AI and Parallelism in CS1: Experiences and Analysis

Steven Bogaerts

DePauw University
602 S. College Ave, Greencastle, IN 46135, U.S.A.
stevenbogaerts@depauw.edu

Abstract

This work considers the use of AI and parallelism as a context for learning typical programming concepts in an introductory programming course (CS1). The course includes exercises in decision trees, a novel game called *Find the Gnomes* to introduce supervised learning, the construction and application of a vectorized neural network unit class, and obtaining speedup in training through parallelism. The exercises are designed to teach students typical introductory programming concepts while also providing a preview and motivating example of advanced CS topics. Students' understanding and motivation are considered through a detailed analysis of pre- and post-survey data gathered in several sections of the course each taught by one of four instructors across five semesters.

Introduction

The exploration of advanced topics or a theme in an introductory programming course ("CS1") is a common strategy for enhancing student outcomes. Such approaches can also demonstrate to students that computer science truly is more than just programming. This paper describes course materials and analyzes survey results regarding the use of AI and parallelism as a context for exploring traditional CS1 topics.

Aside from author interest and experience, AI and parallelism are selected for two reasons. First, while many ideas of parallelism are distinct from AI, the materials described in this paper culminate in the application of parallelism to improving training time of a student-developed machine learning model. Thus, these two topics support an overarching AI theme while giving introductory students experience in two advanced CS topics. Another reason for selecting AI and parallelism in this work is to explore any differences that may arise based on students' preconceived notions of the topics. Specifically, AI may be considered familiar or popular in the public consciousness, while this is arguably less true for parallelism. This conceptualization is supported by survey results presented in this paper, and is useful for drawing certain conclusions.

Challenges exist in finding the right activities within a chosen theme for CS1 students. Activities must capture the excitement of the advanced topic, but also be presented in a level-appropriate way, not burdening students with too many details or complexities. Furthermore, it is often not feasible to *replace* significant CS1 content with additional advanced-theme material unless multi-course curricular changes can be made. More commonly, the activities must enable students to learn core CS1 topics while they are presented in a *context* of other more advanced ideas.

This work aims to follow these principles: the advanced topics integrated into CS1 as described here are carefully-planned to be level-appropriate, and require no shifting of topics between courses. In fact, the materials can be provided to students with little in-class explanation. This makes them adoptable even by non-AI and parallelism specialists and without broader curricular changes.

After exploring related work in CS1, this paper describes how the above design principles are pursued. The AI materials are described first, followed by the parallelism materials and the application of parallelism to an AI task. This work then analyzes survey results for several offerings of the course by four different professors over five semesters, including some course sections in which the thematic materials were integrated into the course, and others serving as a control group in which the materials were not used.

Related Work

Work on advanced topic introductions and themes in CS1 has primarily taken place within the last couple decades (Becker and Quille 2019). Even within this relatively short time period, however, much prior work exists, with encouraging benefits. For example, both Iba (2008) and Dodds (2008) discuss AI-themed CS1 courses, presenting evidence that the AI problems considered are interesting and motivating for the students. Neller, Russell, and Markov (2008) also argue that interesting AI challenges early in the curriculum (CS1 and CS2) are very motivating, providing examples in document classification, neural networks, a logic puzzle, and games. Games, in fact, are a common domain for AI applications in CS1, as in work by Lakanen and Isomöttönen (2013) and Jonas and Sabin (2015). Numerous examples exist in robotics as well, including using AIBO dogs (Chilton and Gini 2007), surveyor SRV-1 robots (Cummins, Azhar, and Sklar 2008), and Scribbler robots (Kumar et al. 2008; Summet et al. 2009).

The incorporation of real-life data can also be a motivating factor for students. For example, Anderson et al. (2015)

use much interdisciplinary data from science, engineering, business, and the humanities in their CS1. Bart et al. (2017) also describe students' enjoyment of a significant data focus in introductory programming. That work also describes the CORGIS project in which dozens of datasets are provided pre-cleaned, to facilitate incorporation into CS1. The preparation of data for this application is further described in work by Bart et al. (2018) and Hamid (2016).

Non-AI themes abound as well. For example, Stone (2019) describes a sustainability theme and notes in particular that this practical problem-based approach was more successful than their traditional course in attracting women to continue in CS. The biology-themed CS1 of Berger-Wolf et al. (2018) also shows increased student interest in a CS1 major and more enrolled students. Media computation courses such as those described by Guzdial (2003) and Rebelsky, Davis, and Weinman (2013) have also shown greater success in attracting women. Finally, Greenberg, Kumar, and Xu (2012) describe students' great interest in a modified CS1 that includes the creation of an art portfolio.

Preparing for the First AI Project (Weeks 1-4)

While in some cases students may be able to apply brief concepts from AI at the start of the course, this work instead focuses initially on simpler applications of programming fundamentals. Weekly lab assignments correspond to CS1 concepts considered in class. In this objects-first Java CS1, the first lab tasks students with writing a very simple class. The second lab encourages students to explore objects and classes further by using provided classes to draw a geometric scene. Following that, two labs prepare students to work with trees, which are needed in the first AI project. First, students are tasked with writing simple `Advisor` and `Student` classes. An `Advisor` has `Student`s, thus students learn about composition relationships in this lab. Recursive compositions are considered in the next lab, in which students model a basic social network. A `User` of the network has `bestFriend` and `otherFriend` fields, also of type `User`. These recursive composition exercises prepare students for the decision tree structure of the first AI project, in which a tree node has other nodes as children (fields).

Exploring AI

Following the preliminary work above, students begin their use of an AI context for learning CS1 topics. This exploration of AI is focused not on in-class lecturing but rather on two weekly lab assignments and two multi-week projects. While some content could be incorporated into an in-class lecture, the materials are designed to be entirely self-contained; the instructor may simply assign them, and students can explore the corresponding concepts through reading and guided exercises. This approach is chosen to facilitate adoption by multiple CS1 instructors, as neither AI experience nor a change in in-class content is required.

Project 1: Use of Decision Trees (Week 5)

The students' first programming experience with AI occurs in the first major project of the course, occurring around week 5. The Project 1 application is a single-player game. A collection of nine royalty-free animal cartoon characters is presented, from (Mama 2018) and used according to the posted terms of use. In order to avoid any student confusion with the Java datatype `char`, these characters are referred to as "personas". In this game, the human player is presented with a grid of the nine personas, and must mentally choose one of them. The computer then asks yes/no questions of the player, ultimately determining which of the personas the player has chosen.

The personas are distinguished by eight yes/no attributes such as "Is the persona wearing a tie?" and "Is the persona's arm raised in the provided image?". Among the eight attributes, some evenly divide the personas into two groups, some split the personas into few:many ratios, and one ("Is the persona an animal?") has the same value, `true`, for each persona. The intent is to provide attributes of varying levels of utility in playing the game.

To attempt to minimize the number of questions the computer asks each game, a simple and effective strategy is to build a decision tree using ID3, greedily choosing to ask the attribute that maximizes information gain (Quinlan 1986). Intuitively, the result is that the computer asks an attribute that splits the personas remaining under consideration into two subsets of as equal size as possible. While code is provided to the students to accomplish this, the details of the algorithm are beyond the scope of the project and treated as a black box to the students.

Rather, the AI component the students must deal with includes how a decision tree, once built, can be used for classification. Students also briefly explore the random generation of trees, observing that trees of varying sizes can result, and that trees of less depth are preferable. Finally, students are exposed to applications of decision trees in other domains: manufacturing decisions in business (Magee 1964), malaria treatment (Rao, Schellenberg, and Ghani 2013), and analysis of chemistry research (Baysal, Günay, and Yıldırım 2017).

Thus, some explicit AI concepts are considered, such as what it means to train and apply a model, and that we can run experiments to consider various models and choose a preferred one. But more precisely the AI serves as a context for the traditional CS1 topics. The context points to advanced ideas, while students write a basic `Persona` class, Boolean expressions and methods to reason about `Persona` objects, and a recursive datatype `DTreeNode`. Students also practice constructing objects and calling methods by building a few decision trees by hand. Thus the AI context provides a motivation for the objects-early CS1 concepts considered so far in the course, clearly showing the value of encapsulation via multiple classes, separating code into multiple methods, and working with fields.

Lab 8: Supervised Learning Concepts (Week 8)

After project 1, students are introduced to loops and 1-dimensional arrays, building somewhat more complex applications of all the concepts learned so far. The second appearance of AI in the course then occurs in lab 8. Students are introduced to a novel game called *Find the Gnomes* to learn the core concepts of supervised learning, while practicing

Nook #	≤ 16	17	18 - 20	21 - 25	≥ 26
Gnome?	No	?	Yes	?	No

Figure 1: Conclusions for the Find the Gnomes game given `observedNooks` = [16, 20, 26, 18] and `hasGnome` = [0, 1, 0, 1].

loops and selection statements applied to 1-dimensional arrays. In the game, there are 100 *nooks* (little hiding places), numbered from 0 to 99. There are also g gnomes, where $2 \leq g \leq 100$. The value of g is unknown to the player and may vary from one game to the next. The gnomes are hiding in certain nooks; in particular, the gnomes always hide in a single set of consecutive nooks. The task of the lab is to construct a computer player for the game that, given some information about gnome presence and absence, will predict whether there is a gnome at certain other nooks.

For example, the computer player may be given parallel arrays `observedNooks` = [16, 20, 26, 18] and `hasGnome` = [0, 1, 0, 1], indicating the knowledge that nook 16 has no gnome, nook 20 has a gnome, and so on. Since the gnomes hide in a single consecutive block, we can conclude as illustrated in Figure 1. Specifically, we know that there are gnomes at nooks 18-20 inclusive, because we are told there is a gnome at 18 and 20, and gnomes hide in a single block of consecutive slots, so there must be a gnome at 19 as well. We also know that there are no gnomes at nooks ≤16 or ≥26, because we are told there are no gnomes at 16 nor 26 – so there must not be any gnomes beyond those nooks either, given that they hide in a single block of consecutive slots and we know that 18-20 is included in that block. For nooks 17 and 21-25 inclusive, though, we are uncertain based on the provided data. For this example, our knowledge is summarized in Figure 1.

This game is then framed as a supervised learning problem. (`observedNooks`[i], `hasGnome`[i]) represents the input-output pair for training example i, with the set of all training examples making the training set. A model is fit to this training set using an algorithm applicable only to this game: find the lowest-numbered nook that contains a gnome, and the highest-numbered nook that contains a gnome; predict 1 for any nook in this range, and 0 for any nook outside the range. This corresponds to predicting 0 when uncertain (for 17 and 21-25 in the above example). Of course other policies for uncertain nook numbers are also possible, and an extension of this assignment could have students try various policies, but this one is selected arbitrarily for this exercise.

This model can then be used to make predictions for other nook numbers corresponding to the same gnome placement. Continuing the above example, suppose we have testing set inputs `testNooks` = [19, 13, 24] with target outputs `testHasGnome` = [1, 0, 1]. Note that these test cases are consistent with the training set, following the rule of gnomes hiding in a single set of consecutive nooks. The model built as described above will correctly classify 19 as 1 and 13 as 0. It will guess that 24 is 0 as well, but will be incorrect in this case, resulting in 66% accuracy on this testing set.

Through this game, students are exposed to the core concepts of supervised learning, including training and testing sets, fitting a model, predicting on new examples, and the goal of model generalization to prediction for unseen input queries. The game and the corresponding models are very simple, making these advanced topics understandable in the time it takes to complete a single lab assignment. Of course, as in Project 1, the primary purpose is to provide a motivating context in which students can practice standard CS1 programming topics. In this lab, students practice working with 1-dimensional arrays, loops, and selection statements.

Lab 9: Matrix Operations on Datasets (Week 9)

Following an introduction to 2-dimensional arrays, the supervised learning concepts from Lab 8 are reinforced and extended as students learn in Lab 9 how to represent a dataset with multiple input attributes as a matrix. With that concept established, the remainder of the lab gives students practice with 2-dimensional arrays: nested loops, selection statements, matrix construction, and accumulation of values. The exercises are chosen with an eye towards Project 2, in which students build a single unit of a neural network. In Lab 9, students create methods for basic arithmetic on matrices: summing all values, multiplying each by a scalar value, summing corresponding elements of two matrices, transposition, and elementwise application of the sigmoid function $g(x) = 1/(1 + e^{-x})$.

Project 2: A Neural Network Unit (Week 10)

This project builds upon the supervised learning concepts of Lab 8 and matrix representation of datasets of Lab 9. After writing one more matrix method (matrix multiplication from provided pseudocode), students are tasked with building a neural "network" consisting of just a single unit. In this process, a few additional concepts are presented, such as unit parameters $\boldsymbol{w}$ (weight vector) and b (bias scalar). Ultimately the students' code is applied to reasonably successful classification in two real-world datasets.

While a derivation is beyond the scope of CS1, the students are presented with the following vectorized batch processing formulas for one epoch of training, ultimately updating unit parameters $\boldsymbol{w}$ and b:

$$\boldsymbol{A} = g\left(\boldsymbol{w}^T \times \boldsymbol{X} + b\right) \tag{1}$$

$$\boldsymbol{dZ} = \boldsymbol{A} - \boldsymbol{Y} \tag{2}$$

$$\boldsymbol{dw} = \frac{1}{m}\left(\boldsymbol{X} \times \boldsymbol{dZ}^T\right) \tag{3}$$

$$db = \frac{1}{m}\left(\boldsymbol{dZ}.sumAll()\right) \tag{4}$$

$$\boldsymbol{w} = \boldsymbol{w} - \alpha \cdot \boldsymbol{dw} \tag{5}$$

$$b = b - \alpha \cdot db \tag{6}$$

for sigmoid fuction g, $\boldsymbol{X}$ and $\boldsymbol{Y}$ the input and output matrices forming the training set, m the number of training examples, and α the learning rate. This vectorized batch processing formulation of neural network training is presented for two reasons: 1) to give students a taste of the techniques

used in deep neural networks, and 2) to provide students with an interesting context in which to practice working with 2-dimensional arrays, objects, and method invocation.

For example, consider Equation (1) above. Once the meaning of each variable and operator is established, students can implement it in code, and the exercise takes its true form – the standard CS1 topic of practicing method calls – while in the context of AI. Using matrix methods the students have written, the formula can be expressed as:

```
Matrix wT = w.transpose();
Matrix product = wT.matMult(X);
Matrix sum = product.plusScalar(b);
Matrix A = sum.sigmoid();
```

or even more succinctly without the intermediate variables. Students follow a similar process to implement fit and predict for their unit, along with a `compare` method that compares testing set predictions with the target values.

After provided basic testing, students apply their single-unit model to two real-world datasets. First is the Wisconsin breast cancer dataset (Street, Wolberg, and Mangasarian 1993); prior to providing the dataset to the students, it is cleaned by dropping the "Sample code number" attribute, standardizing all other input attributes, and converting the output attribute from values 2 and 4 to 0 and 1. Second is the Titanic survivors dataset (Kaggle 2012); prior cleaning includes dropping the "PassengerId", "Name", "Ticket", and "Cabin" attributes, one-hot-encoding the "Embarked" attribute, converting "Sex" to numerical values, and standardizing all other input attributes.

Parallelism (Week 12)

In addition to the AI material above, concepts of parallelism are considered around week 12. They include both "unplugged" activities and programming exercises. The unplugged activities consist of analogies and non-programming exercises that introduce students to the core ideas of parallelism. Activities start with a discussion on the speed-up and overhead of multiple people working together on a task. An exercise then considers how to count people in the building according to various task decomposition and communication strategies. Next, race conditions and locks are considered with analogies to the conch shell from Lord of the Flies by William Golding. This is followed by an analogy of a parent and child picking flowers to illustrate a join. These discussions take place over the course of a few days of class. For more details on these unplugged activities, see (Bogaerts 2017).

The parallelism programming exercises take place as a lab assignment, and is a direct continuation of the Project 2 neural network implementation described above. The most important operation in the neural network training algorithm is matrix multiplication. Thus, in this activity, students learn some basics of multithreaded programming in Java and apply this to a very simple parallel matrix multiplication strategy. In this strategy, one of the matrices to be multiplied is split into two halves (top and bottom), with a distinct thread handling each half. The overhead of parallelism makes this approach sometimes slower for small matrices (e.g. a square matrix of size 100x100 or less), but for larger matrices, parallelism is clearly faster. For example, on a modest laptop similar to a typical student laptop at time of writing, 30x30 matrices are multiplied in a few milliseconds, with it being a toss-up which approach is faster. 300x300 matrices, however, show the parallel approach taking nearly half the time of the sequential approach (e.g. 62ms versus 120ms).

Experimental Setup

The materials described above were integrated into multiple offerings of CS1, henceforth called the *alternative* course. Courses that did not use the materials will be called the *original* course. The original and alternative courses cover the same material except for the concepts of AI and parallelism conveyed in the new materials. Both types of courses use Java in an objects-first approach.

This study considers original and alternative CS1 courses offered from spring 2020 through spring 2022 (five semesters). An IRB-approved, anonymous, non-paired pre-survey (start-of-semester) and post-survey (end-of-semester) were completed by assenting students. The questions on the pre- and post-survey are the same. The first question asks "According to your best guess right now, what are your future plans for computer science (CS) study?" with options of intending to major, minor, take another course or two, or take no additional courses. The second question asks about prior experience, with options essentially divided among "none", "some" (less than an Advanced Placement (AP) course amount of material), or "much" (an AP course or other significant experience). The remainder of the survey consists of questions on computer science (as a whole), artificial intelligence, and parallelism. In each of these categories, the same four statements are provided:

- This topic is important to me.
- I am interested in this topic.
- I could explain some key ideas about this topic.
- I would like to learn more about this topic.

Responses are on a Likert scale with 1 labeled "strongly disagree" and 5 labeled "strongly agree".

Data collection was challenging in some of these semesters, with emergency remote instruction due to COVID in spring 2020 and some remote instruction through the 2020-2021 academic year. Thus, response rates are low in some semesters, and any differences from one semester to the next are uncontrolled variables. Nevertheless, by aggregating across semesters in various ways, some interesting evidence is observed, as discussed in the next section.

Each CS1 offering in this study was taught by one of four instructors, here labeled $\mathbb{A}$, $\mathbb{B}$, $\mathbb{C}$, and $\mathbb{D}$. $\mathbb{A}$ indicates the author of this study and of the described AI and parallelism materials, for whom data is collected for three semesters of the alternative course. For $\mathbb{B}$, data is collected for two semesters of the original course, and two subsequent semesters of the alternative. $\mathbb{C}$ and $\mathbb{D}$ chose to offer only the original course during the study period, thus providing additional control data.

In analysis, various statistical tests are applied. One question is whether to use a t-test or the Mann-Whitney-Wilcoxon test on these Likert scale items. Given statistics research showing similar power results for the two tests on Likert scale responses (e.g., (de Winter and Dodou 2010)), t-tests are applied in this research. F-tests show that on nearly every survey item for every comparison, the two groups have equal variance, thus the Student's t-test is conducted in these cases. For the few items with unequal variance, Welch's t-test is conducted. In both cases, the conducted test is two-tailed with independent samples for $\alpha = 0.05$. To measure effect size, Hedges' g is used, since in all comparisons the population sizes of the two groups differ; commonly, values of 0.2, 0.5, and 0.8 are considered small, medium, and large effect sizes, respectively (Rosenthal et al. 1994). Finally, the adjusted p formulation (Yekutieli and Benjamini 1999) of the Benjamini-Hochberg method (Benjamini and Hochberg 1995) is applied to the p values to account for multiple-hypothesis testing.

Results

Table 1 shows selected results regarding understanding of CS, motivation in CS, and comparisons of AI and parallelism. To illustrate the reading of the table, consider comparison *I*, the first block of rows. In this comparison, data from all instructors and all students is used. Group 1, the control group, consists of *post*-test responses for $N = 19$ students from original courses. Thus, Group 1 is labeled as "Orig Post" in the description for this comparison. Group 2, the experimental group, consists of post-test responses for $N = 75$ students from alternative courses (thus, "Alt Post").

Each comparison also lists the instructor considered (all, $\mathbb{A}$, or $\mathbb{B}$), and students considered (all, intended majors and minors (Mm), or not so intended (!Mm)). Within each comparison, sample means $\overline{x_{\cdot}}$ and variances $s^2_{\cdot}$ are provided for both groups, along with p (t-test) and Hedges' g. For any $p < 0.05$, the Benjamini-Hochberg-adjusted value p' is also reported, accounting for multiple-hypothesis testing. For CS in general, AI, and parallelism (Par), Table 1 provides these statistics for each of the four Likert scale survey items introduced above: importance (Imp), interest (Int), ability to explain (Exp), and desire to learn more (LM).

Some notation will be convenient. For example, consider the 7 statistics presented for comparison *I*, regarding CS, for the statement "I could explain some key ideas about this topic." This 7×1 portion of the table can be named succinctly with the notation [*I* CS Exp] (more generally, [*comparisonID topic surveyItem*]). Similarly, certain results are succinctly presented with notation such as ([*I* CS Exp], $p = 0.01$, $g = 0.66$, $p' = 0.07$), or just ($p = 0.01$, $g = 0.66$, $p' = 0.07$) when the context is clear.

CS Understanding

Do students in the alternative courses have greater understanding of CS in general, compared to those in original courses? Consider the results of [*I* CS Exp]. Note that the alternative group's reported ability to explain CS is significantly higher than the original ($p = 0.01$, $g = 0.66$, $p' = 0.07$), although the Benjamini-Hochberg adjustment suggests a potential false positive that warrants further study. These results might suggest that students in the alternative courses report having greater understanding of CS beyond mere programming, compared to those in original courses. However, a larger, focused study would be helpful given the insignificance of the adjusted p' value.

Comparison *II* considers only $\mathbb{B}$'s students, with group 1 being that instructor's two original courses and group 2 that instructor's two alternative courses. In this comparison, while there is an increase in ability to explain key ideas of CS ([*II* CS Exp], $\Delta\overline{x} = 0.27$), the difference is not statistically significant ($p = 0.30$, $g = 0.36$). This suggests that the majority of the difference found in comparison *I* is coming from $\mathbb{A}$'s courses. Perhaps this is explained by $\mathbb{B}$'s overall unfamiliarity with AI and parallelism. It is important to note, however, that in *pre-post* comparison of $\mathbb{B}$'s use of the integrated materials, a strong improvement is shown ([*III* CS Exp], $p < 0.01$, $g = 1.41$, $p' < 0.01$). The same pre-post comparison for $\mathbb{A}$ is also strong ([*IV* CS Exp], $p < 0.01$, $g = 1.18$, $p' < 0.01$). Interestingly, $\mathbb{B}$'s effect size is higher than $\mathbb{A}$'s here (1.41 vs. 1.18), reflecting the lower starting point for $\mathbb{B}$'s students ([*III* CS Exp], $\overline{x_1} = 2.75$) compared to $\mathbb{A}$'s ([*IV* CS Exp], $\overline{x_1} = 2.98$). This suggests that $\mathbb{B}$ successfully brought their students up from a greater deficit.

Pre-post improvement in CS Exp is maintained when considering only subsets of $\mathbb{A}$'s students according to major/minor plans (comparisons *V* and *VI*) and prior experience (results omitted for brevity). This suggests that students in alternative courses show improvement in their sense of what CS is compared to pre-survey results; this effect is maintained in subsets based on future study plans and prior experience.

CS Motivation

Do students in the alternative courses have greater motivation for CS in general, compared to those in original courses? Consider Imp, Int, and LM for [*I* CS]. While the positive $\Delta\overline{x}$ values might suggest some improvement over the original courses, these increases fail to achieve statistical significance. A similar lack of statistical significance was found in students' reported plans to major or minor in CS (($p = 0.72$, $g = 0.09$); results omitted from the table for brevity). Thus, there is no clear evidence that the integrated materials changed students' motivation in CS compared to students in the original course. Nevertheless, the $\overline{x_2}$ values for Imp, Int, and LM are high, showing that students remained motivated - just not significantly more motivated than those in the original courses. This lack of statistically significant differences for Imp, Int, and LM in CS is consistent in all comparisons of Table 1.

At first glance, these appear to be negative results for the efficacy of the integrated materials. However, consider the pre-survey sample means: CS Imp, Int and LM values of $\overline{x_1}$ for comparisons *III* through *VI*. Even these pre-survey results are high, with values at least in the upper 3 range, and often well above 4. That is, students were already fairly highly motivated for CS; indeed, students not majoring or minoring in CS have many other course options under the in-

Description		CS Imp	CS Int	CS Exp	CS LM	AI Imp	AI Int	AI Exp	AI LM	Par Imp	Par Int	Par Exp	Par LM
Comp.: *I*	$\overline{x_1}$	3.89	4.16	3.68	4.05	3.84	4.26	2.47	4.32	2.68	2.89	1.42	3.42
Instructor: All	s_1^2	0.99	0.92	0.45	0.94	0.58	0.43	1.37	0.56	0.89	1.10	0.37	1.04
Students: All	$\overline{x_2}$	4.21	4.24	4.13	4.41	3.91	4.08	3.00	4.20	3.25	3.36	2.92	3.57
Group 1:	s_2^2	0.85	0.94	0.47	0.79	0.90	1.07	0.97	1.14	1.33	1.42	1.80	1.27
Orig Post, N=19	p	0.19	0.74	**0.01**	0.12	0.78	0.34	**0.05**	0.66	**0.05**	0.12	**0.00**	0.59
Group 2:	g	0.34	0.08	**0.66**	0.40	0.07	0.19	**0.51**	0.11	**0.51**	0.40	**1.21**	0.14
Alt Post, N=75	p'			0.07				0.20		0.20		**0.00**	
Comp.: *II*	$\overline{x_1}$	3.64	3.91	3.82	3.82	3.91	4.18	2.73	4.27	2.64	2.91	1.45	3.09
Instructor: $\mathbb{B}$	s_1^2	1.25	1.09	0.56	1.16	0.49	0.36	1.62	0.62	0.85	0.89	0.27	1.09
Students: All	$\overline{x_2}$	4.22	4.11	4.08	4.36	4.00	4.06	3.14	4.28	3.11	3.25	2.47	3.64
Group 1:	s_1^2	1.03	1.19	0.54	0.75	1.03	1.37	1.04	1.29	1.53	1.51	1.46	1.09
Orig Post, N=11	p	0.11	0.59	0.30	0.09	0.78	0.64	0.27	0.99	0.25	0.40	**0.00**	0.14
Group 2:	g	0.56	0.19	0.36	0.59	0.10	0.12	0.38	0.00	0.40	0.29	**0.93**	0.52
Alt Post, N=36	p'											**0.00**	
Comp.: *III*	$\overline{x_1}$	3.98	4.25	2.75	4.55	3.80	4.15	2.53	4.33	2.83	3.08	1.80	3.60
Instructor: $\mathbb{B}$	s_1^2	0.90	0.76	1.22	0.56	1.14	0.95	1.64	0.74	1.69	1.81	0.83	1.37
Students: All	$\overline{x_2}$	4.22	4.11	4.08	4.36	4.00	4.06	3.14	4.28	3.11	3.25	2.47	3.64
Group 1:	s_1^2	1.03	1.19	0.54	0.75	1.03	1.37	1.04	1.29	1.53	1.51	1.46	1.09
Alt Pre, N=40	p	0.28	0.54	**0.00**	0.31	0.41	0.70	**0.02**	0.84	0.33	0.56	**0.01**	0.88
Group 2:	g	0.25	0.14	**1.41**	0.23	0.19	0.09	**0.53**	0.05	0.23	0.14	**0.63**	0.03
Alt Post, N=36	p'			**0.00**				0.12				**0.04**	
Comp.: *IV*	$\overline{x_1}$	4.23	4.33	2.98	4.67	3.97	4.38	2.16	4.39	3.23	3.31	1.64	3.75
Instructor: $\mathbb{A}$	s_1^2	0.75	0.79	1.42	0.26	0.93	0.67	0.91	0.71	1.05	1.02	0.83	0.86
Students: All	$\overline{x_2}$	4.21	4.36	4.18	4.46	3.82	4.10	2.87	4.13	3.38	3.46	3.33	3.51
Group 1:	s_1^2	0.69	0.71	0.41	0.83	0.78	0.83	0.90	1.01	1.14	1.36	1.81	1.47
Alt Pre, N=61	p	0.89	0.86	**0.00**	0.19	0.45	0.12	**0.00**	0.16	0.47	0.50	**0.00**	0.26
Group 2:	g	0.03	0.04	**1.18**	0.30	0.16	0.32	**0.74**	0.29	0.15	0.14	**1.54**	0.23
Alt Post, N=39	p'			**0.00**				**0.00**				**0.00**	
Comp.: *V*	$\overline{x_1}$	3.91	3.97	2.84	4.47	3.66	4.03	2.03	4.06	3.03	2.97	1.59	3.38
Instructor: $\mathbb{A}$	s_1^2	0.93	1.06	1.75	0.32	1.07	0.87	0.93	0.83	1.00	1.06	0.83	0.89
Students: !Mm	$\overline{x_2}$	3.50	3.69	3.94	3.94	3.25	3.50	2.75	3.44	2.63	2.81	2.88	2.81
Group 1:	s_1^2	0.53	0.76	0.46	1.40	0.60	0.80	1.13	0.93	0.65	0.96	1.85	1.10
Alt Pre, N=32	p	0.14	0.35	**0.00**	0.11	0.17	0.07	**0.02**	**0.03**	0.17	0.62	**0.00**	0.07
Group 2:	g	0.45	0.29	**0.95**	0.65	0.42	0.58	**0.72**	**0.67**	0.43	0.15	**1.19**	0.58
Alt Post, N=16	p'			**0.00**				0.12	0.15			**0.00**	
Comp.: *VI*	$\overline{x_1}$	4.57	4.75	3.11	4.89	4.36	4.79	2.25	4.75	3.46	3.68	1.68	4.14
Instructor: $\mathbb{A}$	s_1^2	0.33	0.19	1.06	0.10	0.53	0.17	0.79	0.34	1.07	0.74	0.89	0.50
Students: Mm	$\overline{x_2}$	4.70	4.83	4.35	4.83	4.22	4.52	2.96	4.61	3.91	3.91	3.65	4.00
Group 1:	s_1^2	0.22	0.15	0.33	0.15	0.54	0.44	0.77	0.52	0.81	1.17	1.60	1.18
Alt Pre, N=28	p	0.41	0.52	**0.00**	0.50	0.50	0.11	**0.01**	0.44	0.11	0.39	**0.00**	0.59
Group 2:	g	0.23	0.18	**1.45**	0.19	0.19	0.49	**0.80**	0.22	0.46	0.24	**1.79**	0.16
Alt Post, N=23	p'			**0.00**				**0.04**				**0.00**	

Table 1: Statistics for comparisons *I* through *VI*. "Orig" courses do not use the new materials, while "Alt" courses do. "Pre" and "Post" describe the survey used. "(!)Mm" refers to (non-)majors and (non-)minors only. Likert items are: importance (Imp), interest (Int), ability to explain (Exp), and interest in learning more (LM) for CS, AI, and parallelism (Par). Several statistics are provided for every survey question: sample means $\overline{x_\cdot}$, sample variances $s_\cdot^2$, t-test p value, and Hedges' g. Results corresponding to $p < 0.05$ are in **bold**. $p = 0.05$ signifies $p < 0.05$ rounded up; $p = 0.00$ signifies $p < 0.005$ rounded down. When $p < 0.05$, Benjamini-Hochberg-adjusted p' for multiple-hypothesis testing is also reported.

stitution's graduation requirements. It may be argued, then, that to observe no statistically significant *reduction* of high values from pre- to post-survey suggests success of the materials. In short, these results suggest that student perceptions of importance of, interest in, and desire to learn more about CS likely face a ceiling effect, and thus, statistically significant increases are not observed.

AI and Parallelism

Parallelism results in the "ability to explain" (Exp) item achieve statistical significance in each of the six comparisons in Table 1. Results are very strong for $\mathbb{A}$ (e.g., ([*IV* Par Exp], $p < 0.01$, $g = 1.54$, $p' < 0.01$) and also for $\mathbb{B}$ (e.g., ([*II* Par Exp], $p < 0.01$, $g = 0.93$, $p' < 0.01$)). Thus, the overall adoptability of the materials is reflected in $\mathbb{B}$'s success. It is also interesting that students' perception of the importance of parallelism may have increased statistically significantly in ([*I* Par Imp], $p = 0.05$, $g = 0.51$, $p' = 0.20$), though the insignificant p' value suggests a possible false positive and requires further study. Across all comparisons, most of the changes in parallelism survey items from group 1 to group 2 were positive, though not statistically significant.

Consider also the AI results. The AI-Exp increases are statistically significant (by unadjusted p) in every comparison except *II*, though their significance is weaker than for parallelism, and in some cases the adjusted values suggest potential false positives. There was also one possibly significant decrease: ([*V* AI LM], $p = 0.03$, $g = 0.67$, $p' = 0.15$) (a reduction in desire to learn more about AI for $\mathbb{A}$'s students with no major/minor plans) – although the adjusted value suggests insignificance. This is perhaps not so surprising given that these students have no intention of further CS study at all; they received their preview of AI, and that was sufficient for their curiosity. Most of the other changes in AI survey items from group 1 to group 2 were negative (not positive, as in parallelism), though not statistically significant.

There are likely various influences behind these parallelism versus AI results. Regarding motivation, recall the previously stated idea that AI is more popular than parallelism in the public consciousness. This idea is supported in pre-survey results; for example, for [*III* AI Int], $\overline{x_1} = 4.15$, while for Par, $\overline{x_1} = 3.08$. These prior attitudes mean there was more room to decrease in motivation for AI, but more room to increase in motivation for parallelism. The higher results of AI over parallelism continue in the post-survey as well, through the entire Imp, Int, and LM columns (Table 1). For example, consider [*I* AI Int], $\overline{x_2} = 4.08$, while for Par, $\overline{x_2} = 3.36$. Thus, by absolute measures, students continue to be more motivated towards AI than parallelism.

In comparing responses to the Exp question for AI and parallelism, AI's increases are less statistically significant (e.g., ([*I* AI Exp], $p = 0.05$, $g = 0.51$, $p' = 0.20$) versus ([*I* Par Exp], $p < 0.01$, $g = 1.21$, $p' < 0.01$)), and in some cases AI's Exp values are lower in an absolute sense than parallelism's as well (e.g., ([*IV* AI Exp], $\overline{x_2} = 2.87$) versus ([*IV* Par Exp], $\overline{x_2} = 3.33$)). It is possible that the AI materials are simply not as effective as they could be; however, while improvement is always possible, anecodotally students reacted very well to them in both $\mathbb{A}$'s and $\mathbb{B}$'s classes, so this seems less likely to be a majority of the explanation. Rather, these results are likely influenced by the fact that the AI materials actually give a broader (yet incomplete) picture of the field than do the parallelism materials. More specifically, the AI materials contain some black boxes: students learn about decision trees, but do not study ID3 for their construction. Similarly, students implement a training algorithm for a neural network unit, but do not learn where the parameter update formulas come from. Thus, in the AI materials, students see hints about a broader area of study, while also being shown by the black boxes that they don't yet understand it fully. In contrast, the parallelism materials do not contain any such black boxes; those materials only present the most basic facts that students are then ready to apply immediately. Thus, students report less significant increases in understanding of AI than parallelism, and in some cases lower absolute understanding of AI than parallelism. On the other hand, students report higher motivation and interest in AI than parallelism. (Of course, AI also benefits from the public perceptions discussed previously.)

Future Work

So, which is preferable? Is it helpful to show students black boxes and learn more about their *use*, even if they don't have a chance to *look inside*? Or is it better to avoid black boxes, thereby necessitating a simpler treatment of the subject? A future study may provide more insight into these questions. One could imagine, for example, a comparison of two different AI integrations into CS1: one as described here (with black boxes), and one considering fewer topics but with no black boxes. The latter approach might, for example, invest much more time in learning ID3 and do no unit training, or study linear regression as a precursor to neural networks and not work with decision trees. Both approaches have significant design challenges, as discussed in the introduction of this paper: to capture the excitement of the advanced topic yet keep the material level-appropriate, all while helping students learn the traditional CS1 topics.

In other potential future work, a revised survey instrument would provide more detail about students' understanding; of course, survey revisions make comparisons to prior collected data challenging. Another potentially useful data source is grade information, including ideally one or more standard exams used in the experimental and control groups. A more detailed survey instrument regarding student attitudes and interest may also bring more insights. Given multiple instructors' commitment to the overall strategy of these integrated materials, various options could be compared in the design and delivery of the materials.

Conclusion

This work presents novel exercises for practicing typical introductory programming concepts in the context of AI and parallelism. The materials are described in detail, and questions on their effect are explored through surveys. Results suggest statistically significant increases in student's self-reports of understanding in various comparisons. Results on student motivation are influenced by a ceiling effect and prior attitudes about AI and parallelism, but show promise and lead to ideas for further study.

Acknowledgments

The author expresses sincere gratitude for the collaboration of Prof. Allana Johnson in the work behind this paper.

References

Anderson, R. E.; Ernst, M. D.; Ordóñez, R.; Pham, P.; and Tribelhorn, B. 2015. A data programming CS1 course. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, 150–155.

Bart, A. C.; Subramanian, K.; Anderson, R. E.; and Hamid, N. A. 2018. Preparing, Visualizing, and Using Real-world Data in Introductory Courses. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, 676–677.

Bart, A. C.; Whitcomb, R.; Kafura, D.; Shaffer, C. A.; and Tilevich, E. 2017. Computing with corgis: Diverse, real-world datasets for introductory computing. *ACM Inroads*, 8(2): 66–72.

Baysal, M.; Günay, M. E.; and Yıldırım, R. 2017. Decision tree analysis of past publications on catalytic steam reforming to develop heuristics for high performance: A statistical review. *International Journal of Hydrogen Energy*, 42(1): 243–254.

Becker, B. A.; and Quille, K. 2019. 50 Years of CS1 at SIGCSE: A Review of the Evolution of Introductory Programming Education Research. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, SIGCSE '19, 338–344. New York, NY, USA: Association for Computing Machinery. ISBN 9781450358903.

Benjamini, Y.; and Hochberg, Y. 1995. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)*, 57(1): 289–300.

Berger-Wolf, T.; Igic, B.; Taylor, C.; Sloan, R.; and Poretsky, R. 2018. A biology-themed introductory cs course at a large, diverse public university. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, 233–238.

Bogaerts, S. 2017. One Step at a Time: Parallelism in an Introductory Programming Course. *Journal of Parallel and Distributed Computing*, 105: 4–17.

Chilton, J.; and Gini, M. L. 2007. Using the AIBOs in a CS1 Course. In *AAAI Spring Symposium: Semantic Scientific Knowledge Integration*, 24–28.

Cummins, J.; Azhar, M.; and Sklar, E. 2008. Using surveyor SRV-1 robots to motivate CS1 students. In *Proceedings of the AAAI 2008 Artificial Intelligence Education Colloquium*.

de Winter, J. F.; and Dodou, D. 2010. Five-point likert items: t test versus Mann-Whitney-Wilcoxon (Addendum added October 2012). *Practical Assessment, Research, and Evaluation*, 15(1): 11.

Dodds, Z. 2008. AI assignments in a CS1 course: reflections and evaluation. *Journal of Computing Sciences in Colleges*, 23(6): 262–271.

Greenberg, I.; Kumar, D.; and Xu, D. 2012. Creative coding and visual portfolios for CS1. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, 247–252.

Guzdial, M. 2003. A media computation course for non-majors. In *Proceedings of the 8th annual conference on Innovation and technology in computer science education*, 104–108.

Hamid, N. A. 2016. A generic framework for engaging online data sources in introductory programming courses. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, 136–141.

Iba, W. 2008. There's Something about AI Exercises. In *AAAI Spring Symposium: Using AI to Motivate Greater Participation in Computer Science*, 44–49.

Jonas, M.; and Sabin, M. 2015. Computational thinking in Greenfoot: AI game strageties for CS1: conference workshop. *Journal of Computing Sciences in Colleges*, 30(6): 8–10.

Kaggle. 2012. Titanic - Machine Learning from Disaster. Accessed: 2020-10-30.

Kumar, D.; Blank, D. S.; Balch, T. R.; O'Hara, K. J.; Guzdial, M.; and Tansley, S. 2008. Engaging Computing Students with AI and Robotics. In *AAAI Spring Symposium: Using AI to Motivate Greater Participation in Computer Science*, 55–60.

Lakanen, A.-J.; and Isomöttönen, V. 2013. High school students' perspective to university CS1. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*, 261–266.

Magee, J. F. 1964. *Decision trees for decision making*. Harvard Business Review.

Mama, G. 2018. Cartoon Animals - Free Vector Collection. Accessed: 2020-10-11.

Neller, T. W.; Russell, I.; and Markov, Z. 2008. Throw Down an AI Challenge. In *AAAI Spring Symposium: Using AI to Motivate Greater Participation in Computer Science*, 67–73.

Quinlan, J. R. 1986. Induction of decision trees. *Machine learning*, 1(1): 81–106.

Rao, V. B.; Schellenberg, D.; and Ghani, A. C. 2013. The potential impact of improving appropriate treatment for fever on malaria and non-malarial febrile illness management in under-5s: a decision-tree modelling approach. *PLoS One*, 8(7): e69654.

Rebelsky, S. A.; Davis, J.; and Weinman, J. 2013. Building knowledge and confidence with mediascripting: a successful interdisciplinary approach to CS1. In *Proceeding of the 44th ACM technical symposium on Computer Science Education*, 483–488.

Rosenthal, R.; Cooper, H.; Hedges, L.; et al. 1994. Parametric measures of effect size. *The handbook of research synthesis*, 621(2): 231–244.

Stone, J. A. 2019. A sustainability theme for introductory programming courses. *International Journal of Modern Education and Computer Science*, 11(2): 1.

Street, W. N.; Wolberg, W. H.; and Mangasarian, O. L. 1993. Nuclear feature extraction for breast tumor diagnosis. In *Biomedical image processing and biomedical visualization*, volume 1905, 861–870. SPIE.

Summet, J.; Kumar, D.; O'Hara, K.; Walker, D.; Ni, L.; Blank, D.; and Balch, T. 2009. Personalizing CS1 with robots. *ACM SIGCSE Bulletin*, 41(1): 433–437.

Yekutieli, D.; and Benjamini, Y. 1999. Resampling-based false discovery rate controlling multiple test procedures for correlated test statistics. *Journal of Statistical Planning and Inference*, 82(1-2): 171–196.