line graphs and graceful labeling are frequently encountered in advanced graph theory courses and remain active areas of research.

Additional work has examined structural properties and construction techniques for graceful graphs [7, 13]. More recent studies explore line graph transformations in computational settings such as network analysis and machine learning, demonstrating the continued relevance of these concepts beyond purely theoretical contexts [30, 21]. The depth and diversity of this literature make line graphs and graceful labeling well suited for evaluating reasoning in mathematically rigorous settings.

2.2 Large Language Models in Education and Research

LLMs are increasingly used in higher education to support conceptual understanding, literature exploration, assignment assistance, and programming tasks. Surveys and empirical studies report potential benefits such as improved accessibility, personalized learning support, and increased efficiency in instructional contexts [27, 25, 9, 20]. Additional research examines the broader lifecycle of LLM adoption in education, including issues related to bias, transparency, and long term impact on learning practices [15].

At the same time, concerns have been raised about students' tendency to overestimate the reliability of LLM generated output, particularly for tasks involving complex reasoning. Studies in STEM education document cases in which learners accepted incorrect or incomplete explanations produced by LLMs, leading to misunderstandings or the reinforcement of errors [14, 17]. These findings motivate closer examination of how LLMs behave in mathematically demanding contexts, where correctness depends on precise definitions and rigorous logical structure.

2.3 Hallucination and Reasoning Behavior in LLMs

A substantial body of work investigates hallucination and reasoning limitations in LLMs. Recent surveys categorize different forms of hallucination, analyze contributing factors, and discuss the implications for reliable deployment of these systems [11, 2]. In mathematical and logical domains, LLMs have been shown to generate incorrect statements, misapply known results, or construct arguments that violate formal reasoning principles [6, 1, 28, 10]. Such behavior raises concerns about the suitability of LLMs for tasks requiring deductive validity.

Other studies examine internal signals associated with hallucination risk [12, 23] and analyze the educational consequences of erroneous AI generated feedback [26]. Proposed mitigation approaches include self consistency checks, structured reasoning prompts, and fine grained hallucination detection techniques [19, 22, 16, 18, 24]. While these methods improve reliability in some settings, they do not eliminate the fundamental limitations of current models when reasoning requires original insight or extrapolation beyond known patterns.

Taken together, prior work suggests that LLMs are effective at organizing and explaining established material but remain limited when tasks require the construction of new mathematical arguments. This distinction motivates our evaluation of LLM behavior on both a solved and an unsolved problem in graph theory and informs the educational implications discussed in the remainder of this paper.

3 Methodology

For a graph G , the line graph $L(G)$ has a vertex for each edge of G , and two vertices in $L(G)$ are adjacent if the corresponding edges of G are adjacent [5]. This study examines the behavior of a LLMs when engaged in a structured mathematical problem solving workflow involving two closely related statements about gracefulness and the line graph operator. The two statements differ in logical direction and in their status in the literature. Problem 1 is solved, meaning that established arguments and known counterexamples exist. Problem 2 is open, meaning that no general proof or counterexample is currently known. This design supports a direct comparison between model behavior when a canonical resolution exists and when genuine mathematical uncertainty remains.

3.1 Why Problem 1 Is Solved and Why Problem 2 Is Difficult

Problem 1 asks:

Problem 1: If G is nongraceful, is $L(G)$ graceful?

This statement is solved because the implication is false: there are graphs G that are nongraceful for which $L(G)$ is not graceful. In other words, counterexamples exist, so the statement can be resolved by exhibiting a specific G and verifying that neither G nor $L(G)$ admits a graceful labeling. This type of resolution is typical in graph labeling, where disproving an implication often reduces to identifying a concrete graph family with a provable obstruction.

Problem 2 asks the converse implication:

Problem 2: If $L(G)$ is graceful, is G graceful?

Problem 2 remains difficult because it requires reasoning about how graceful labelings behave under an inverse transformation. The line graph operator loses information: different nonisomorphic graphs can share the same line graph structure, and structural properties of $L(G)$ do not uniquely determine labeling properties of G . As a result, techniques that work for forward implications often do not transfer to the reverse direction. In practice, progress on Problem 2 typically requires either a general structural theorem that lifts labelings from $L(G)$ back to G or a construction of a counterexample family. Neither is currently known.

3.2 Experimental Environment and Materials

All experiments were conducted using the publicly available web interface of ChatGPT 5.1². The interface supports multimodal inputs such as PDF uploads, which is consistent with common student practice in advanced coursework. Students frequently provide a problem statement together with lecture notes or papers and then ask the model to interpret definitions, suggest research directions, or critique partial solutions.

In the first prompt for each problem, the model received two background documents: *A Dynamic Survey of Graph Labeling* [8] and *On Graceful Line Graphs*³. These documents provide definitions and known results related to line graphs and graceful labelings. Each problem was evaluated in a separate conversation to avoid cross conversation persistence. All prompts and model outputs were saved verbatim. No feedback, correction, or reinforcement was provided during any stage of the evaluation.

3.3 Evaluation Protocol and Prompt Design

The evaluation uses an eight stage protocol that reflects a typical workflow students follow when using LLMs for difficult mathematical tasks: first clarifying the problem and definitions, then generating ideas and relevant context, and only later attempting formal proof and revision. The same prompt templates were used for both problems, with only the underlying problem statement changed. Several prompts explicitly instructed the model to avoid inventing theorems or unsupported claims and to indicate uncertainty when appropriate,

²<https://openai.com/index/gpt-5-1/>.

³This document was obtained from the authors at the 2025 CCSC Midwest Conference and was used as background material for definitions and context.

in order to reduce fabricated academic content and to observe model behavior under this constraint.

For completeness, the exact wording of the prompts as presented to the model for Problem 1 is provided below. Italics indicate the text displayed to the model.

Prompt 1: Problem Understanding.

I want your help in developing proof (either prove or disprove) for the below unsolved problem:

Problem: If G is nongraceful then $L(G)$ is graceful?

I am providing two papers that have information on the definitions of line graph, graceful labeling, and related concepts. These papers also provide you with necessary background that can be useful to develop the proof for the problem.

Please understand the problem and read the attached PDFs carefully and thoroughly. Then:

- 1. Restate the main problem or conjecture in your own words.*
- 2. List all key definitions and properties involved.*
- 3. Explain why the problem is challenging or interesting.*
- 4. Identify any natural simplifications or special cases that might be useful starting points.*

Prompt 2: Brainstorming.

Based on your understanding of the problem, please brainstorm 6–10 possible approaches or ideas that a researcher might explore. For each idea:

- 1. Describe the core insight.*
- 2. Explain why it might be promising.*
- 3. Point out potential obstacles.*

Prompt 3: Related Areas.

Before we proceed, can you identify the areas of graph theory or graph labeling that are most closely related to the problem. For each area:

- 1. Name it.*
- 2. Explain the relationship to the problem.*
- 3. Mention concepts or tools from that area that might be relevant.*

Prompt 4: Theorem and Result Recall.

Now list 8–12 theorems, known results, or named conjectures that you believe are directly relevant to the problem. For each one:

- 1. Provide the name of the result.*
- 2. State the authors (if known).*
- 3. Give a 2–3 sentence summary.*
- 4. Indicate if you are unsure.*

Please do not invent results or authors. Only include items you have high confidence in.

Prompt 5: Strategy Formation.

Propose 3–5 distinct high level strategies to try to resolve the problem. They do not need to be correct. For each strategy:

- 1. Explain the core idea.*
- 2. List any structural lemmas that might be needed.*
- 3. Predict the hardest obstacle in making the strategy succeed.*

Prompt 6: Proof Attempt.

Now attempt to prove the main conjecture or implication. Write the proof in a rigorous, step by step manner, as if submitting to a mathematics journal. State each lemma clearly, justify each inference, and avoid vague reasoning.

Prompt 7: Self Evaluation.

Please re read your entire proof carefully. Identify any steps that may be incorrect, incomplete, or unjustified. For each such step:

- 1. Explain why it might be flawed.*
- 2. Attempt to repair or clarify the argument.*

Prompt 8: Revised Proof.

Now, can you correct the identified flaws and provide a final rigorous proof for the problem without any flaws.

3.4 Expert Evaluation Procedure

All outputs were evaluated by a faculty expert in graph theory with research experience in line graphs and graph labelings. The expert was not an expert in artificial intelligence, which reduces the risk that judgments were influenced by assumptions about typical LLM behavior. Prior to evaluation, the expert was provided with explicit written instructions specifying what aspects of each prompt response to analyze. These prompt specific evaluation guidelines were designed to align with the goals of each stage of the protocol and are publicly available in the project repository.

For each prompt and each problem, the expert recorded qualitative comments and assessed the LLM output against the provided criteria. For prompts focused on interpretation and exploration (Prompts 1 to 5), the expert evaluated conceptual correctness of definitions, fidelity of problem restatement, relevance of identified themes, and appropriateness of proposed research directions. For the theorem recall prompt (Prompt 4), the expert verified whether listed theorems and author attributions were correct, relevant, or potentially fabricated. For proof oriented prompts (Prompts 6 to 8), the expert evaluated deductive validity by checking whether claims followed logically from prior statements, whether definitions were applied correctly, and whether the argument addressed the stated problem. The expert also examined whether the model’s self evaluation and revision stages identified substantive flaws and made meaningful corrections.

This evaluation procedure supports a systematic comparison between the model’s exploratory usefulness and its limitations in proof oriented reasoning. It also enables a direct contrast between performance on a solved problem, where established reasoning exists, and an unsolved problem, where no reference solution is available.

4 Results

The evaluation examined the model’s behavior on two graph theoretic problems that differ in logical direction and in their status within the literature: a solved problem and an unsolved problem. Expert analysis shows that the model performed reliably when established solutions and reasoning patterns were available, while its limitations became apparent when the task required reasoning beyond known mathematical results. The findings are presented by problem and by stage of the eight prompt evaluation protocol.

4.1 Problem 1: If G is nongraceful, is $L(G)$ graceful?

4.1.1 Prompts 1 to 5: Exploratory and Conceptual Reasoning

Across the first five prompts, the model demonstrated strong conceptual understanding and generated reasoning that aligned closely with expert expectations.

Prompt 1: Problem Interpretation. The model restated the problem accurately, identified all relevant definitions, and extracted the structural role of the line graph transformation. The expert noted that the interpretation was precise and reflected a correct understanding of the conjecture.

Prompt 2: Idea Generation. The model produced several correct and relevant approaches, including use of degree patterns, properties of known graceful families, and the examination of structural counterexamples. The expert found these ideas appropriate and aligned with genuine mathematical strategies used in the literature.

Prompt 3: Related Areas. The model identified areas such as graph labelings, structural characterizations of line graphs, tree labelings, and decomposition methods. The expert evaluation confirmed the relevance of these areas and the correctness of the connections drawn.

Prompt 4: Recall of Results. A notable outcome is that the model did not hallucinate any theorems. All referenced results were correct, relevant, and stated with appropriate caution. This behavior contrasts with patterns documented in prior work and demonstrates unusually reliable performance on this prompt.

Prompt 5: Strategy Formation. The strategies were coherent, structurally sound, and consistent with how a human expert might begin approaching the problem. The expert noted that the model displayed a clear understanding of how to decompose the problem into meaningful subproblems.

4.1.2 Prompts 6 to 8: Proof Construction and Self Evaluation

Prompt 6: Proof Attempt. The model produced a correct argument for the solved problem. It identified a counterexample and provided a complete proof that the expert confirmed as valid. No circular reasoning, fabricated lemmas, or unjustified generalizations were observed.

Prompt 7: Self Evaluation. The model reviewed its proof and did not introduce new errors. The expert indicated that the self evaluation was reasonable given that the proof was already correct.

Prompt 8: Revised Proof. The model presented a refined version of the proof that remained correct. The expert confirmed that the proof was complete, logically sound, and consistent with known results.

Overall, for Problem 1 the model demonstrated reliable conceptual reasoning, accurate recall of known results, and correct formal reasoning.

4.2 Problem 2: If $L(G)$ is graceful, is G graceful?

4.2.1 Prompts 1 to 5: Exploratory and Conceptual Reasoning

The model again showed strong performance on conceptual and exploratory tasks, despite the fact that the problem is open and has no known resolution.

Prompt 1: Problem Interpretation. The model restated the problem clearly and identified correct definitions and structural components. It also articulated the reasons the conjecture is difficult, such as limited understanding of inverse line graph behavior in the context of graceful labelings. The expert noted that the explanation was accurate and appropriate.

Prompt 2: Idea Generation. The model proposed several useful exploratory ideas, including analysis of special graph families and examination of structural features in the inverse direction of the line graph operation. While some ideas were incomplete or required refinement, the expert found them reasonable and aligned with authentic research directions.

Prompt 3: Related Areas. The model correctly connected the problem to labeling theory, line graph characterizations, and decomposition methods. The expert evaluation confirmed that these connections were sound.

Prompt 4: Recall of Results. As in Problem 1, the model did not hallucinate any theorems. The results it listed were correct and relevant. This consistency across two problems suggests reliability in recalling established literature.

Prompt 5: Strategy Formation. The model produced coherent strategies for investigating the conjecture, including systematic examination of special graph classes and potential structural reductions. The expert found these strategies meaningful and aligned with the type of exploratory work expected in early research stages.

4.2.2 Prompts 6 to 8: Proof Construction and Self Evaluation

Prompt 6: Proof Attempt. The model attempted to construct a proof but did not succeed. Importantly, its argument did not contain fabricated theorems or logically incoherent steps. Instead, the expert observed that the model produced partial reasoning that remained bounded by known results and did not invent unsupported claims.

Prompt 7: Self Evaluation. The model correctly identified its own limitations, noting instances of incomplete reasoning and uncertainty. The expert described this as a reasonable and honest assessment of its own output.

Prompt 8: Revised Proof. The model did not claim to have resolved the open problem. Instead, it shifted toward describing partial progress and suggesting relevant subproblems. The expert found this appropriate and remarked that the model accurately refrained from fabricating a solution.

For Problem 2, the model demonstrated strong exploratory skills but was unable to produce new mathematical insight, which is expected given the open nature of the problem.

Summary Across both problems, the model showed reliable conceptual and exploratory reasoning. It performed particularly well on the solved problem, producing a correct proof and accurate references, which demonstrates an ability to organize and apply established knowledge. For the open problem, the model again displayed coherent exploratory reasoning and did not fabricate results or claim a solution, instead acknowledging uncertainty. At the same time, its performance revealed a consistent limitation: the model was unable to engage in the type of critical thinking required to generate new structural insight or evaluate the deeper consequences of its own partial arguments. This inability to move beyond known patterns and to reason flexibly in unfamiliar settings highlights a fundamental constraint of current Large Language Models. These findings suggest that while the model is effective for engaging with established material, it remains limited in tasks that require original mathematical discovery, critical evaluation, or synthesis beyond existing knowledge.

5 Discussion

The results reveal a clear contrast between the model’s behavior on a solved problem and an unsolved one. When established theory was available, the model interpreted the problem accurately, recalled relevant results, and constructed a valid proof. In contrast, for the unsolved problem, the model produced coherent exploratory reasoning but did not generate new insight or extend existing results. This distinction has important implications for computing education as students increasingly use LLMs in mathematically rigorous settings.

5.1 Implications for Student Use of LLMs

The model’s performance on the exploratory stages of both problems indicates that these systems can be useful in helping students begin work on complex or unfamiliar material. The model was able to restate each conjecture accurately, identify relevant definitions and structures, and propose plausible strategies. These capabilities are consistent with prior research showing that LLMs can support conceptual understanding and reduce initial barriers in advanced coursework [27, 25, 9].

The inability of the model to advance the unsolved problem, however, illustrates a limitation that students may overlook. Although the model did not hallucinate theorems or assert false conclusions, it also did not perform the type of reasoning that produces new mathematical ideas. Students may interpret

a coherent explanation as evidence of underlying understanding, even when the model is only reorganizing existing patterns. Prior work documents that learners may accept fluent but unverified reasoning as authoritative [14, 26]. Students therefore need explicit guidance to recognize the distinction between exploratory assistance and reliable mathematical reasoning.

5.2 Critical Thinking Limitations of LLMs

The distinction between the solved and unsolved problems provides a focused setting in which to observe the limitations of current models in tasks that require critical thinking. The model did not propose new constructions, search for counterexamples, refine partial ideas based on structural consequences, or assess the implications of its own reasoning steps. These behaviors are central to mathematical discovery but are not part of the inference process used by LLMs.

LLMs generate responses by estimating patterns in training data rather than by performing deliberate analysis or hypothesis testing. They do not build abstractions, evaluate alternative possibilities, or engage in reflective reasoning about correctness. As a result, they can reproduce reasoning that resembles known arguments but cannot reliably extend those arguments into new domains. This structural limitation explains why the model performed well on the solved problem but could not progress on the unsolved one, despite producing coherent and relevant intermediate ideas.

5.3 Implications for Instruction and Course Design

For instructors, the results suggest that LLMs can be integrated productively into the early phases of learning. Tasks involving conceptual restatement, identification of related areas, or preliminary strategy formation align with the strengths observed in the evaluation. Activities that require students to compare their own ideas with model generated ideas or to critique these suggestions may support metacognitive development.

It is equally important to design assignments that require students to verify claims, assess the structure of arguments, and recognize the limitations of model generated reasoning. The inability of the model to produce novel insight on the unsolved problem demonstrates that these tools cannot replace the critical analysis that is central to advanced study in graph theory and related areas. Instruction should therefore emphasize the need to evaluate model output independently and to understand the boundary between known results and open problems.

5.4 Relevance for Advanced Topics in Computing

Graph theory remains an essential area of instruction in many computer science programs, with continued research activity and broad applicability [3, 4]. As students encounter topics that approach current research frontiers, they may increasingly rely on LLMs to clarify definitions or explore structural themes. The findings of this study indicate that such use can be productive when the material is grounded in established knowledge, but that these tools cannot be expected to resolve open questions or generate new theoretical contributions.

Summary Overall, the findings show that LLMs are effective exploratory tools but remain limited as reasoning engines in tasks that require original mathematical insight. Their strengths can support conceptual engagement, problem interpretation, and strategic planning, but they must be complemented by explicit training in verification and mathematical rigor. Responsible integration in computing education requires helping students understand both the capabilities and the boundaries of these systems.

6 Recommendations

The results of this study show that LLMs are effective for exploratory engagement with established material but remain limited when tasks require verification or novel mathematical reasoning. Based on these findings, we offer concrete recommendations for students, instructors, and researchers who use LLMs in advanced computing contexts.

6.1 For Students

Use LLMs to initiate problem solving, not to conclude it. Students can use LLMs to restate problems, recall definitions, identify relevant concepts, and generate possible approaches. However, correctness must be established through independent reasoning, as demonstrated by the model’s inability to resolve the unsolved problem.

Treat model output as a draft, not an answer. When reviewing LLM generated arguments, students should explicitly check each logical step, verify assumptions, and confirm whether conclusions follow from definitions.

Use LLMs to orient literature exploration. LLMs can help students identify relevant terminology, subtopics, and canonical references that serve as starting points for literature searches. This use is most effective when model output is treated as an initial guide and all references are verified using authoritative academic sources.

Verify claims using external sources. References to theorems or known results should be checked against textbooks, peer reviewed literature, or instruc-

tor provided materials. Prior studies show that students may accept plausible but incorrect reasoning unless verification is explicitly required [14, 26].

6.2 For Instructors

Integrate LLMs into early phases of coursework. The model performed consistently well in tasks such as problem interpretation and idea generation. Instructors can incorporate LLMs into activities that emphasize understanding problem structure or exploring alternative approaches.

Require explicit validation of LLM generated reasoning. Assignments should ask students to assess the correctness of model output, identify limitations, or explain why an argument succeeds or fails.

Set clear expectations for appropriate use. Explicit guidelines can help students recognize when LLMs are useful and when independent analysis is required, particularly for open problems or proof based tasks.

6.3 For Researchers

Use LLMs for exploration and organization. The model generated coherent strategies and contextual information but did not produce new insight for the open problem. Researchers should treat LLMs as tools for organizing known material rather than resolving open questions.

Use LLMs as orientation tools for unfamiliar areas. LLMs may assist researchers in identifying common terminology, related problem areas, and possible connections across subfields. This role is limited to initial orientation and should precede systematic literature review.

Maintain transparency and verification. When LLMs are used during exploratory stages, prompts and outputs should be documented and all claims independently verified.

Overall, these recommendations emphasize responsible and practical use of LLMs in mathematically rigorous settings. LLMs can support orientation and exploration, but verification and critical reasoning must remain central to learning and research.

7 Conclusion

This study evaluated how an LLM responded to two graph theoretic problems, one solved and one unsolved, using an eight stage protocol that examined conceptual reasoning, strategy formation, and proof construction. The results show that the model performs reliably when the problem has an established solution. For the solved problem, it interpreted the conjecture accurately, recalled relevant results, and produced a correct proof confirmed by an expert.

These findings indicate that LLMs can assist learners in understanding definitions, recognizing structures, and engaging with known material.

For the unsolved problem, the model offered coherent interpretations and exploratory ideas but did not generate new insight or progress toward a solution. It did not fabricate results and instead acknowledged uncertainty. This behavior reflects a structural limitation of current models: they can organize and apply existing knowledge but cannot perform the critical thinking or abstraction required for novel mathematical reasoning.

These observations have direct implications for computing education. LLMs can support early stage exploration but cannot replace verification or formal argumentation. Effective use in advanced coursework requires guiding students in validating model output, understanding its limits, and developing independent reasoning skills. Overall, the findings highlight both the value and the boundaries of LLMs as tools for learning in mathematically rigorous domains.

References

- [1] Janice Ahn et al. “Large language models for mathematical reasoning: Progresses and challenges”. In: *arXiv preprint arXiv:2402.00157* (2024).
- [2] Aisha Alansari and Hamzah Luqman. “Large Language Models Hallucination: A Comprehensive Survey”. In: *arXiv preprint arXiv:2510.06265* (2025).
- [3] Jay Bagga. “Old and new generalizations of line graphs”. In: *International Journal of Mathematics and Mathematical Sciences* 2004.29 (2004), pp. 1509–1521.
- [4] Jay Bagga and Lowell Beineke. “A survey of line digraphs and generalizations”. In: *Discrete Math. Lett* 6 (2021), pp. 68–83.
- [5] Lowell Beineke and Jay Bagga. *Line graphs and line digraphs*. Springer, 2021.
- [6] Johan Boye and Birger Moell. “Large language models and mathematical reasoning failures”. In: *arXiv preprint arXiv:2502.11574* (2025).
- [7] Charles Michael Cavalier. “Graceful labelings”. In: *Bachelor of Science (Louisiana State University 2006)* <https://people.math.sc.edu/czabarka/Theses/Graceful.pdf> (2009).
- [8] Joseph A Gallian. “A dynamic survey of graph labeling”. In: *Electronic Journal of combinatorics* 1.DynamicSurveys (2018), DS6.
- [9] Sghaier Guizani et al. “A systematic literature review to implement large language model in higher education: issues and solutions”. In: *Discover Education* 4.1 (2025), pp. 1–25.

- [10] Alex Heyman and Joel Zylberberg. “Reasoning large language model errors arise from hallucinating critical problem features”. In: *arXiv preprint arXiv:2505.12151* (2025).
- [11] Lei Huang et al. “A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions”. In: *ACM Transactions on Information Systems* 43.2 (2025), pp. 1–55.
- [12] Ziwei Ji et al. “Llm internal states reveal hallucination risk faced with a query”. In: *arXiv preprint arXiv:2407.03282* (2024).
- [13] Kyrie Johnson. *Constructing Graceful Graphs by Extending Paths from Graceful Graphs*. 2018. URL: <https://sites.math.duke.edu/~kjohnson/Extending%20paths%20from%20graceful%20graphs.pdf>.
- [14] Lars Krupp et al. “Unreflected acceptance—investigating the negative consequences of ChatGPT-assisted problem solving in physics education”. In: *arXiv preprint arXiv:2309.03087* (2023).
- [15] Jinsook Lee et al. “The life cycle of large language models in education: A framework for understanding sources of bias”. In: *British Journal of Educational Technology* 55.5 (2024), pp. 1982–2002.
- [16] Ruosen Li, Ziming Luo, and Xinya Du. “FG-PRM: Fine-grained Hallucination Detection and Mitigation in Language Model Mathematical Reasoning”. In: *arXiv preprint arXiv:2410.06304* (2024).
- [17] X Li et al. “The impact of large language models on higher education”. In: *Frontiers in Education*. Vol. 9. 2024.
- [18] Yihan Li et al. “Mitigating Hallucination in Large Language Models (LLMs): An Application-Oriented Survey on RAG, Reasoning, and Agentic Systems”. In: *arXiv preprint arXiv:2510.24476* (2025).
- [19] MingShan Liu and Jialing Fang. “Enhancing mathematical reasoning in large language models with self-consistency-based hallucination detection”. In: *arXiv preprint arXiv:2504.09440* (2025).
- [20] Inigo Lopez-Gazpio. “Integrating Large Language Models into Accessible and Inclusive Education: Access Democratization and Individualized Learning Enhancement Supported by Generative Artificial Intelligence”. In: *Information* 16.6 (2025), p. 473.
- [21] Zhixin Ming, Jie Li, and Jing Wang. “Deep Link Strength Prediction: Leveraging line graph transformations and neural networks”. In: *Journal of Computational Science* (2025), p. 102661.
- [22] Sagnik Mukherjee et al. “Premise-augmented reasoning chains improve error identification in math reasoning with llms”. In: *arXiv preprint arXiv:2502.02362* (2025).

- [23] Hadas Orgad et al. “Llms know more than they show: On the intrinsic representation of llm hallucinations”. In: *arXiv preprint arXiv:2410.02707* (2024).
- [24] Jialin Ouyang. “TreeCut: A Synthetic Unanswerable Math Word Problem Dataset for LLM Hallucination Evaluation”. In: *arXiv preprint arXiv:2502.13442* (2025).
- [25] Sahil Sharma et al. “The role of large language models in personalized learning: a systematic review of educational impact”. In: *Discover Sustainability* 6.1 (2025), pp. 1–24.
- [26] Marlene Steinbach et al. “When LLMs Hallucinate: Examining the Effects of Erroneous Feedback in Math Tutoring Systems”. In: *Proceedings of the Twelfth ACM Conference on Learning@ Scale*. 2025, pp. 139–150.
- [27] Shen Wang et al. “Large language models for education: A survey and outlook”. In: *arXiv preprint arXiv:2403.18105* (2024).
- [28] Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. “Hallucination is inevitable: An innate limitation of large language models”. In: *arXiv preprint arXiv:2401.11817* (2024).
- [29] Lixiang Yan et al. “Practical and ethical challenges of large language models in education: A systematic scoping review”. In: *British Journal of Educational Technology* 55.1 (2024), pp. 90–112.
- [30] Fan Yang and Xingyue Huang. “Theoretical insights into line graph transformation on graph learning”. In: *arXiv preprint arXiv:2410.16138* (2024).

Exploring the Impact of Slide vs. Poster Presentations on Student Engagement in a Data Analytics Class*

Julia Olivieri
Computer Science
University of the Pacific
Stockton, CA 95211
jolivieri@pacific.edu

Abstract

Effective oral presentation skills are essential for aspiring data analysts, yet they are often underdeveloped at the undergraduate level. This paper compares poster presentations and slide presentations using survey responses from 34 students in a data analytics class. The two formats are evaluated across several dimensions, including presenter and viewer engagement, perceived learning as both presenter and audience member, and levels of presentation-related anxiety. The results suggest that poster presentations may offer a more engaging and lower-stress alternative for showcasing data analytics projects.

1 Introduction

Clear and effective communication of results is one of the most important skills for aspiring data analysts [14, 16]. In one study, every data analytics job posting reviewed listed “communication” as a required skill [3]. Yet faculty often report feeling less confident teaching these “soft” skills than teaching technical content [6]. At the same time, growing concerns about AI-assisted cheating

*Copyright ©2026 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

have made it increasingly important to rethink how students demonstrate their understanding of class material. As a result, many instructors are turning to oral assessments as alternatives to traditional written work [18, 13, 2, 12]. Oral presentations are known to be effective learning tools, with their own sets of benefits and drawbacks [7, 10, 18]. Stephenson et. al. found that while undergraduate students tend to believe that oral assessments are more useful than written examinations, oral assessments also tend to make them feel more stressed and anxious [18].

Oral presentations in the college classroom most commonly take the form of students taking turns presenting slide decks to the whole class (hereafter referred to as “slide presentations”), perhaps with some time for questions at the end. However, opportunities for engagement are limited in this format: not all audience members can interact directly with the presenter due to time constraints, and presenters often follow a predetermined script.

Poster presentations offer a different structure. Rather than presenting once to the entire class, students engage in multiple short, individualized conversations with viewers, which has been shown to increase interaction and participation [6, 17]. Poster presentations have also been associated with lower stress and anxiety compared to other assessment types [9], and have been promoted as effective alternatives to written assessments in mathematics, computer science, and engineering courses [20, 11]. More broadly, prior work suggests that incorporating multiple presentation formats within a course can support the development of students’ communication skills [4, 5]. Although there have been some studies directly comparing the advantages of poster and slide presentations, most of this prior work focuses on presentations for language learners [15, 19], and has rarely been quantified extensively.

This paper compares student attitudes toward slide and poster presentations in a data analytics course, with the goal of investigating whether poster presentations can serve as an effective alternative to traditional slide-based presentations for a data analytics class.

2 Study Design

This study was conducted during the Spring 2025 semester in the class “Data Analytics Programming.” The course enrolled 36 students across all four undergraduate class years. Data Analytics Programming is a data analytics class that has only the introductory computer science course as a pre-requisite, and is often taken as an elective by computer science students (See Appendix 6.1 for the course description).

This class included two projects, the first of which culminated in slide presentations, and the second of which culminated in a poster session. For both

projects students were given the option to work in pairs, but only three pairs of students worked together for the first project, and two for the second. In each project, students presented visualizations and statistical analyses based on a dataset of their choice. The first project was in R, and required students to test statistical hypotheses using their data. The second project was in Python, and required students to train a machine learning model on their data.

For the first project, students shared their results in front of the class in the form of a slide presentation. There were 32 presentations over two 110-minute class periods. Each individual or pair was given 5 minutes to present, and questions were limited to one per presentation due to time concerns.

For the second project, students shared their results during a poster session open only to members of the class. There were 32 posters presented during the three-hour session allotted for the class's final. The first 15 minutes were dedicated to setup, followed by one hour in which half of the class presented their posters while the other half circulated and viewed the presentations. The groups then switched roles for the next hour, giving all students the opportunity to present and view posters.

At the end of the final exam period the students filled out a "presentation modes" survey comparing their experience with both presentation styles. There were 34 responses to the survey, corresponding to the 34 students who were present at the poster session. This survey included 14 multiple-choice questions (see Table 1), as well as three free-response questions: "Please share any additional details about your engagement during the slide and poster presentations. For example, what factors made you feel more or less engaged?", "Use this space to elaborate on any thoughts or insights you have about your experience with poster versus slide presentations. You may include examples or personal anecdotes," and "Please share any comments, concerns, or suggestions about how these presentation formats could be improved or better tailored for student learning."

This paper presents the results of this anonymous survey. This study was reviewed by the University of the Pacific Institutional Review Board and was determined to be exempt from further review (Protocol IRB2025-49).

3 Results

Responses to the "Presentation Modes" survey highlighted differences in student experience, engagement, anxiety, and learning between poster and slide presentations. The results are presented below, along with a discussion of student recommendations for future classes.

Table 1: “Presentation modes” multiple-choice survey questions.

Question	Response options
Have you given a slide presentation before this class?	Yes, many times; Yes, a few times; No
Have you presented a poster before this class?	Yes, many times; Yes, a few times; No
Rate your level of engagement during the poster session (as a viewer)	Not engaged at all; Slightly engaged; Moderately engaged; Highly engaged
Rate your level of engagement during the poster session (as a presenter)	Not engaged at all; Slightly engaged; Moderately engaged; Highly engaged
Rate your level of engagement during the slide presentations (as a viewer)	Not engaged at all; Slightly engaged; Moderately engaged; Highly engaged
Rate your level of engagement during the slide presentations (as a presenter)	Not engaged at all; Slightly engaged; Moderately engaged; Highly engaged
Which took more time to prepare for?	The poster presentation; The slide presentation; Both took an equal amount of time to prepare for
Which presentation style challenged you more?	The poster presentation; The slide presentation; Both challenged me the same amount
How nervous do you typically feel speaking in front of a group?	Not nervous at all; Slightly nervous; Moderately nervous; Very nervous
Which presentation style were you most nervous for?	Poster presentation; Slide presentation; The same for both
Which did you learn more from as a viewer?	Poster presentation; Slide presentation; I learned an equal amount from both
Which did you learn more from as a presenter?	Poster presentation; Slide presentation; I learned an equal amount from both
Which presentation were you most proud of?	My slide presentation; My poster presentation; I feel equally proud of both
Which presentation style would you recommend for future versions of this class?	Poster presentations; Slide presentations; One of each (like this semester)

3.1 Previous Student Experience with Slide and Poster Presentations

As expected, students tended to have more experience with slide presentations than poster presentations (see Table 2). Out of the 34 students who filled out the survey, half of them had never given a poster presentation before, while only two had never given a slide presentation. The majority of students had given a slide presentation “Many” times, but had only given a poster presentation “None” or “A few” times (21/34).

Table 2: Cross-tabulation of students’ prior experience with slide and poster presentations.

Slide presentations given	Poster presentations given		
	None	A few	Many
None	2	0	0
A few	5	3	1
Many	10	11	2

3.2 Student Engagement in Slide vs Poster Presentations

As might be expected, students reported higher levels of engagement when they were presenting versus viewing presentations, regardless of the presentation format (see Figure 1). However, engagement was consistently higher for poster presentations for both viewers and presenters. No students reported being “slightly engaged” while presenting a poster, compared with five students who reported being “slightly engaged” while presenting slides. The contrast was even more pronounced for viewers: 13 of the 34 respondents reported being “slightly engaged” or “not engaged at all” while viewing slide presentations, compared to only 2 of 34 being “slightly engaged” while viewing poster presentations.

In freeform responses, several students noted that the one-on-one nature of the poster session naturally led to higher levels of engagement compared to a slide presentation to the whole class. As one student said, “I felt more engaged since we were kind of forced to talk to others, but it was an overall positive experience!” Another student said “When you are right next to the presenter you should naturally feel the need to be engaged or else you would feel disrespectful so posters really lock you in.” Several students noted that their engagement during the poster session was hampered by difficulty hearing presenters due to the room’s poor acoustics, and some reported that their attention waned toward the end of the session.

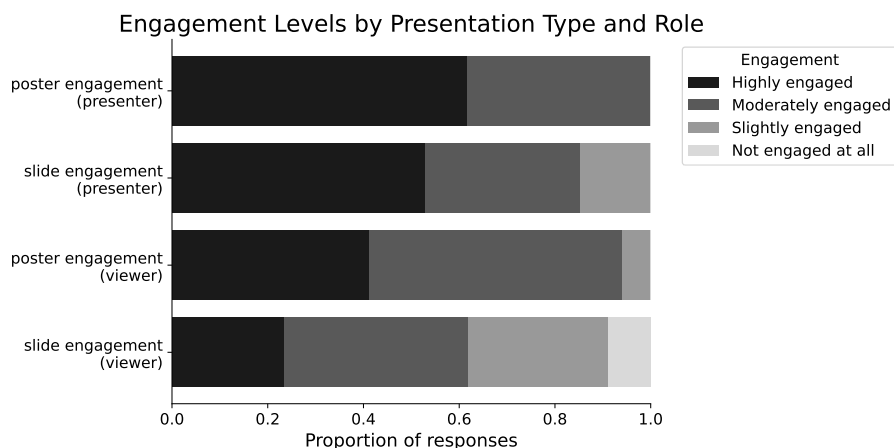


Figure 1: Students were more engaged as a presenter than a viewer regardless of presentation type. However, poster viewers had higher levels of moderate and high engagement compared to slide viewers.

3.3 Student Anxiety in Slide vs. Poster Presentations

Despite its importance across disciplines, previous studies have found that over 50% of undergraduates have a fear of public speaking [8]. Varied opportunities to practice public speaking, including lower-stakes opportunities, could help students become comfortable with this essential skill.

Previous work suggests that poster presentations may lessen the presenter's anxiety [15]. We see the same trend in our survey results: 17/34 students identified as being more nervous for the slide presentation compared to 10/34 for the poster presentation. Breaking these numbers down by students' experience with poster presentations gives more context: For students who had never presented a poster before, the same number of students reported being more nervous for the poster presentation as the slide presentation. However, among students with any experience giving poster presentations, almost all students were more nervous for the slide presentation (10/17) compared to the poster presentation (3/17, see Figure 2). Notably, students reported being more nervous for the slide presentation than the poster presentation regardless of their typical level of anxiety when speaking in front of a group (see Appendix 6.2). Although these results are intriguing, the small sample size prevents us from drawing specific conclusions.

In their free responses, students discussed being less stressed when presenting their posters because they were presenting in front of a smaller number

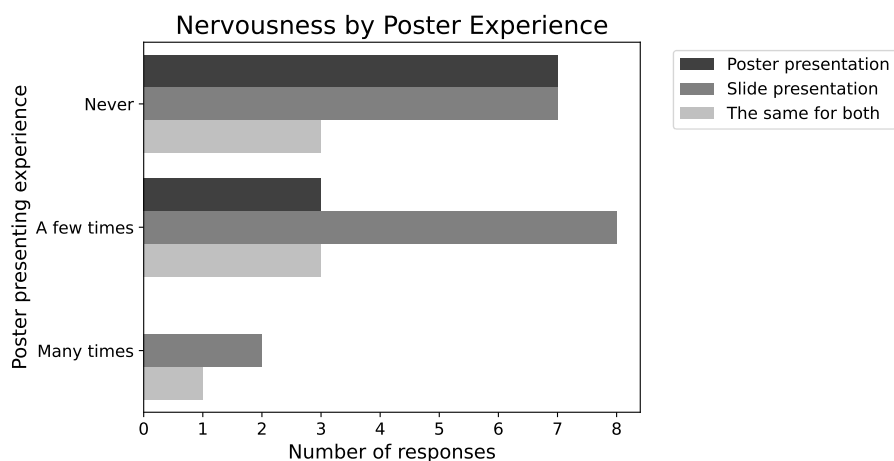


Figure 2: Students who had never given a poster presentation before were equally split between being more nervous for the slide presentation and the poster presentation. However, students who had any experience presenting a poster were more likely to be more nervous to give a slide presentation than a poster presentation.

of people, the presentations were more casual, and they were surrounded by other people who were presenting so the attention was not exclusively on them. One student said “For the poster presentations it took a lot of stress away being more one on one and not everyone just staring at you.” Another said “I guess when you are by people who are doing the same thing it kicks off some pressure.”

3.4 Student Reflections on Preparation, Challenge, and Learning in Slide vs. Poster Presentations

Student perceptions of their engagement and nervousness paint a generally favorable picture of poster presentations. However, it is also important to examine other points of comparison between the two formats. The survey included additional questions about how much time students put into their presentations, how much they were challenged, how much they learned, and which presentation they were more proud of.

Students reported putting more effort into their poster presentations, and getting more out of them, both as a presenter and a viewer. On average, they indicated that the poster presentation required more preparation time, was more challenging, and was the presentation they felt most proud of (see

Figure 3).

Although student assessment of their own learning is not always accurate [1], it is worth noting that students consistently reported learning more from the poster presentations than from the slide presentations, both as viewers (21/34 vs. 7/34) and as presenters (23/34 vs. 6/34) (see Figure 3). Given that promoting learning is a primary goal of college-level instruction, such a substantial discrepancy between presentation modes warrants attention.

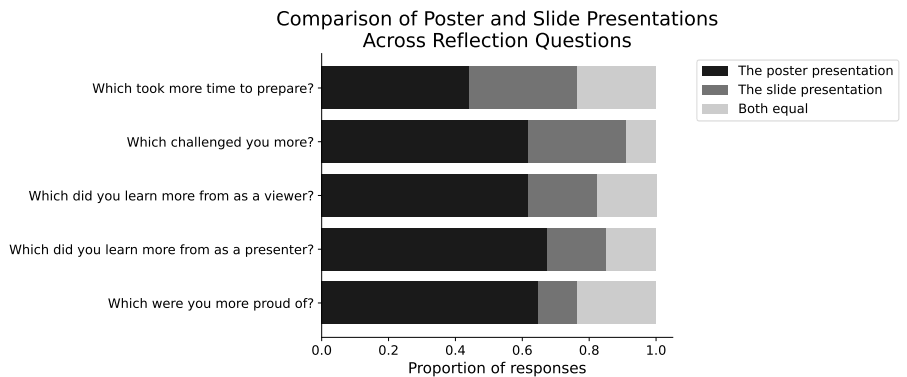


Figure 3: Students tended to spend more time on poster presentations and find them more challenging, but they also learned more from poster presentations both as a viewer and presenter, and tended to be prouder of their poster presentations.

3.5 Student Recommendations for Future Classes

When asked what they would recommend for future versions of the class, 19 students recommended having one slide presentation and one poster presentation (maintaining the current structure of the class), 14 recommended having two poster presentations, and one person recommended having two slide presentations. This indicates a desire for variety in presentation opportunities, as well as a strong preference for poster over slide presentations. As one student said, “I feel like having both platforms [was] the most beneficial because I needed to analyze my project in multiple ways.”

4 Discussion

This case study suggests that poster presentations could be a fruitful alternative to classic slide presentations as a communication tool for data ana-

lytics projects. Based on student survey responses, free-form feedback, and in-classroom observations, poster sessions have several advantages over slide presentations:

1. Poster presentations create a more casual and conversational environment, which supports higher engagement, more opportunities for questions, the development of one-on-one connections, and a reduced sense of being put on the spot.
2. Because students present their posters multiple times during a session, they gain additional fluency with their topic, experience less pressure on any single presentation, and can feel themselves learning throughout the process.
3. Poster presentations are a novel format for most students, giving them an opportunity to experience a presentation type common in the “real world” of academia, and give them an opportunity to see their work ‘on paper’ in an increasingly digital world.

Anecdotally, the unscripted nature of the poster presentations and subsequent conversations made it easier to assess students’ comprehension of the material. More work is necessary to quantify the power of poster sessions as assessment tools.

Despite all of these advantages, we acknowledge that having a full poster session is not realistic for all classes. Paying for posters can be expensive, and not everyone has access to poster boards and a large enough space for students to set up their posters and circulate. For those whom a full poster session is not realistic, we expect many of the above listed benefits could be achieved through other presentation types. For example, students could print out a small version of a poster on printer paper (across one or multiple sheets) and rotate around the room and share their projects in a similar way. The effectiveness of this “breakout session” format would be valuable to study in future work.

Although the survey results presented in this paper paint an overall favorable view of poster presentations as compared to slide presentations, there are several factors that could have biased these results. Because the poster presentation occurred after the slide presentation, students may have improved their performance based on what they learned from the first project. In addition, although both projects carried the same weight in the course grade, the second project served as the final exam and may therefore have been taken more seriously. Additionally, because the survey was distributed directly after the poster session, there could have been a recency bias in student responses favoring the poster session. Finally, the second project involved students training

machine learning models on their datasets, which they may have found more interesting than the standard statistical analyses performed in the first project.

Due to these confounding factors, as well as the small size of the study, this work should be viewed as an exploratory case study that opens up new lines of investigation rather than a conclusive verdict on presentation modes. Future research should examine whether these findings can be replicated at a larger scale across multiple courses, with improved controls to reduce confounding effects. Additionally, further work could expand the comparison to include other alternative presentation formats.

5 Conclusion

For students to be successful data analysts, they must have strong communication skills in addition to technical skills. Slide presentations are the most common mode of oral assessment in data analysis classes. However, poster sessions form a more naturalistic environment that require students to have conversational familiarity with their work. Based on the survey presented in this paper, poster sessions can provide a novel oral presentation mode that may be more engaging as a presenter and viewer, is potentially less stressful to students, and may result in products students are more proud of and learn more from. Further work is needed to determine whether these results are replicable at a large scale.

References

- [1] Heidi L Andrade. “A critical review of research on student self-assessment”. In: *Frontiers in education*. Vol. 4. Frontiers Media SA. 2019, p. 87.
- [2] Kyle Bittle and Omar El-Gayar. “Generative AI and academic integrity in higher education: A systematic review and research agenda”. In: *Information* 16.4 (2025), p. 296.
- [3] Queen E Booker et al. “Data Analytics Position Description Analysis: Skills Review and Implications for Data Analytics Curricula.” In: *Information Systems Education Journal* 22.3 (2024), pp. 76–87.
- [4] Lee Bracher. “The process of poster presentation: A valuable learning experience”. In: *Medical Teacher* 20.6 (1998), pp. 552–557.
- [5] Vincent Chan. “Teaching Oral Communication in Undergraduate Science: Are We Doing Enough and Doing it Right?.” In: *Journal of learning design* 4.3 (2011), pp. 71–79.

- [6] Evan Farber and Sara Penhale. “Using poster sessions in introductory science courses: an example at Earlham”. In: *Research Strategies* 13 (1995), pp. 55–55.
- [7] Andrea Fenton. “Reconsidering the use of oral exams and assessments: An old way to move into a new future”. In: *Educational Researcher* 54.7 (2025), pp. 430–436.
- [8] Anna Carolina Ferreira Marinho et al. “Fear of Public Speaking: Perception of College Students and Correlates”. In: *Journal of Voice* 31.1 (2017), 127.e7–127.e11. ISSN: 0892-1997. DOI: <https://doi.org/10.1016/j.jvoice.2015.12.012>. URL: <https://www.sciencedirect.com/science/article/pii/S0892199715003082>.
- [9] Lydia Fritz. “Effective Assessment for Early Courses in Computer Science: Instruments Other than Out-of-Class Programming Assignments.” In: *Journal of Instructional Research* 8.2 (2019), pp. 118–121.
- [10] Clara S Kim, Ching-Hsiu Chiu, and Tobias K Boehm. “Developing students’ collaboration and critical thinking skills via oral examination”. In: *Journal of Dental Education* 88.Suppl 3 (2023), p. 1892.
- [11] Kathleen L Kitto. “Developing and assessing conceptual understanding in materials engineering using written research papers and oral poster presentations”. In: *2008 38th Annual Frontiers in Education Conference*. IEEE. 2008, F4A–1.
- [12] Jason M Lodge. “The evolving risk to academic integrity posed by generative artificial intelligence: Options for immediate action”. In: *Tertiary Education Quality and Standards Agency* 8 (2024).
- [13] Jin Mao, Baiyun Chen, and Juhong Christie Liu. “Generative artificial intelligence in education and its implications for assessment”. In: *TechTrends* 68.1 (2024), pp. 58–66.
- [14] Patrick Mikalef et al. “The human side of big data: Understanding the skills of the data scientist in education and industry”. In: *2018 IEEE Global Engineering Education Conference (EDUCON)*. 2018, pp. 503–512. DOI: 10.1109/EDUCON.2018.8363273.
- [15] Caleb Prichard and Dan Ferreira. “The effects of poster presentations and class presentations on low-proficiency learners”. In: *TESOL journal* 5.1 (2014), pp. 172–185.
- [16] Stephanie Rosenthal and Tingting Chung. “A data science major: Building skills and confidence”. In: *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 2020, pp. 178–184.

- [17] LC Square and VP van de Heyde. “Poster presentations as an approach to implementing a ‘flipped learning’ pedagogy in introductory physics”. In: *Journal of Physics: Conference Series*. Vol. 1512. 1. IOP Publishing. 2020, p. 012005.
- [18] Zoe Stephenson, Nicole Johnson-Glauch, and Sam Cruchley. “Interventions and facilitators of oral assessment performance in higher education: a systematic review”. In: *Assessment & Evaluation in Higher Education* (2025), pp. 1–14.
- [19] Paul Tanner and Jim Chapman. “Poster presentations speak for themselves”. In: *The Language Teacher* 36.3 (2012), pp. 15–20.
- [20] Ethel R Wheland et al. “The poster session: A tool for education, assessment, and recruitment”. In: *Mathematics & Computer Education* 43.2 (2009), pp. 141–150.

6 Appendix

6.1 Course description

Data Analytics Programming: (4 units) This course develops programming skills for computational data analysis. The course emphasizes programming for statistical analysis, machine learning and predictive modeling. Other topics include programming packages for handling, preparation, and manipulation of data, as well as visualization tools for exploration and presentation of data and results. The course emphasizes hands-on data and analysis using a variety of real-world data sets and analytical objectives. Prerequisites: Completion of all Fundamental Skills; COMP 051 or COMP 061 [an introduction to computer science class].

6.2 Breakdown of student nervousness by nervousness speaking in front of a group

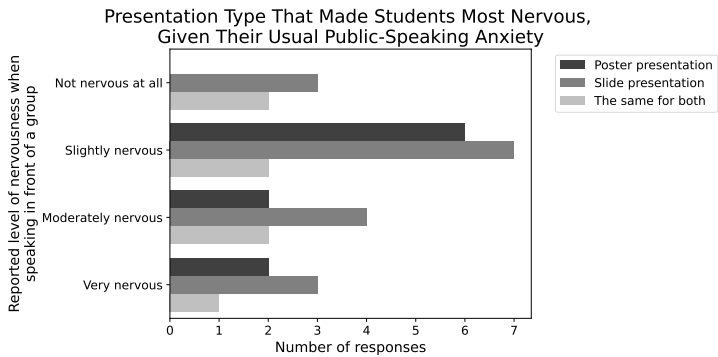


Figure 4: Regardless of typical nervousness when speaking in front of a group, students tended to be less nervous for the poster presentation than the slide presentation.

Reviewers — 2026 CCSC Southwestern Conference

Al-saadi, Oleksiy California State University Chico, Chico, CA
Attarwala, Abbas California State University Chico, Chico, CA
Dixon, Bryan California State University Chico, Chico, CA
Dodds, Zachary Harvey Mudd College, Claremont, CA
Garcia, Sat University of San Diego, San Diego, CA
Gross, Joshua California State University Monterey Bay, Seaside, CA
Grujic, Olivera California State University Stanislaus, Turlock, CA
Houston, Dan San Diego State University, San Diego, CA
Imhmed, Essa Abubaker Eastern New Mexico University, Portales, NM
Isenegger, Kathleen California State University Chico, Chico, CA
Klefsstad, Raymond University of California Irvine, Irvine, CA
Krause-Levy, Sophia University of San Diego, San Diego, CA
Ogden, Sam California State University Monterey Bay, Seaside, CA
Pang, Ashley University of California Riverside, Riverside, CA
Randolph, Tim Harvey Mudd College, Claremont, CA
Shindler, Michael University of California Irvine, Irvine, CA
Siewert, Samuel California State University Chico, Chico, CA
Thomas, Megan California State University Stanislaus, Turlock, CA