

A Visual Unplugged Activity to Introduce PDC

Mary Smith^{*}, Srishti Srivastava[†], David P. Bunde[‡], April Crockett[§], Michael Gerten[‡],
Peter Maher[¶], Jaime Spacco[‡], Xiaoyuan Suo[¶], Jiayin Wang^{||}, Michelle Zhu^{||}

^{*}*Dept. Computer Science and Engineering, Hawaii Pacific Univ., USA*

[†]*Dept. Computer Science, Univ. Southern Indiana, USA*

[‡]*Dept. Computer Science, Knox College, Galesburg, IL, USA*

[§]*Dept. Computer Science, Tennessee Tech Univ, USA*

[¶]*Dept. Math and Computer Science, Webster University, USA*

^{||}*School of Computing, Montclair State Univ., USA*

Email: *mlsmith@hpu.edu, †fsrishti@usi.edu, ‡{dbunde,mgerten,jspacco}@knox.edu,

§acrockett@tntech.edu, ¶{maherp,xiaoyuansuo51}@webster.edu, ||{wangji,zhumi}@montclair.edu

Abstract—We introduce an unplugged activity designed for CS1 students to explore fundamental parallel computing concepts. The activity requires only gridded paper and basic coloring tools, such as pens, markers, crayons, or colored pencils. It was piloted in CS1 courses across six universities, where faculty successfully incorporated the activity into various CS1 curricula taught in different programming languages. Learning outcomes were assessed through surveys and examination of student work product. Student engagement was measured using a survey that evaluated participants’ perceptions of engagement (enjoyment, participation, and focus), understanding (comprehension of the material and computing concepts), and instructor effectiveness (preparedness, enthusiasm, and availability). Qualitative student feedback was favorable, and survey results suggest the activity effectively introduced parallel and distributed computing concepts.

Index Terms—Parallel computing education, Unplugged activity

I. INTRODUCTION

The importance of teaching parallel computing to undergraduate students has also been established in existing literature [7]. The ACM/IEEE-CS 2013 curriculum report [17] and the NSF/IEEE-TCPP curriculum initiative [26] both underscore the significance of integrating parallel and distributed computing (PDC) into undergraduate Computer Science education. The 2023 report advocates for PDC as a core curricular component, while the TCPP initiative actively contributes by offering guidance on essential PDC topics and organizing training workshops for educators to effectively teach PDC concepts in introductory CS courses.

One approach used for introducing advanced concepts in early courses is the use of CS Unplugged activities, which teach CS concepts without using a computer, often by having the students play the role of computational agent. In this paper, we present an unplugged version of an existing CS1 programming assignment, flag coloring. In the existing assignment [9], introductory students practice loops by drawing flags using a library that allows them to set pixel values.

This work was partially supported by the National Science Foundation through awards OAC-2321020, OAC-2321017, OAC-2321015, and OAC-2321015.

In our unplugged version, students play the role of the processor by coloring cells of a paper grid to produce the flag. They first do this individually at first to simulate sequential processing. Then they repeat the activity collaboratively, with multiple students working together to color a single flag to simulate parallel processing. This exercise effectively demonstrates several core principles of parallel computing, but does so in a very accessible manner since students are coloring rather than dealing with the complexities of actual parallel code.

The contributions of this paper are the following:

- a description of the novel unplugged flag coloring activity, complete with advice on running it, and
- an evaluation of the activity based on its implementation in CS1 courses at six different institutions in the United States: Hawaii Pacific University (HPU), University of Southern Indiana (USI), Knox College (Knox), Tennessee Tech University (TNTech), Webster College (Webster), and Montclair State University (Montclair).

II. RELATED WORK

Prior attempts have been made to educate inexperienced students about PDC. Mullen et al. [24] offered a Massive Open Online Course (MOOC) focused on teaching high-performance computing (HPC) to professionals. While MOOCs provide accessibility, learning a complex field like PDC through online resources can be challenging. Universities have also explored incorporating PDC education into their curriculum. Lin [18] presented a study evaluating the effectiveness of teaching PDC as an elective computer science (CS) course. Their research used surveys and tests to assess student learning. Building on this work, several studies [19][8][37][30] documented the introduction of PDC into CS courses at junior, senior, and graduate levels. Further, the ACM has recommended including parallel computing as a knowledge area for the undergraduate curriculum. Significant research has explored integrating PDC concepts across the entire CS undergraduate curriculum through a modular approach [5][15][27][6][13][10]. Other research explores innovative approaches to PDC education, such as incorporating

“parallel thinking” concepts into undergraduate curricula and developing and sharing effective teaching materials for PDC courses [20][29][28][35].

Several studies [14][12][2][3][30][4][23] have observed that students encounter significant challenges in grasping both the theoretical concepts and the practical aspects of parallel programming. These challenges stem from the inherent complexity of PDC topics. These findings suggest that more effective pedagogical approaches beyond traditional lectures and programming-based instruction are necessary to enhance student learning in PDC.

Unplugged activities have proven effective in teaching core computer science concepts to younger learners [1]. Recognizing this potential, researchers have explored the use of unplugged activities to introduce PDC concepts to a larger student populations, including non-CS majors [25]. Game-based learning has also emerged as a valuable pedagogical tool. Kitchen et al. [16] and Bogaerts [2] successfully employed game-based scenarios where students assume the roles of processors and computational cores, simulating parallel computing processes. Furthermore, Maxim [22] demonstrated the effectiveness of an unplugged activity in a data structures course (CS2), where students actively participated as processes. These innovative approaches, utilizing unplugged activities and game-based learning, offer promising avenues for enhancing student engagement and understanding of complex PDC concepts. The topic of parallel activities has been studied before; Matthews compiled a repository of computing activities on this topic [21]. Ghafoor et al. [11] used the unplugged activity to introduce parallel computing concepts in the CS1/2 courses. This work resulted in a significant improvement in student learning. Recently, Srivastava et al. [33] and Smith and Srivastava [32], successfully implemented active learning unplugged modules in their respective CS1 and CS2 courses. Their work demonstrated the effectiveness of these modules in improving student engagement and fostering a deeper understanding of PDC concepts.

III. ACTIVITY DESCRIPTION

Now we describe the activity itself, starting with the core activity and then discussing variations that some of us did.

A. Core Activity

The core activity is based around four scenarios where the students color the flag of Mauritius, which has four equally-sized stripes colored red, blue, yellow, and green. Other flags can also be used, but we selected this one since it provides a natural subdivision of the task into equal-sized parts for two and four people.

The students are split into teams of size 5 (with extra students joining to create teams of size 6) or teams of size 2–3 that will merge for the later scenarios. The course staff distributes gridded paper and drawing implements (markers, crayons, and/or bingo daubers) to the students. Each team gets one drawing implement of each color (red, blue, yellow, and green). The instructor also introduces the activity: that the

students will be pretending to be a computer coloring the flag of Mauritius by filling in “pixels” of color to complete the image. Specifically, they are told that they will be taking the role of processors and introduced to the idea of the computer having multiple processing units (cores) that can each do work simultaneously.

Each scenario is explained to the entire class and the students are given time to organize their teams (mainly assigning the roles specific to the scenario). The instructor answers questions before starting all the teams coloring simultaneously. After each scenario, the instructor collects the completion time from each group, posting it publicly.

The scenarios are depicted in Figure 1, each subfigure of which is shown to the students as part of the description of that scenario. In the first scenario, one student colors the entire flag while a second student times them using their cellphone. If desired, this scenario can be repeated a second time since the first run is likely slowed down by the students being unfamiliar with the task. (We discuss the merits of repeating the first scenario in Section III-C.) In the second scenario, two students color the flag, with one student coloring the red and blue stripes while the other colors the yellow and green ones. A third student times them. In the third scenario, four students color the flag, each of them doing one stripe, while a fifth times them. Finally, in the fourth scenario, four students again color the flag, but now each of them is responsible for a vertical slice of the flag which includes part of each stripe. Since each team only has one marker of each color, this requires handing off the markers.

After all the scenarios are complete, the instructor leads a discussion about what the class observed during the activity, encouraging them toward the lessons discussed in the next section.

B. Prerequisites

This activity has very limited prerequisites. At Knox College, it was used shortly after introducing the students to the flag coloring programming activity. At this point in the term (week 3 of 9), the students were just learning about loops and had called methods, but had not even learned about conditionals. Even without specific discussion of parallel computing, the students have heard the terms “dual-core” and “quad-core” so telling them that this allows the computer to perform multiple operations simultaneously is not a big stretch. We avoided discussion of how this is managed (processes, threads, etc) and let students observe the potential (and relevant issues) through the activity, but more vocabulary could be introduced if desired.

C. Lessons to Highlight

There are a number of lessons that students articulated in the discussion after the core activity. The instructor should solicit their observations, but then lead them to any of these ideas that the students miss.

Since the instructor collects the completion time of each group for each scenario and puts them on the board, students

P1								

P1	1	2	3	4	5	6	7	8
	9	10	11	12	13	14	15	16
	17	18	19	20	21	22	23	24
	25	26	27	28	29	30	31	32
P2	1	2	3	4	5	6	7	8
	9	10	11	12	13	14	15	16
	17	18	19	20	21	22	23	24
	25	26	27	28	29	30	31	32

P1	1	2	3	4	5	6	7	8
	9	10	11	12	13	14	15	16
P2	1	2	3	4	5	6	7	8
	9	10	11	12	13	14	15	16
P3	1	2	3	4	5	6	7	8
	9	10	11	12	13	14	15	16
P4	1	2	3	4	5	6	7	8
	9	10	11	12	13	14	15	16

P1		P2		P3		P4	
1	9	1	9	1	9	1	9
2	10	2	10	2	10	2	10
3	11	3	11	3	11	3	11
4	12	4	12	4	12	4	12
5	13	5	13	5	13	5	13
6	14	6	14	6	14	6	14
7	15	7	15	7	15	7	15
8	16	8	16	8	16	8	16

Fig. 1. Scenarios students complete during the activity using the flag of Mauritius. P1 through P4 correspond to the four students completing the activity, with the numbers indicating the execution order. The scenarios are completed in the order shown (top to bottom).

were quick to point out that the times decreased as more processors were added (at least for the first 3 scenarios). Trying to quantify this naturally leads into the concept of speedup and its calculation. The question of what the speedup “should” be leads into the introduction of linear speedup.

If the first scenario was repeated a second time, the students are also quick to observe that its completion times are significantly better than in the first trial. This is attributable mainly to their getting used to the task and tools during that first run. The instructor can then make an analogy to system warmup, which causes subsequent runs of a program to be faster than the first because of factors such as caching, the system exiting

power-saving modes, and just-in-time compilation. Even if the first scenario was not repeated, the students can still be led toward this discussion by observing that the first scenario was particularly slow.

If different student groups are given different drawing instruments, they are bound to notice that some are better suited to the task. In our experience, daubers were the fastest, followed by thick markers, and then thin markers. Once the students get past the sense of unfairness, these differences also reflect an important issue in performance evaluation: technology differences matter. For example, it is not possible to compare running times on different hardware to evaluate algorithmic or system software differences. Comparisons need to be either between systems that are identical in all respects except the one being compared or they need to be “whole system”.

Comparing the third and fourth scenarios shows that the number of processors is not the only factor affecting performance. When asked to explain the difference between the results for these scenarios, the students were readily able to identify the conflict over drawing implements as the main issue; everyone needed the same color at the beginning and only one person at a time could use it. This is an example of contention, another important PDC concept. It is also possible to introduce the idea of dependencies here; certain cells are dependent on previous ones finishing because the drawing implement will be in use by someone else. It would also be possible to discuss how having extra resources would reduce the contention; the students may have even found this themselves if one group had extra drawing implements.

The fourth scenario sometimes also exposed the principle of pipelining; an effective coordination strategy is to pass the drawing implements around so that each processor gets the right one at any given moment, mimicking the movement of data through an arithmetic pipeline where the data is being passed between stages as it is needed. From pipelining, it is a small step to seeing that the pipeline takes time to fill (the processors are idle until they get the first implement).

D. Variations

Several variations of this activity are possible, and we implemented two specific ones. At Webster University, students studied the impact of more complex flag designs by coloring the French flag (equal vertical stripes of blue, white, and red) and the Canadian flag (a white background with red side stripes and a red maple leaf in the center). To assist with the latter, students were given a gridded paper with the maple leaf outlined (see Figure 2). Each flag was colored in two scenarios: one with a single student and one with three students dividing the task.

The speedup varied between the two flags. The simpler French flag saw greater efficiency gains, while the intricate maple leaf in the Canadian flag slowed progress. This allowed for a discussion of load balancing and its effect on speedup.

The instructor at Webster also used two multimedia resources as part of the discussion. The first of these were

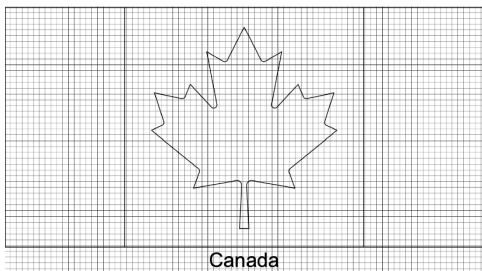


Fig. 2. Elements of the Canadian flag with superimposed grid

custom-created animations [34] to visualize schedules with different numbers of processors. These visualizations reinforced key concepts by showing the efficiency gains and potential bottlenecks when multiple processors work together.

The second multimedia resource was a video from NVIDIA meant to compare CPU and GPU execution [31]. It uses a coloring application as well, but the coloring is done by computer-controlled paintball guns. For CPU drawing, a single barrel is repeatedly aimed and fired to produce one dot at a time. The GPU example uses one barrel per pixel so that the entire image (the Mona Lisa) is drawn in a single shot. This is an extreme example of data parallelism and aligns well with our activity. Since they use the term “GPU”, it also provides an opening to talk about GPUs and how they are used for non-graphical data parallel applications.

At Knox College, the unplugged activity was preceded by students working on the flag coloring programming assignment. They had begun it several days before during a lab period and were approximately halfway through the time between the assignment’s release day and its deadline; they could all be assumed to be familiar with the premise of assigning pixel values. This, plus having slightly longer class periods (70 minutes), allowed a followup to the core activity during the same class meeting.

This follow-up activity introduced the idea of dependencies. The idea of coloring the pixels in parallel is in tension with an important technique for more complicated flags: coloring different elements of the flag in layers. For example, the flag of Great Britain (Figure 3) is most easily created by coloring the entire flag blue, then adding the crossing diagonal white lines, and then finally coloring the red vertical and horizontal lines. This approach avoids having to make complicated intersection tests between the flag’s different features. (The idea is the same as the Painter’s algorithm in 3D graphics, which renders complex scenes by drawing polygons in the order of their distance from the camera.) Unfortunately, this approach also limits parallelism by introducing dependencies: the background must be colored before the diagonals, which must be colored before the rectilinear lines. Since the students had been working on more complicated flags as part of the flag coloring programming assignment, they readily identified this issue and the difference between parallelizing the flag coloring for Mauritius and Great Britain.

To formalize this idea, the students were given the definition

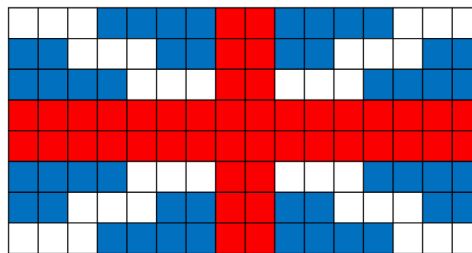


Fig. 3. Flag coloring assignment version of the flag of Great Britain

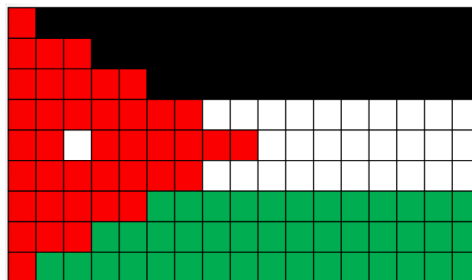


Fig. 4. Flag coloring assignment version of the flag of Jordan

of a dependency graph (vertices are tasks and directed edges denote dependencies) and asked to draw one for coloring the flag of Jordan, shown in Figure 4.

IV. PRACTICAL ADVICE FOR THE ACTIVITY

While running the activity a number of times in diverse settings, we have developed some suggestions for how to make it run most smoothly. First of all, it is important for the instructor to complete a “dry run” of the activity with other faculty or with students who are not in the class. Some of the instructions to give students are not easy to convey. This also checks that the drawing implements are appropriate (Are the markers dead? Will they bleed through the paper?). If teaching assistants or other course staff will be running the activity or assisting during the activity, they should be included so they understand student questions.

We strongly suggest projecting slides with each scenario during the activity to show the task decomposition. Number the cells to efficiently convey the order in which they should be filled, which is otherwise a tricky concept. Our images are shown in Figure 1.

We also suggest showing the students examples of properly filled cells before the activity. There was a wide variety of how well students colored the grid cells; some completely covered the paper and others added a minimal amount of color to each cell. The class as a whole moved in the later direction during the course of the activity to minimize the tedium of coloring and to reduce the time as they got competitive. We suggest taking a middle road on this, using a back and forth scribble that touches all edges of the cell, but not trying to cover it entirely. This is faster than completely filling a cell while still making it possible to achieve uniformity of time per cell. A

nice way to generate sample colored cells is to preserve the results from the instructor’s dry run.

We also feel that it is advantageous to provide students with a variety of drawing implements rather than giving them all equivalent supplies. (Originally, we made this decision by default due to a lack of sufficient supplies of a single type.) Having diverse implements does lead to some complaints in the room since it offends students’ sense of fairness, but it does show the effect of different hardware. We also found that the students preferred markers to crayons—the institution that used crayons got many complaints about them in the open-ended parts of the activity’s survey.

V. ASSESSMENT

We used various approaches to evaluate this activity, with some differences among the six participating institutions. At some of the institutions, a pre-test survey was administered before conducting the activity, followed by a post-test survey. The pre- and post-survey questions were designed to assess student comprehension of key parallel distributed computing concepts. All six institutions utilized an engagement survey based on the ASPECT (Assessing Student Perspective of Engagement in Class Tool) survey [36], which measures student engagement in active-learning exercises, including perceived effort, instructor contribution, and the activity’s value. Our survey examined three key aspects: the student experience (their engagement, enjoyment, participation, and focus), their understanding (encompassing comprehension of the material and computing concepts), and instructor effectiveness (preparedness, enthusiasm, and availability). Additionally, at one institution, students’ understanding of dependencies was assessed by collecting the dependency graphs they created while coloring the flag of Jordan.

The following sections discuss each of these measures and their results.

A. Student engagement survey

Student engagement was measured with a survey administered following the activity. This survey utilized a Likert scale ranging from 1 (Strongly Disagree) to 5 (Strongly Agree). The survey questions are presented in Figure 5. The student engagement survey was administered at all six of our institutions.

The bar chart in Figure 6 presents the median scores for each question across the various institutions. A further breakdown of the questions follow.

As shown in Table I, the questions include responses to students’ engagement in the activity, such as enjoyment, participation, and focus. Students from USI and Webster reported the highest engagement levels (mostly 5.0). Knox consistently had lower engagement scores (~4.0). Montclair and TNTech had mixed responses, with Montclair scoring lower in stimulating interest in parallel computing.

As shown in Table II the questions include the students’ perceived learning of concepts through discussion, group work, and activities. Webster and USI again show the highest

- Student Engagement Survey**
(all questions used a 5-point Likert scale 1=Strongly disagree, 5=Strongly agree)

 - Explaining the material to my group improved my understanding of it
 - Having the material explained to me by my group members improved my understanding of it
 - Group discussion during the activity contributed to my understanding of parallel computing
 - I had fun during the activity
 - Overall, the other members of my group made valuable contributions during the activity
 - I would prefer to take a class that includes this group activity over one that does not
 - I am confident in my understanding of the material presented during the activity
 - The activity increased my understanding of parallel computing
 - The activity stimulated my interest in parallel computing
 - The activity increased my understanding of loops
 - I made a valuable contribution to my group during the activity
 - I was focused during the activity
 - I worked hard during the activity
 - The instructor seemed prepared for the activity
 - The instructor put a good deal of effort into my learning from the activity
 - The instructor’s enthusiasm made me more interested in the activity
 - The instructor and/or TAs were available to answer questions during the activity
 - *I like that the activity tied into the class’s current programming assignment *

Fig. 5. Student Engagement Survey Questions

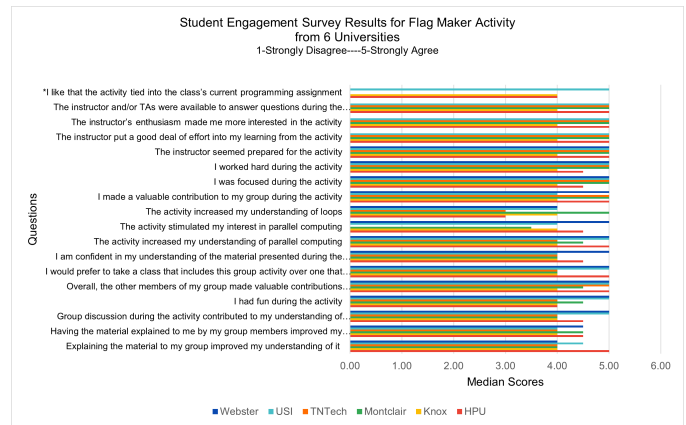


Fig. 6. Student Engagement Survey Questions

Question	HPU	Knox	Montclair	TNTech	USI	Webster
I had fun during the activity	4.0	4.0	4.5	4.0	5.0	5.0
I made a valuable contribution to my group	5.0	4.0	5.0	5.0	4.0	5.0
I was focused during the activity	4.5	4.0	5.0	5.0	5.0	5.0
I worked hard during the activity	4.5	4.0	5.0	5.0	5.0	5.0
The activity stimulated my interest in parallel computing	4.5	4.0	3.5	NA	4.0	5.0

TABLE I
MEDIAN SCORES FOR ENGAGEMENT (PARTICIPATION, ENJOYMENT, AND FOCUS)

scores (mostly 5.0). Knox and Montclair have slightly lower perceived learning (~4.0). HPU and TNTech show a lower perceived learning of loops (3.0), indicating a challenge in this area.

Question	HPU	Knox	Montclair	TNTech	USI	Webster
Explaining material to my group improved my understanding	5.0	4.0	4.0	4.0	4.5	4.0
Having material explained to me by my group improved my understanding	4.5	4.0	4.5	4.0	4.0	4.5
Group discussion contributed to my understanding of parallel computing	4.5	4.0	4.0	4.0	5.0	5.0
I am confident in my understanding of the material presented	4.5	4.0	4.0	4.0	4.0	5.0
The activity increased my understanding of parallel computing	5.0	4.0	4.5	4.0	5.0	5.0
The activity increased my understanding of loops	3.0	4.0	5.0	3.0	4.0	4.0

TABLE II
MEDIAN SCORES FOR UNDERSTANDING (COMPREHENSION OF MATERIAL AND COMPUTING CONCEPTS)

As shown in Table III, the questions assess students' perceptions of the instructor's preparedness, enthusiasm, and availability. Instructor ratings were consistently high (mostly 5.0) in all universities except Knox (4.0). The NA in Table III includes the "NA" (Not applicable) for certain questions, indicating that Webster University did not include these questions in the survey.

Question	HPU	Knox	Montclair	TNTech	USI	Webster
The instructor seemed prepared for the activity	5.0	4.0	5.0	5.0	5.0	5.0
The instructor put effort into my learning	5.0	4.0	5.0	5.0	5.0	NA
The instructor's enthusiasm made me more interested in the activity	5.0	4.0	5.0	5.0	5.0	NA
The instructor and/or TAs were available to answer questions	5.0	4.0	5.0	5.0	5.0	NA

TABLE III
MEDIAN SCORES FOR INSTRUCTOR-RELATED QUESTIONS

The student engagement survey also included two open-ended questions, asking students to share the most interesting thing they learned from the activity and suggest improvements to the activity for future classes.

1) *Summary of student comments on what was the most interesting thing they learned from the activity:* Student feedback highlighted several key takeaways from the Flag Maker activity related to parallel computing concepts. Many students commented that they better understood how parallel computing operates, particularly that adding more processors does not always result in increased efficiency. Several responses emphasized the concept of diminishing returns, noting that excessive parallelization can lead to resource contention and even slowdowns. The students also appreciated the hands-on nature of the activity, stating that it helped them visualize and better grasp parallel computing principles in a fun and engaging manner. Others mentioned learning about workload distribution, task synchronization, and coordination challenges among multiple processors. Some students reflected on the complexity of parallel processing, recognizing that effective parallelism requires careful planning and appropriate task allocation. A few students reported that they were already familiar with parallel computing concepts, while others expressed interest in applying their new knowledge to programming. In addition, some responses focused on the collaborative aspect of the activity, drawing parallels between teamwork and multiprocessor computing.

2) *Student feedback on improving the Flag Maker activity highlighted several recurring themes:* Many students requested better quality crayons or alternative coloring tools, such as markers, to avoid breakage and improve usability. Some students suggested modifying the activity structure, including making the tasks more engaging, incorporating more problem-solving elements, or integrating coding exercises to better connect with computing concepts. Others recommended making the activity shorter to avoid redundancy. Several responses emphasized the need for clearer instructions and explanations, particularly on how the activity relates to computing topics like pipelining and parallel processing. Some students requested that key vocabulary be introduced during the activity. There were also calls for larger paper sizes, improved classroom setup to enhance collaboration, and better

Pre-Post Test	
1. Task Decomposition: Which of the following best describes task decomposition?	
a) The process of breaking down a large task into smaller, independent tasks that can be executed concurrently.	
b) The method of organizing tasks in a sequential manner.	
c) The technique of reducing the number of tasks to improve performance.	
d) The strategy of assigning tasks to a single processor.	
2. Speedup: Speedup is defined as the ratio of the time taken to solve a problem on a single processor to the time taken on a parallel system.	
a) True	b) False
3. Contention: What is contention in parallel computing?	
a) The process of dividing a task into smaller subtasks.	
b) The competition between multiple processors for shared resources.	
c) The increase in computational speed by adding more processors.	
d) The ability of a system to handle a growing amount of work.	
4. Scalability: Scalability refers to the ability of a parallel system to increase its performance proportionally with the addition of more processors.	
a) True	b) False
5. Pipelining: What is pipelining in the context of parallel computing?	
a) The process of executing multiple tasks simultaneously.	
b) The technique of overlapping the execution of multiple instructions to improve performance.	
c) The method of dividing a task into smaller subtasks.	
d) The strategy of reducing contention among processors.	

Fig. 7. Pre- Post Test Questions

organization of group work to ensure smoother participation. Some students suggested making the activity more interactive, possibly incorporating a competitive element such as leaderboards or timed challenges. Finally, some students stated that the activity worked well and did not require significant changes.

B. Pre/Post Test Analysis of Student Learning

Before starting the Flag Maker activity, students from many of the universities were given a pre-test quiz consisting of five multiple-choice and true/false questions to assess their basic understanding of task decomposition, speedup, contention, scalability, and pipelining. After completing the activity, the same questions were administered in a post-test quiz. These quizzes were designed to evaluate the learning outcomes of the activity. Figure 7 presents the list of multiple choice and true/false questions on the test given to the students before and after the activity.

1) *Summary of Pre- and Post-Quiz Results Across USI, TN Tech, and HPU:* Figure 8 summarizes the pre- and post-quiz results from three universities, USI, TNTech, and HPU, assessing students' understanding of key parallel and distributed computing concepts. The analysis highlights knowledge retention, learning gains, and areas where students struggled the most. Scalability and Speedup demonstrated strong retention across institutions, reflecting a solid foundational understanding among students. Conversely, Contention and Pipelining revealed lower initial comprehension, significant incorrect retention, and knowledge loss after the quiz. These findings highlight the need for targeted instructional interventions to improve students' conceptual grasp of these parallel and distributed computing concepts.

C. Analysis of Dependency Graphs

As noted above, students at Knox were not given the pre/post test for student learning. Instead, their learning about dependencies during their followup activity was assessed by collecting dependency graphs they drew for parallel coloring of the flag of Jordan (see Figure 4). They were given the last

1. Task Decomposition

- Most students retained correct answers from pre- to post-quiz (76.9% USI, 87.2% TN Tech, 83.3% HPU).
- Minimal improvement in learning (0% growth in USI, 4.1% in TN Tech, 16.7% in HPU).
- Some knowledge loss observed in USI (23.1%) and TN Tech (6.4%).

2. Speedup Concept

- There was high initial understanding, with most students retaining correct answers (69.2% USI, 66.3% TN Tech, 100% HPU).
- Some knowledge gains (15.4% USI, 18% TN Tech).
- There was minimal reduction in TN Tech (7%).

3. Contention in Parallel Computing

- There was a lower baseline knowledge, with fewer correct pre-quiz answers (46.2% USI, 37.2% TN Tech, 33.3% HPU).
- There were significant growth post-quiz (38.5% USI, 25% TN Tech, 16.7% HPU).
- There was a high incorrect retention, especially in TN Tech (28.5%) and HPU (50%).

4. Scalability

- This question was the strongest retention across all institutions (92.3% USI, 82.6% TN Tech, 100% HPU).
- There was minimal reduction and growth, indicating prior understanding.

5. Pipelining

- There was the lowest initial understanding, with few correct pre-quiz responses (23.1% USI, 4.1% TN Tech, 50% HPU).
- The highest knowledge loss was in USI (23.1%) and HPU (50%). The majority of TN Tech students (74.4%) remained incorrect even post-quiz.

Fig. 8. Pre- and Post Quiz Results across USI, TNTech and HPU

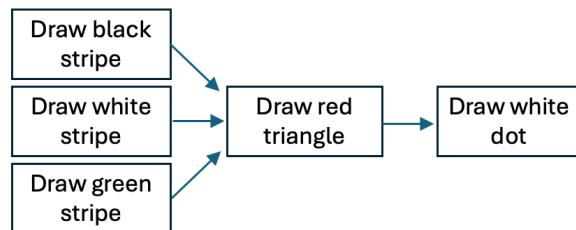


Fig. 9. Dependency graph for coloring the flag of Jordan

few minutes of the class period to complete their drawing. At the end of the period, following the procedure approved by IRB, they were asked to submit their work but told that it was voluntary and there was no effect on their grade either way.

With this procedure, we collected 29 drawings from a class of 65 (45% response rate) split into three sections. Sixty-five is the total class size because we do not know how many students attended that day. We also believe the response rate was artificially suppressed by the first of the three sections, which had less time for the drawing activity due to other parts of the activity running longer; this section only submitted 4 drawings.

All student submissions were examined to evaluate the level of student understanding demonstrated. Our intended solution for the problem is shown in Figure 9; the stripes form the first layer and must be drawn first, followed by the red triangle, and then the white dot (a star in the actual flag). We do not consider this a difficult problem, and it is similar to the dependency graph for coloring the flag of Great Britain, which was shown as an example, but completing it does demonstrate an understanding of when tasks are dependent.

When evaluating student submissions, we counted the graph as correct if it omitted the box for drawing the white stripe; in the programming version of flag coloring that students had been doing, the background is initially white so a white stripe can be achieved by not drawing anything. Some students were definitely thinking along these lines because they started with a task to draw the white stripe and crossed it out.

Another variation we saw from some students (5=14%) was splitting the red triangle into two parts. This is again consistent with how they were creating this kind of triangle in the programming assignment (split horizontally into two right triangles). The latter actually complicates the dependency graph because the top triangle should be independent of the green stripe and the bottom triangle should be independent of the black stripe. None of the students reflected this in their graph, but we still count them as “mostly correct” since the true correct answer with a split triangle is significantly more complicated than without it.

Of the submissions, 10 (34%) were perfectly correct. Seven (24%) more were mostly correct; these include the 5 mentioned above who split the triangle, one who used one task for all the stripes, and another who suggested the dependencies spatially but omitted the arrows.

The most common error for the remaining students was to give a linear chain of tasks. This suggests that they either thought about the graph in terms of sequential code or misunderstood the meaning of a dependency. There were a couple of incomplete submissions, though they all seemed to be working toward a linear solution as well. There were also a few students (4=14%) who did not demonstrate any learning; they drew the flag or started giving code to draw it.

The students who were at least mostly correct made up 59% of the respondents. Because this level was achieved with a single example, we suggest that a small amount of additional time (and examples) would suffice to teach the concept.

VI. DISCUSSION AND FUTURE WORK

Overall, the flag coloring unplugged activity provides an engaging introduction to parallel and distributed computing. The students appreciate its active nature and we believe that students seeing examples of parallel concepts in practice makes it easier for instructors to teach those concepts. Specifically, the students are exposed to speedup, system warmup, how hardware differences can make results incomparable, and the challenges presented by interprocessor communication and resource management.

While we are satisfied with the core activity, we plan to enhance the supporting components to improve student learning further. More instructors intend to incorporate the video shown at Webster, which demonstrates data parallelism through coloring. This video may become a pre-activity assignment to introduce key concepts or a post-activity reinforcement that connects data parallelism to GPU processing. Additionally, we aim to expand the discussion of dependencies, as implemented at Knox, to provide a deeper understanding while maintaining the overall brevity of the activity. Furthermore, with continued

implementation and additional data collection, we plan to conduct a more in-depth statistical analysis to identify trends, assess the activity's effectiveness, and refine strategies for improving student engagement and comprehension.

REFERENCES

- [1] Tim Bell, Ian Witten, and Michael Fellows. CS Unplugged: An enrichment and extension programme for primary-aged students, 2015.
- [2] Steven A Bogaerts. Limited time and experience: Parallelism in cs1. In *Parallel & Distributed Processing Symposium Workshops (IPDPSW), 2014 IEEE International*, pages 1071–1078. IEEE, 2014.
- [3] Steven A Bogaerts. One step at a time: Parallelism in an introductory programming course. *Journal of Parallel and Distributed Computing*, 105:4–17, 2017.
- [4] Adriano Branco, Ana Lúcia De Moura, Noemi Rodriguez, and Silvana Rossetto. Teaching concurrent and distributed computing—initiatives in rio de janeiro. In *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International*, pages 1318–1323. IEEE, 2013.
- [5] R. Brown and E. Shoop. Csinparallel and synergy for rapid incremental addition of pdc into cs curricula. In *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops PhD Forum*, 2012.
- [6] Richard Brown and Elizabeth Shoop. Modules in community: injecting more parallelism into computer science curricula. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, pages 447–452. ACM, 2011.
- [7] Davi Jose Conte, Paulo Sergio Lopes de Souza, Guilherme Martins, and Sarita Mazzini Bruschi. Teaching parallel programming for beginners in computer science. In *2020 IEEE Frontiers in Education Conference (FIE)*, pages 1–9, 2020.
- [8] Joshua Eckroth. A course on big data analytics. *Journal of Parallel and Distributed Computing*, 118:166 – 176, 2018.
- [9] Fawzi Emad. Flag coloring programming assignment. Personal communication to J. Spacco.
- [10] William B. Gardner. Should we be teaching parallel programming? In *Proceedings of the 22Nd Western Canadian Conference on Computing Education, WCCCE '17*, 2017.
- [11] Sheikh K. Ghafour, David W. Brown, Mike Rogers, and Thomas Hines. Unplugged activities to introduce parallel computing in introductory programming classes: an experience report. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '19*, page 309, New York, NY, USA, 2019. Association for Computing Machinery.
- [12] Dan Grossman and Ruth E Anderson. Introducing parallelism and concurrency in the data structures course. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, pages 505–510. ACM, 2012.
- [13] David J John and Stan J Thomas. Parallel and distributed computing across the computer science curriculum. In *Parallel & Distributed Processing Symposium Workshops (IPDPSW), 2014 IEEE International*, pages 1085–1090. IEEE, 2014.
- [14] Matthew Johnson, Robert H Liao, Alexander Rasmussen, Ramesh Sridharan, Daniel D Garcia, and Brian Harvey. Infusing parallelism into introductory computer science curriculum using mapreduce. *EECS Department, University of California, Berkeley, Tech. Rep.*, 2008.
- [15] David W Juedes and Frank Drews. Engineering a new curriculum: Experiences at Ohio University in incorporating the IEEE-TCPP curriculum initiative during a transition to semesters. In *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International*, pages 1279–1282. IEEE, 2012.
- [16] Andrew T Kitchen, Nan C Schaller, and Paul T Tymann. Game playing as a technique for teaching parallel computing concepts. *ACM SIGCSE Bulletin*, 24(3):35–38, 1992.
- [17] Amruth N. Kumar, Rajendra K. Raj, Sherif G. Aly, Monica D. Anderson, Brett A. Becker, Richard L. Blumenthal, Eric Eaton, Susan L. Epstein, Michael Goldweber, Pankaj Jalote, Douglas Lea, Michael Oudshoorn, Marcelo Pias, Susan Reiser, Christian Servin, Rahul Simha, Titus Winters, and Qiao Xiang. *Computer Science Curricula 2023*. Association for Computing Machinery, New York, NY, USA, 2024.
- [18] Hong Lin. Teaching parallel and distributed computing using a cluster computing portal. In *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International*, pages 1312–1317. IEEE, 2013.
- [19] Jie Liu. 20 years of teaching parallel processing to computer science seniors. In *Proceedings of the Workshop on Education for High Performance Computing*, pages 7–13. IEEE Press, 2016.
- [20] Suzanne J Matthews. Teaching with parallella: a first look in an undergraduate parallel computing course. *Journal of Computing Sciences in Colleges*, 31(3):18–27, 2016.
- [21] Suzanne J. Matthews. Pdcunplugged: A free repository of unplugged parallel distributed computing activities. In *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 284–291, 2020.
- [22] Bruce R Maxim, Gregory Bachelis, David James, and Quentin Stout. Introducing parallel algorithms in undergraduate computer science courses (tutorial session). *ACM SIGCSE Bulletin*, 22(1):255, 1990.
- [23] Shirley V Moore and Steven R Dunlop. A flipped classroom approach to teaching concurrency and parallelism. In *Parallel and Distributed Processing Symposium Workshops, 2016 IEEE International*, pages 987–995. IEEE, 2016.
- [24] Julia Mullen, Chansup Byun, Vijay Gadepally, Siddharth Samsi, Albert Reuther, and Jeremy Kepner. Learning by doing, high performance computing education in the mooc era. *Journal of Parallel and Distributed Computing*, 105:105–115, 2017.
- [25] Henry Neeman, Lloyd Lee, Julia Mullen, and Gerard Newman. Analogies for teaching parallel computing to inexperienced programmers. *ACM SIGCSE Bulletin*, 38(4):64–67, 2006.
- [26] Sushil K Prasad, Almadena Yu Chtchelkanova, Sajal K Das, Frank Dehne, Mohamed G Gouda, Anshul Gupta, Joseph Jaja, Krishna Kant, Anita La Salle, Richard LeBlanc, et al. Nsf/ieee-tcpp curriculum initiative on parallel and distributed computing: core topics for undergraduates. In *SIGCSE*, volume 11, pages 617–618, 2011.
- [27] Sushil K Prasad, Anshul Gupta, Krishna Kant, Andrew Lumsdaine, David Padua, Yves Robert, Arnold Rosenberg, Alan Sussman, Charles Weems, et al. Literacy for all in parallel and distributed computing: guidelines for an undergraduate core curriculum. *CSI Journal of Computing*, 1(2):82–95, 2012.
- [28] Sushil K. Prasad, Anshul Gupta, Arnold Rosenberg, Alan Sussman, and Chip Weems. *Topics in Parallel and Distributed Computing: Enhancing the Undergraduate Curriculum: Performance, Concurrency, and Programming on Modern Platforms*. Springer International Publishing, 2nd edition edition, 2018.
- [29] Sushil K. Prasad, Anshul Gupta, Arnold Rosenberg, Alan Sussman, and Chip Weems. *Topics in Parallel and Distributed Computing: Introducing Concurrency in Undergraduate Courses*. Morgan Kaufmann, 1st edition edition, August, 2015.
- [30] Erik Saule. Experiences on teaching parallel and distributed computing for undergraduates. In *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2018.
- [31] A. Savage and J. Hyneman. Art, science and GPUs. <https://www.youtube.com/watch?v=P28LKWTzrI>, 2009.
- [32] Mary L. Smith and Srishti Srivastava. Introducing parallel and distributed computing concepts through the use of flashcards and a card game. In *2023 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 278–283, 2023.
- [33] Srishti Srivastava, Mary Smith, Amrita Ghimire, and Shaun Gao. Assessing the integration of parallel and distributed computing in early undergraduate computer science curriculum using unplugged activities. In *2019 IEEE/ACM Workshop on Education for High-Performance Computing (EduHPC)*, pages 17–24, 2019.
- [34] Xiaoyuan Suo. Flag coloring activity, 2025. Accessed: Jan 31, 2025.
- [35] David Valentine. Monte carlo simulations: Parallelism in cs1/cs2.
- [36] Benjamin L Wiggins, Sarah L Eddy, Leah Wener-Fligner, Karen Freisem, Daniel Z Grunspan, Elli J Theobald, Jerry Timbrook, and Alison J Crowe. Aspect: A survey to assess student perspective of engagement in an active-learning classroom. *CBE—Life Sciences Education*, 16(2):ar32, 2017.
- [37] Ali Yazici, Alok Mishra, and Ziya Karakaya. Teaching parallel computing concepts using real-life applications. *International Journal of Engineering Education*, 32(2):772–781, 2016.