Who Wants to Collaborate?

A Step Towards Understanding Collaboration as Choice

Matthew Bojey
Department of Computer Science
University of British Columbia
mbojey@gmail.com

Bowen Hui
Department of Computer Science
University of British Columbia
bowen.hui@ubc.ca

***Abstract*—Emphasis on 21st century skills has placed much importance on providing students with collaboration opportunities, despite the resistance by some students who prefer to work alone. In order to facilitate a flexible learning environment that fosters both individual productivity as well as collaborative problem solving, we designed a study to better understand the factors influencing students' choice to collaborate in an online setting. We developed a web-based learning software for practicing linked list exercises and conducted a study with 67 participants in a CS2 class. Our results indicate that online collaboration provides a peer learning opportunity for students with lower confidence to become more comfortable with the material. Moreover, we analyzed student data and report on the performance tradeoffs (speed vs. number of mistakes) between working collaboratively and working solo.**

Keywords—Online collaboration as choice, computer science education, linked lists

I. Introduction

Recent emphasis on 21st century skills has placed much importance on providing students with collaboration opportunities. While we believe that teamwork skills promote deeper understanding, some students still prefer working alone. Reasons reported typically include discrepancy in workload contributions, difficulties in scheduling, and personality differences [10]. In accordance with our teaching experience, research indicates that although some students express much desire for collaborative teamwork, there is an equally strong voice from students who dislike being forced to work in teams [8]. While we can potentially let students freely pick how they want to complete their work, there is an absence of studies in the literature that investigate students' attitudes and performance in these contexts. In this work, we aim to gain a better understanding of the impact on student performance when collaboration is a choice.

As educators, it is our goal to provide a learning environment that fosters both individual productivity as well as collaborative problem solving. This research is part of a larger project that seeks to answer the following research questions: What type of students prefer collaboration over working individually? Could forced collaboration positively impact students' attitudes toward a subject? Which learning contexts (e.g., topic, homework, lab assignments, year of study, individual preference) are more suited for peer collaboration rather than individual assessments? How should learning technology be designed to support students studying in both conditions? Our study here focuses on the differences in confidence and performance for first year computer science students working on linked list exercises, with the option to work collaboratively or individually.

For this purpose, we developed a web-based learning software called BALLY (Better At Linked Lists, Yay!) which provides a set of drill-and-practice exercises with automated hints support. Section 2 reviews the linked list domain and Section 3 presents the design and implementation of BALLY. As linked lists is a common topic in first and second year programming courses, this web-based system is our first contribution to the research community.

We designed a study to explore the impact of online collaboration on student confidence and performance in a CS2 class. Section 4 describes the experiment conducted with 67 participants and Section 5 presents the results. Our findings indicate that confidence indeed plays a role in a student's choice to collaborate. In accordance with student beliefs, working solo results in achieving significantly faster answers than working in teams. However, working solo results in making more mistakes. Taken together, we find that collaboration is actually more efficient. Moreover, those who chose to work collaboratively gain the confidence they did not have about the knowledge domain. This intangible outcome has important implications in the way we design and teach introductory computer science courses.

II. Background and Related Work

In this section, we review linked lists and provide an overview of related literature. We are unaware of any studies on learning linked lists in an online collaborative environment and studies on student confidence and performance when collaboration is a choice. To put our work in perspective, we review learning software for linked lists and collaborative learning in introductory computer science courses below.

A. A Brief Review of Linked LIsts

The main idea behind a linked list is that the data it stores is connected in a sequential manner. While variations exist, we focus specifically on linear, singly connected linked lists.

Figure 1 shows a box-and-arrow diagram representation of a *listed list* – a data structure made up of a sequence of *nodes*.

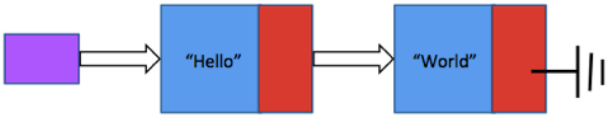

Figure 1: A visualization of a linked list with two nodes.

Each node contains some *data* (such as numbers, words, or another abstract data type) and a *pointer*. This pointer stores the address of the next node in the list, or stores the null pointer to designate the end of the list. A special node called the *list head* is used to point to the first node in the list. By following the pointers in these nodes, one can traverse a linked list from start to end.

Figure 1 shows an example of a linked list with two nodes. Here, the list head (the purple node with no data) points to the first node that stores "Hello". The first node has a pointer to the second node that stores "World", which in turn has a null pointer designating the end of the list. This visualization provides immediate feedback to students as they work through basic operations, such as inserting and deleting nodes.

To master linked lists, students must be comfortable editing these diagrams as well as writing programming statements to manipulate the list. Such operations involve a deep understanding of the underlying structure and careful planning of the code instructions. For example, if the wrong order of programming statements were used, the linked list could end up with undesired *orphaned* nodes (that is, a node that is not being pointed to and can no longer be accessed). This is a common mistake with novice programmers when learning to add a new node to a list. Thus, in the process of performing these operations, students must keep track of where all the pointers are pointing to after each programming statement is executed. This is why the visualization is so handy in helping students work through these data operations.

B. *Linked Lists Learning Software*

Due to the challenging and abstract nature of the linked list concept, existing work focuses on providing a graphical user interface for students to visualize changes made to the data structure. An example is JVALL (Java Visual Automated Linked List), a Java applet that animates code using the Java LinkedList class [3]. This program makes use of the visualization to help students become familiar with linked list concepts but it does not provide any tutorial explanations, practice exercises, or assessment means. Moreover, it does not support students in building a linked list from scratch.

An excellent tool for helping students make a solid connection between the code and the visual representation is iList [6,7]. This desktop application assumes students have mastered the basic concept and visual representation of linked lists. To practice their knowledge, students may work on one of two types of exercises: a step-by-step version where students submit a line of code at a time, or a code snippet version where students submit a chunk of code at once. Depending on the types of mistakes made, iList offers feedback about the syntax or the execution problems of the code. Successful code submissions result in an updated visual representation of the linked list. While positive learning gains were reported, due to the complexity of the system, it is unclear which aspect of iList contributed to its success.

In comparison, our system, BALLY, complements iList in that it supports novices in becoming familiar with basic linked list concepts and its visual representation *before* they have to write code to manipulate the data structure. We envision that BALLY can be further enhanced with a module that allows students to submit programming statements and see the consequential changes in the data structure.

C. *Collaborative Learning in CS1/CS2*

Many studies on using in-person cooperative and collaborative activities in introductory computer science courses have shown positive results in student engagement, performance, and retention (for example, see [1,5,11]). Of particular interest is a study involving pair programming in a CS1 course that found differences in performance between CS majors and non-CS majors [8]. This study showed that only non-CS majors under the pair programming condition had a significantly higher success rate than those working individually. In addition, they reported that 60% of all students were positive about pairing in the future. While there is some positive impact reported, we note that not all students enjoy pair programming and not all students benefitted equally. It is our ultimate goal to explore deeper into the issue of understanding the types of students that benefit more from collaborative work and how classroom activities can be personalized to these different learning styles correspondingly.

Despite much existing work in computer supported collaborative learning (CSCL), to our knowledge, there is a lack of studies done on the use of CSCL in computer science education. In contrast, we are investigating various aspects of online collaboration in computer science education. In the context of BALLY, we employ an online chat system to support online collaboration during a problem solving session.

Meanwhile, there is a large body of literature on intelligent tutoring system (ITS) where the software acts as a tutor to help a single student user with particular concepts. Traditionally, ITS has focused on issues surrounding student modeling and adaptive instruction and feedback to deliver a personalized experience for an individual student learner. Although there are new synergies to incorporate collaboration into ITS, there is no consensus as to how an ITS is best designed to accommodate both collaborative and individual needs [9]. In the future, we foresee extending BALLY with intelligent features to better support online collaboration.

III. System Design

BALLY is implemented as a web-based application using Ruby on Rails as the underlying framework. It is designed to complement class lectures and to allow students to practice linked list concepts with automated hints and immediate feedback in a series of drill-and-practice exercises (see Figure 2 for a sample screenshot of the application). User accounts were used to monitor student performance and to detect

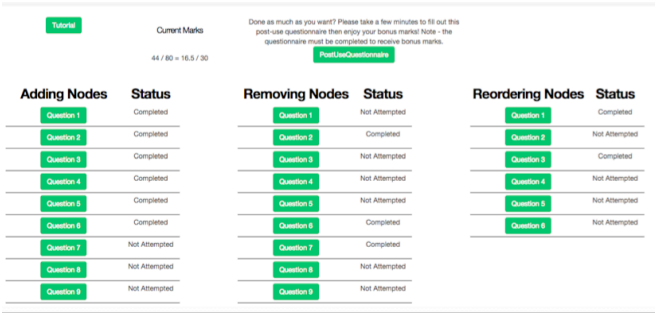

Figure 2: Screenshot showing the available exercises and their completion statuses. Students may choose any exercise to work on at any time, or they may submit a completed exercise for automatic grading to get feedback.

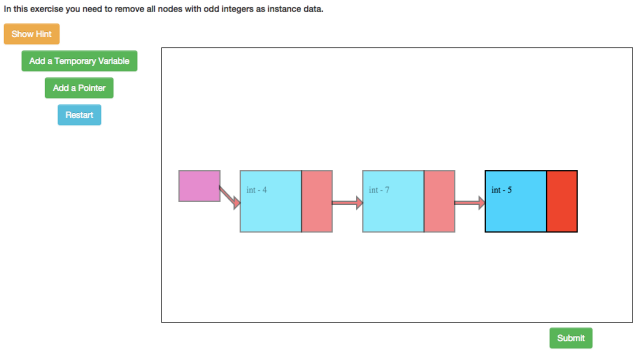

Figure 3: The student workspace when solving an exercise. Instructions for the exercise are shown at the top, with action buttons on the left-hand side.

available students for online collaboration. A rudimentary tutorial on linked lists to help users become familiar with BALLY's interface was also incorporated.

A. Linked List Exercises

At the time linked lists are introduced in the programming course, students have typically mastered loops and arrays, and are comfortable with some object-oriented concepts such as inheritance and polymorphism. Linked lists present a new challenge to students in that they must become comfortable with the concept of a data structure, learn new terminology (e.g., node, list head), visualize the list, and manipulate list components. Over the years, our experience indicate that students have the most difficulty with node deletion. Hence, we designed the exercises around this problem.

Altogether, we created three types of exercises to allow students to practice a broad range of skills with linked lists. With respect to exercises on node deletion, we included easier exercises for adding a new node to an existing list. We also included harder exercises for reordering nodes because these exercises combine the knowledge required to master node additions and deletions. In addition, a skill needed to master reordering is the use of temporary variables to keep track of parts of the list while other parts are being modified. Thus, the exercises on reordering nodes provide an indication of how well students understand node deletions and how well they are able to transfer that knowledge to another activity.

B. The Student Workspace

Once a student selects an exercise, a blank workspace appears. Instructions are presented at the top with context-dependent actions shown as buttons to the left of the workspace as shown in Figure 3. These action buttons include: add a new node, delete a node, add a next pointer, add a null pointer, and clear all.

The workspace is programmed in JavaScript using the Kinetic.js open source library. This library enables the creation and manipulation of clickable objects within a workspace called a *scene*. Each exercise has its own scene in the server and each scene has associated nodes and actions. Asynchronous JavaScript and XML (AJAX) is used to update changes made to the scene.

Students working together will see a shared workspace. When one student makes a change (e.g., adding a new node), the other will see the modification a second later after an AJAX update is made.

Each exercise has pre-defined solution criteria that BALLY uses to check for the correctness of a submitted answer. For example, if the system were checking the solution shown in Figure 3, it would start by making sure that there were three nodes in the list. Then it would check that the node contents are in order. If either of these checks failed, a notification would be given to the student indicating the type of mistake made and prompting the student to fix it.

C. Collaborative Chat

To enable online collaboration, we built a text chat system that allows students to interact with each other as shown in Figure 4. The chat interface is designed based on standard messaging interface found in common applications (such as iMessages and Facebook). Each chat message is stored in the server and AJAX is used to update the chat screens at regular intervals. Strictly speaking, this chat component is not instantaneous but asynchronous with a short delay.

IV. EVALUATION

The objective of this study is to explore the utility of collaboration on a web-based software for first year computer science students. We first report on a small usability inspection conducted in order to eliminate issues with the interface and interaction, then we report on the main study of this work.

Figure4: The chat system interface.

A. Usability Inspection

Since BALLY was built specifically for this study, we conducted a small study using usability inspection methods to remove major bugs from the system and to ensure it is easy to use. Two second-year undergraduate computer science students volunteered in the evaluation using the think-aloud protocol.

Participants were asked to individually create a new user account, go through the tutorial, do the pre-test (described in the Methodology section below under Materials), complete one exercise in each of the three categories, and do the post-test (also described in the Methodology section below under Materials). When they were both done, the participants were asked to complete one additional exercise of their choice collaboratively using the online chat system. Overall, the participants spent about 45 minutes on these tasks and an additional 15 minutes giving feedback to the researchers.

As a result, minor changes such as wording and widget layout were made. We also added an interactive sandbox mode to the tutorial so that students can see how the addition or movement of nodes and pointers works with BALLY. These changes improved usability by allowing students to focus on the exercises rather than the way the system is designed.

B. Pilot Testing

Using the system resulting from the usability inspection study, we conducted a pilot study with six participants who were undergraduate and graduate teaching assistants in a first year computer science course.

Participants were asked to create a new user account, go through the tutorial, do the pre-test, complete as many exercises as needed to receive 80 points, and do the post-test (refer to the Methodology section below regarding the pre- and post-tests). Two participants worked solo and the others worked in pairs. Individual participants completed the required tasks in about 40 minutes while collaborative participants were finished in about 55 minutes.

As a result, we fixed a few errors with the system's solution verification module so now the system is equipped with exercises that have correct solution checkers and automated hints. Furthermore, the chat system was made more responsive by decreasing the update interval and database calls.

C. Methodology

In Feburary 2015, we conducted the study in a CS2 class. Students had the choice of completing the experiment in a regularly scheduled lab or from home. We let participants self-select the experimental condition as well as their partner if they chose to work in the collaborative condition.

Due to the limited time scheduled in the lab session, we did not design a pre- and post-test to evaluate the participants' knowledge on the subject matter. Instead, we used task completion time in their first and last exercises that were successfully completed in the system as a proxy to assess their learning gain over time.

1) ***Hypotheses:*** Our experience indicates that students often avoid group work because they may have to put in extra effort to achieve the same grade they would otherwise have gotten on their own. Thus, as our first hypothesis, we expect students who choose to work collaboratively to be less confident with the subject matter than those who choose to work alone:

H1: Students who choose to work collaboratively are less confident in their knowledge with linked lists than students who choose to work solo.

Next, to assess the educational value of BALLY, we hypothesize that:

H2: Students will become better at the linked list concepts through the drill-and-practice exercises provided by BALLY.

Finally, we are interested in validating whether students working alone are faster than students working in pairs, keeping in mind that there is contrary evidence in the literature [11]. Thus, our last hypothesis is:

H3: Students working individually are faster at completing the required exercises than students working collaboratively.

2) ***Conditions:*** Two conditions were developed in this study: collaborative (pairs) and individual. As explained earlier, in order to collect data on choice, participants self-selected the experimental condition as well as their partner (if applicable).

3) ***Participants:*** The participant pool consists of students who are either required to take the computer science course as part of their major (e.g., Mathematics and Engineering) or students who have chosen to major or minor in Computer Science. A total of 67 students opted to participant in this study. Among them, 51 worked individually and 16 worked in pairs.

4) ***Materials:*** Two sets of materials are described here.

Questionnaires: A pre- and post-questionnaire containing 5-point Likert scale questions were developed. The pre-questionnaire consisted of questions related to confidence, while the post-questionnaire consisted of these same questions plus a set of questions on system usability. The pre- and post-questionnaires were presented electronically at the beginning and end of the study respectively.

Exercises: Three categories of exercises were available for students to work on: adding a new node, deleting a node, and reordering nodes. For node addition, there were nine questions available in this category. We divided these questions into three subcategories with three questions each: adding a node to the front of a list, adding a node to the middle of a list, and adding a node to the end of a list. Similarly, nine questions were made for node deletion, with three exercises in three subcategories each: removing a node from the front of a list, removing a node from the middle of a list, and removing a node from the end of a list. Lastly, six questions were available for reordering nodes, with three exercises in two

subcategories each: reversing the nodes in a list and sorting a list according to a pre-specified order. Note that to create some variation, the number of nodes in a list and the node contents were randomly generated for each exercise.

5) ***Task and Procedure:*** During recruitment, students were presented with a demonstration of BALLY and an overview of the study. The course instructor provided incentive for students to participate by giving 2% bonus marks upon completion.

At the start of the session, participants signed up to create an account, completed the pre-questionnaire, and completed the tutorial individually. Thereafter, participants begin to work on the exercises in any order they want until they are done.

Points are rewarded for every correct answer submitted to the system. Participants who work individually receive four points for each correct answer while participants who work collaboratively receive five points for each correct answer. This score difference was designed as an incentive to motivate students to work together. Each exercise also has an optional hint available. If a hint is used, a correct answer will be rewarded with two points only. The rationale for giving fewer points for using hints is that we wanted to encourage participants to find a solution on their own (or through discussion with their partners) without being overly dependent on hints. Once a participant accumulates 80 points, the experiment is completed. For example, a participant working individually would need to complete 20 exercises correctly in order to receive 2% bonus marks for the course.

Participants are allowed to attempt an exercise as many times as they wish with no penalty to their score for incorrect answers. This design decision was made because we envisioned BALLY to be used as a practice tool rather than an assessment tool.

6) ***Measures:*** Four measures were developed in this study.

Confidence: A short list of 5-point Likert scale questions were used to assess confidence with linked list concepts. These questions asked how comfortable a student is with linked lists overall, as well as their comfort level on specific operations such as finding a node, inserting a node, and deleting a node. We also asked how well they understood the box-and-arrow diagrams as a way to manipulate the lists and in connection with the code.

System Usability: A short list of 5-point Likert scale questions were asked to assess system usability in the post-questionnaire. These questions targeted the system's learnability, ease of use, enjoyment, and utility as a supplemental activity in the course. The questions also asked if a similar system would be useful for other subjects and whether BALLY would be recommended to their friends.

Number of Attempts: The number of times an answer was submitted for the same exercise. This may include a series of incorrect submissions followed by a final, correct submission.

Task Completion Time (Per Attempt): The time it takes for a participant to make an attempt at an exercise starting from the time the exercise is presented in the workspace to when an (incorrect or correct) answer is submitted for it.

V. Results

On average, participants spent 3.45 minutes to complete the pre-questionnaire, 6.23 minutes on the tutorial, 26.34 minutes solving exercises, and 7.78 minutes to complete the post-questionnaire. We first report our findings on confidence and usability, then analyze performance data under the two experimental conditions.

A. Confidence

To begin with, we were interested in the differences in confidence between the participants in the two conditions. Using the pre-questionnaire data, we confirm H1 that participants who chose to work individually were significantly more confident in their own knowledge of linked lists than those who chose to work collaboratively ($p < 0.01$). While not surprising, it is noteworthy that participants who were not as confident chose to work with a partner. As we report below, our data shows that less confidence does not necessarily translate to worse performance.

Comparing the reported scores before and after using BALLY, participants in both conditions had a significant increase in confidence ($p < 0.01$) as seen in Figures 5 and 6 respectively. These results indicate that novice users are more comfortable with the material after using BALLY.

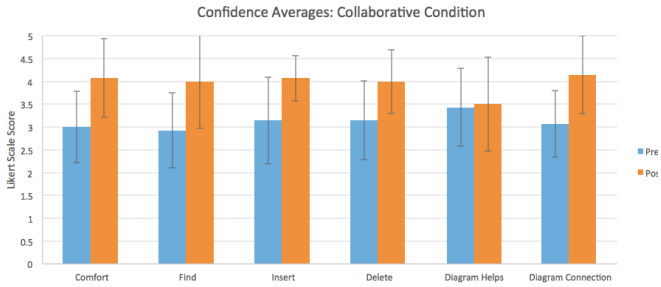

Figure 5: Average confidence scores from the pre- and post-questionnaires for participants who worked collaboratively (n=16). Overall confidence average: 3.12 (pre) and 3.96 (post).

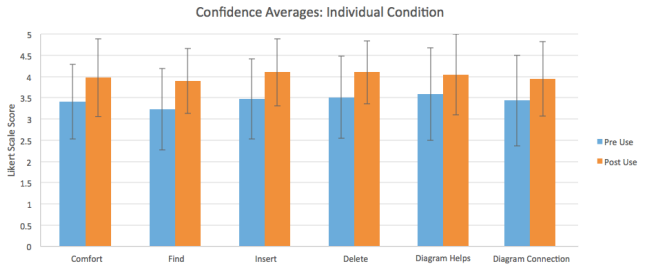

Figure 6: Average confidence scores from the pre- and post-questions for participants who worked individually (n=51). Overall confidence average: 3.44 (pre) and 4.01 (post).

By inspection, we see that the collaborative participants start off with less confidence than the individuals, but both groups end up with similar scores by the end of the study. When we compare the change in confidence across the conditions as shown in Figure 7, we see that the collaborative participants became significantly more confident than the individuals ($p < 0.01$). Indeed, aside from the stand-alone utility of the digrams, the collaborative participants felt much more confident with their knowledge on linked lists after using the system. With respect to confidence, these results show that the collaborative participants benefitted more from using BALLY than their individual counterparts.

B. System Usability

Figure 8 reports the average usability scores. Although the feedback was generally positive, individual participants found the system significantly more usable than their collaborative counterparts ($p < 0.01$). Despite having conducted a usability inspection study and a pilot study, this result suggests that the interaction for collaboration in BALLY needs improvement.

C. Performance

*1) **Number of Attempts:*** This measure describes the number of attempts a participant took to get an answer correct for a given exercise. Table 1 reports the average number of attempts made for each exercise according to the question type and the experimental condition. As expected, we see that participants have little difficulty with adding a node, but they find deleting a node harder and reordering nodes the hardest. While some participants submitted a correct answer on first try, on average, the data shows that more than one attempt was needed. In accordance with the self-reported confidence levels, this result suggests that more practice is recommended for the participants.

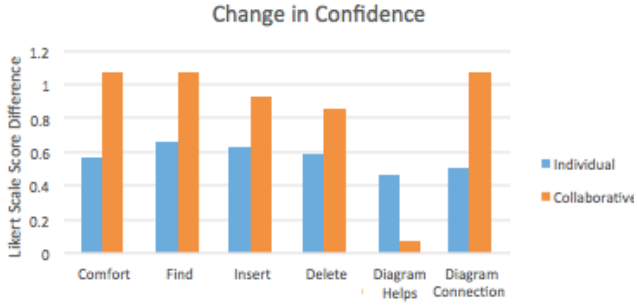

Figure 7: Average increase in confidence after using the system based on a difference in Likert scale averages. Average confidence gain for collaborative student = 0.87, average confidence gain for individual student = 0.47.

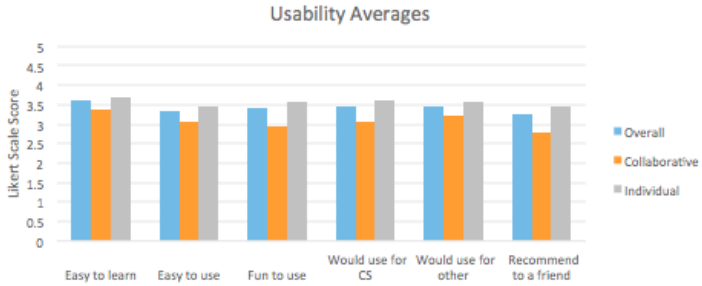

Figure 8: Average usability scores after using the system. The average score overall is 3.45, while the average score for the collaborative condition is 3.07 and the average score for the individual condition is 3.56.

Table 1: Average number of attempts by exercise type and by experimental condition.

	Individual	Collaborative
Adding a Node	1.35	1.13
Deleting a Node	1.56	1.15
Reordering Nodes	1.72	1.70
Overall	**1.52**	**1.24**

Although the collaborative participants were less confident than those in the individual condition (cf. Section 6.1), collaborative participants took significantly fewer attempts to get an exercise correct than individuals ($p < 0.05$). This finding suggests that collaboration helps students reduce mistakes and tease out misunderstandings in their learning. Similar to findings in other studies exploring the relationship between self-efficacy and performance (e.g., [2]), higher confidence does not directly translate to better performance.

*2) **Task Completion Time:*** To assess learning progress, we considered task completion time over time. Figures 9 and 10 report this data for the individual and collaborative conditions respectively. In Figure 9, we see that on average, the first time participants get an answer correct takes about 110 seconds, but the second time (on a different exercise) takes about 60 seconds, and so forth. Over time, participants are faster at getting an answer correct. On average, participants in the individual condition take 35.70 seconds to arrive at a correct answer for an exercise. A similar trend is observed in Figure 10, where participants get faster at getting answers correct. On average, participants in this condition take 53.51 seconds.

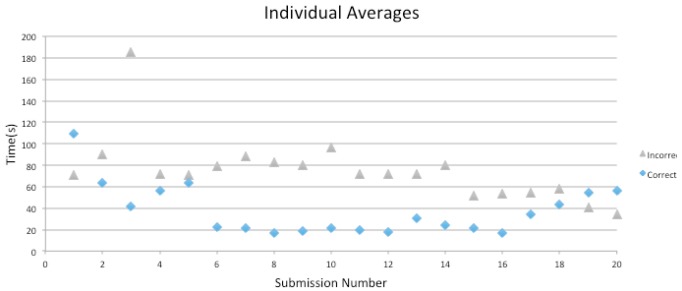

Figure 9: Average task completion time for the individual condition. The average time for incorrect answers is 79.38s and the average time for correct answers is 35.70s.

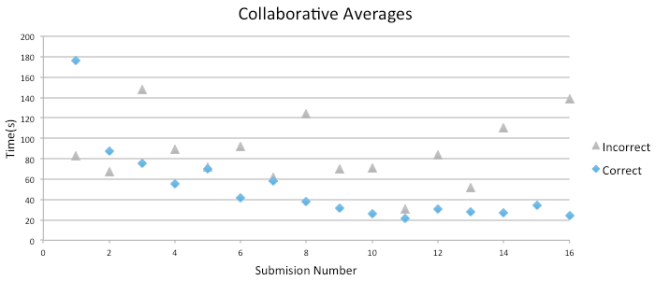

Figure 10: Average task completion time for the collaborative condition. The average time for incorrect answers is 83.39s and the average time for correct answers is 53.51s.

When comparing the task completion times at the first submission and at the last submission, we see there is a significant decrease ($p < 0.01$). This result supports H2 and suggests that participants' learning improves through the use of BALLY.

In addition, individual participants are significantly faster at getting a correct answer than those in the collaborative condition ($p < 0.01$). At first glance, this result supports H3 (however, see Section 6.3.3 for further discussion). This is not surprising since participants who work collaboratively must spend time communicating and negogiating how they want to solve the problem.

Finally, the data shows that when the participants are wrong, they usually take a long time. While there is no statistical significance in the times it takes for the participants to come up with an incorrect answer across the two conditions, we see that at times, it can take the participants twice as long to arrive at a wrong answer in the collaborative condition. This is not surprising, since we expect the collaborative participants to take time to discuss and resolve conflicts as they come across problems.

3) ***Combined Performance:*** As mentioned above, task completion times for correct answers indicate that individual participants are significantly faster than collaborative participants. However, not everyone gets an answer right on their first try. Thus, our performance comparison should consider the time it takes for one to complete an exercise, potentially get it wrong, and make corrections as necessary until a correct answer is submitted.

With consideration to correctness as well as task time, we found that participants in the individual condition (average 78.70 seconds) are actually slower than those in the collaborative condition (average 74.04 seconds), but not significantly so. Although we fail to reject the null hypothesis, we cannot accept H3. Recall that there is no consensus in the literature that indicates whether working alone or working collaboratively is more efficient [11]. While this study points to higher efficiency when working collaboratively, this result may be discipline-specific or subject-specific. Further studies to explore this issue is desired.

VI. Conclusions and Future Work

In this paper, we developed a web-based system called BALLY to let students practice linked list operations visually. By allowing students to freely choose whether they want to work on their own or work collaboratively, we found that students who had a lower confidence in the subject matter opted to collaborate. Interestingly, after using BALLY, we saw a significant increase in the participants' confidence. More importantly, we saw that collaborative participants became significantly more confident than those who worked solo. In addition, we found that while individuals were faster at completing exercises, they also made more mistakes. When considering both task completion time and correctness, collaborative participants were in fact more efficient, although not significantly so.

We believe these findings have important implications in teaching introductory computer science courses. Our main recommendation is to ensure there are collaboration opportunities available in first year courses so that students who are not as comfortable with the subject matter may choose to work with others. As we saw in our study, collaboration provides a peer environment to support students with lower confidence. In particular, the literature reports low self-confidence issues for visible minority students in STEM disciplines despite the strong academic achievements they may have [4]. As a result, these students end up dropping out of the program for the wrong reason. Therefore, as a way to help with retention issues, we propose that learning activities enabling students to work in collaboration as a choice be an integral part of introductory computer science courses.

Our next step is to replicate the study in a larger scale in order to better understand what other factors influence students to choose collaboration and the nature of their interaction. To further support students on linked lists, we would adopt the approach taken by iList and implement a code evaluation module that lets students submit Java code to manipulate a linked list visually. Ultimately, we wish to evolve BALLY into an intelligent learning environment that supports both collaborative and individual work.

Acknowledgment

We would like to thank all the participants of this project.

References

[1] L. Beck and A. Chizhik. Cooperative Learning Instructional Methods for CS1: Design, Implementation, and Evaluation. *ACM Transactions on Computing Education (TOCE)*, 13(3):Article 10, 2013.

[2] T. Busch. Gender differences in self-efficacy and attitudes toward computers. *Journal of educational computing research*, *12*(2), 147–158, 1995.

[3] H.L. Dershem, R.L., McFall, and N. Uti. Animation of Java Linked Lists. In *Proc. of the SIGCSE technical symposium on Computer Science Education (SIGCSE)*, pages 53–57, 2002.

[4] A. Duran and D. Lopez. "Women from diverse backgrounds in the science, technology, engineering, and math (STEM) professions: Retention and career development." Impact of diversity on organization

and career development, C. Hughes (Ed.). Business Science Reference. Pages 214–242, 2015.

[5] K. Falkner and E. Palmer. Developing Authentic Problem Solving Skills in Introductory Computing Classes. In *Proc. of the SIGCSE technical symposium on Computer Science Education (SIGCSE)*, pages 4–8, 2009.

[6] D. Fossati, B. Di Eugenio, C. Brown, and S. Oblsson. Learning Linked Lits: Experiments with the iList System. In *Proc. of the International Conference on Intelligent Tutoring Systems (ITS)*, pages 80–89, 2008.

[7] D. Fossati, B. Di Eugenio, C.W. Brown, S. Ohlsson, D.G. Cosejo, and L. Chen. Supporting Computer Science Curriculum: Exploring and Learning Linked List with iList. *IEEE Transactions on Learning Technologies*, 2(2):107–120, 2009.

[8] N. Nagappan, L. Williams, M. Ferzli, E. Wiebe, K. Yang, C. Miller, and S. Balik. Improving the CS1 Experience with Pair Programming. In Proc. of the SIGCSE technical symposium on Computer Science Education (SIGCSE), pages 359–362, 2003.

[9] J.K. Olsen, V. Aleven, and N. Rummel. Toward Combining Individual and Collaborative Learning Within an Intelligent Tutoring System. In *Proc. of the International Conference on Artificial Intelligence in Education (AIED)*, pages 848–851, 2015.

[10] B. Poellhuber, T. Anderson, and N. Roy. Distance students' readiness for social media and collaboration. *The International Review for Research in Open and Distributed Learning*, 12(6), 102–125, 2011.

[11] L. Williams, E. Wiebe, K. Yang, M. Ferzli, and C. Miller. In Support of Pair Programming in the Introductory Computer Science Course. *Computer Science Education*, 12:197–212, 2002.