Reflecting on the Impact of a Course on Inclusive Strategies for Teaching Computer Science

Alark Joshi
Department of Computer Science
University of San Francisco
San Francisco, California 94117
Email: apjoshi@usfca.edu

Amit Jain
Department of Computer Science
Boise State University
Boise, Idaho 83725
Email: amit@cs.boisestate.edu

***Abstract*—As the number of teachers teaching computer science grows, it is increasingly important to be mindful of the training they receive with respect to broadening participation in computer science. Through our program, we have trained over 50 teachers in the greater Boise Metropolitan region, who have in turn taught over 1400 students computing concepts through courses such as Exploring CS, AP CS Principles, and AP CS A. These courses have an excellent curriculum that contains a mix of computational thinking concepts such as a focus on creativity, abstraction, coding, as well as increasing awareness about the cyber footprint of the students with respect to security and privacy. While the curriculum is excellent, we need to be more mindful about incorporating pedagogical strategies that promote inclusive teaching especially for women and minorities who are traditionally underrepresented in computer science.**

To address the challenges associated with teaching a truly inclusive course, we developed a new course titled "Inclusive Strategies for Computer Science Education" that draws attention to the strategies that have been studied over the years in STEM and CS education literature. We present the contents of the course along with a post-hoc qualitative survey on the applicability and practicality of the material discussed in the course.

I. Introduction

With the National Science Foundation's CS 10K program [24] there was a huge push towards training teachers to teach Computer Science courses in their respective schools. The argument against introducing coding/computing in the classrooms was the paucity of trained high-quality teachers. The CS 10K program took that challenge head-on and has led to many successful programs all over the United States where universities, local non-profits, government organizations, and local software industry have all worked together to make teacher training possible.

The other challenge has always been curriculum that engages and excites students and that is seen by teachers and universities professors as rigorous enough to prepare them for their first computer science course at university. This challenge was met with enthusiasm by the creation of the introductory *Exploring Computer Science* [35] course as well as the subsequent new AP Computer Science course titled - *AP Computer Science Principles* [4], [11]. While the Exploring CS course provided teachers with detailed lesson plans and hands-on activities regarding what can be done on a daily basis, the AP CS Principles course focused on a framework that provided principles that students should have before they go on to take a computer science course at the university level. These courses have been developed over the last 10 years and the AP CS Principles course was available as a viable option for students for the first time in the 2017-2018 academic year [11]. Based on the results, the number of students taking the AP CS Principles exam has increased significantly as compared to the other AP CS A course that was mostly focused on teaching Java. There has also been a substantial increase in participation by under-represented students [20].

While this combination of teacher training and curriculum development has created an ideal situation for students at their local high schools to take computer science courses, we believe that we need to incorporate inclusive pedagogical strategies into our curriculum as well. While access to these courses is extremely essential, challenges with *self-efficacy* [50] and *self-identity* [64] continue to persist within under-represented groups in any STEM field. Fortunately, there has been significant research that addresses the various issues that under-represented students (women and minorities) experience on a regular basis.

In our program over the last three years, we have trained over 56 in-service teachers who in turn have taught computer science courses to over 1400 students at their respective schools during the 2016-2017 academic year. Based on our past experiences with summer professional development of teachers, we found that teaching computer science for in-service teachers was a daunting task [63]. To address the issues related to self-efficacy of the teachers, we created a two-year Master of Science in STEM Education program with an emphasis on Computer Science Education and a Graduate Certificate - Computer Science Teacher Endorsement program. These programs provide the teachers with the skills necessary to teach courses such as Exploring Computer Science, AP Computer Science Principles, and AP Computer Science A at their respective schools. The curriculum comprised of a combination of technical courses and pedagogical skills to provided comprehensive coverage of the material.

In this paper, we discuss the content and impact of a course in our curriculum that focuses on inclusive pedagogy in the classroom. Through the readings in the course, the course

aims to increase awareness of the challenges that women and minorities may be facing in their class as they take their first/second computer science class. The discussion-based format of the course leads to in-service teachers sharing their stories with similar experiences in their other courses which make the course content more relevant to the participating teachers. After having taught the courses for the last three years, we reflect on the course and present our findings on the same in the paper.

II. Related Work

Training Computer Science teachers is a crucial task in addressing the inequity in computer science education. Short professional development workshops are limited in their ability to prepare and train teachers to teach a year/semester long course at their school. Research by Bernier and Margolis [9] showed that the majority of CS teachers in the past have been trained computer scientists, which makes it hard to hire and retain them. They found that the Los Angeles Unified School District had 'lost' more teachers than were retained. Training teachers to teach computer science through sustained support by a community of practice [40], [52], [58] is a better way to increase their self-identity [57] and efficacy in the classroom.

Our project (IDoCode) was inspired by the National CS10K program led by Jan Cuny [25]. With the vibrant software industry in the Boise Metropolitan area, there was a huge need to train the next generation of computer scientists and that led us to think about the entire pipeline from K-12 all the way through the undergraduate curriculum. To broaden the access to computer science education to women [48] and minorities [47], we created a teacher-focused curriculum that incorporates *pedagogical considerations* as well as computer science education courses such as the Exploring Computer Science [35] and the new AP Computer Science Principles [22].

As per Goode et al. [34] though, merely an excellent curriculum is not sufficient and requires careful attention to ensure that women and minorities are **truly included** in the computer science education at the K-12 level. We believe that through our comprehensive education and *reflection-based approach* to teacher training, we are able to instill a sense of inclusiveness in our teachers to inspire them to provide equal and excellent education in their respective schools. Our comprehensive program instills a sense of identity [56] and self-efficacy [45] into the teachers, which is crucial for teachers with a non-CS background.

DesJardins et al. [28] discuss their efforts in Maryland to provide high quality training for high school teachers through community building, summer workshops, the creation of a local CSTA chapter, and other meetings to increase awareness among educators and administrators at their constituent schools. Other similar efforts include Georgia Computes! [14]. It is a program that was started in 2006 to change the landscape of computer science education through summer camps for K-12 students, teacher training, 1-on-1 mentoring for high school students by pairing them with graduate students, creation of educational material that can be used by teachers at the high school level, and so on.

In the state of Massachusetts, the Commonwealth Alliance for Information Technology Education (CAITE) [1] is focused on increasing opportunities for women and minorities in computing. They have worked with local community colleges to create pathways for students to transition into a 4-year college. They have created new alliances with STEM initiatives to train high school teachers and other constituents at local schools such as principals and counselors. They are sustaining their community of practice [52], [58] through professional development workshops for their teachers. The CAITE effort is led by the University of Massachusetts at Amherst. Similar to our program, they have also been involved in the statewide efforts for defining computing curricula and standards for the Massachusetts K-12.

Illinois State University has spearheaded a program to train high school teachers - *Teacher Education in Computer Science* [41]. This program has some similarities with our program where a teacher can receive Illinois State endorsement through their program for middle-school and high-school teachers.

The course by Carol Fletcher titled "Strategies for Effective and Inclusive CS Teaching" [29] from the University of Texas at Austin has some similarities with our course, but their focus is more on *social justice*. Their course objectives include topics such as "Gain an understanding of equity and social justice as it applies to computer science field from classroom to career", and "Explore their own unconscious biases, beliefs and stereotypes can influence their teaching and student recruiting; program and material selection, and even their subjectivity." They also propose that student will learn to "Develop the skills to *advocate* for computer science using facts, dispelling myths, and sharing the impact on the bottom line when students gain life-long career skills."

Our course is for a focused audience that includes in-service teachers that are about to teach Exploring Computer Science, AP Computer Science Principles, AP CS A or some other computer science related course. Our course is designed to instill a reflective pedagogical approach towards creating a truly inclusive and welcoming culture in the classroom for **all**.

III. Course Content

Our course was titled "Inclusive Strategies for Computer Science Education" and was aimed at in-service teachers who were enrolled in our two-year Master of Science in STEM Education or the Graduate Certificate program. The primary reading for our course was the excellent book "Stuck in the Shallow End: Education, race, and computing" by Margolis et al. [47]. The book contains a comprehensive landscape of the problems and opportunities with respect to teaching computer science at the high-school level. The authors discuss various schools each of which have their set of problems and highlight that an ideal setting is one where the teacher is prepared, the curriculum is exciting, the students are engaged

and the technological resources are available for them to learn effectively.

A. Course Structure

The course was structured in a way where the participants would read the textbook as well as research papers from the computer science education and STEM education literature. In addition to participating in an online forum to discuss the reading before class, the students were also expected to turn in three existing lesson plans at the beginning of the course. These lesson plans were to be modified as the semester progressed and the *modified* lesson plans were submitted by the students at the end of the course. Therefore, in addition to watching the discussions and concepts impact the students, it was also possible to compare the lesson plans that were submitted before the course with those that were submitted at the end of the course.

The main modules of our course are:

- self-identity,
- strategies (pair programming, team-based learning),
- curriculum design,
- growth mindset,
- and teachers as change agents.

B. Self-Identity

In this module, the emphasis was on increasing the teachers' awareness about issues that their students may face in their classroom. We discuss the *ambient belonging* paper by Cheryan et al. [18] that presents the various stereotypical cues that affect the sense of belonging for women and minorities in Computer Science. The researchers found that "objects that are stereo typically associated with computer scientists" such as a Star Trek poster, video games, and so on affected the women's perception of belonging and they were less drawn to that environment. On the other hand, women were drawn to potentially working at a company whose environment contained non-stereotypical objects such as nature posters, plants, and so on.

Self-Identity deals with the ability of an individual to see themselves as a working professional in that field. Underrepresented students frequently struggle with a self-identity issue [64] either due to the impostor syndrome [42] or the lack of role models [10] that they see around them or in the media. The National Center for Women and Information Technology (NCWIT) has distilled various research papers into an article on "Ways to Engage Underrepresented Students in Computing" [55]. Articles like these provided our teachers with tangible strategies that they could use to help underrepresented students succeed in their classes.

We discussed maintaining an open climate in the classroom rather than a "defensive climate" [6] where students constantly feel judged and where there is low tolerance for disagreement. Even though a teacher may want to foster cooperation, a defensive climate can make it more competitive for the students in that class. We discussed how teachers must make an effort to call on all students equally and not only on the students who

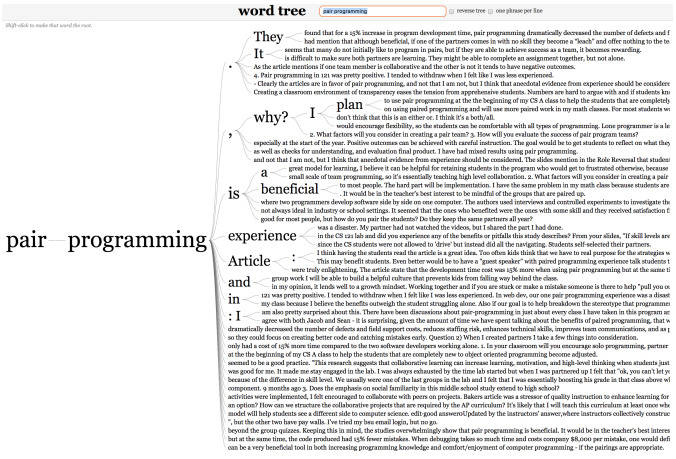

Fig. 1. This word tree representation [68] of the discussions on the online forum show that there was signification conversation related to pair programming and its impact on students with varying levels of exposure to computer science.

they know. In computer science, there are frequently students who have more programming experience than others and they are perceived by their peers as being "better/smarter" than the rest of the class. It is the instructor's responsibility to dispel any such myths and discuss that those students have more experience due to their previous training/exposure to coding and that the rest of the class need not judge themselves because you are not judging them.

C. Strategies - Pair Programming and Team-Based Learning

To increase the awareness of various teaching strategies that promote cooperation, teamwork, and community in the classroom, we introduce the Pair Programming as well as the Team-Based Learning strategies to our students. Both strategies have their strengths and weakness and we discussed when each of them could be used in their respective classrooms.

Pair Programming [19] is a technique that was popular in the late 90's and consists of two students/programmers working together to solve a problem. The key to pair programming is that only one of the programmers can type at a computer (known as the *driver*) at a time whereas the other individual (known as the *navigator*) provides verbal help and feedback to the one typing. This technique was widely used in industry to help with increasing productivity of their employees, but was subsumed by other software engineering paradigms such as Agile development. Seasoned programmers found that, in some cases, they were twice as productive using pair programming [8].

Computer Science Education researchers [26] have studied pair programming for its use in educational settings and have found that it leads to increased confidence and an ability to solve problems quickly. Denner et al. [26] found that teams of middle-school students that are stuck ask for questions more frequently than an individual who may feel uncomfortable asking for help in a classroom. They also found that less experienced students in class gained computational thinking

skills as well as gained knowledge from their partners as they worked together.

Figure 1 shows a *Word Tree* [68] representation of all the discussions on the online forum that were pertaining to "pair programming" and their pros and cons. A Word Tree allows the exploration of large amounts of text by providing a keyword/phrase which in turn is used to filter the data and show only the themes that emerge from that keyword/phrase.

Thomas et al. [66] conducted a study into the composition of the pairs. Should one pair consisting of two strong students work together or should each pair consist of a student with more experience and one with less experience? Based on their study with first-year students in a university course, they found that students actually enjoyed working with students who were approximately at the same level. Students who were more experienced did not actually like pair programming as compared to students who had less experience. The less experienced students gained a lot from the exercise.

We discussed these resources in class and in an online forum where teachers shared their experiences with group work in class and how it needs to be managed carefully. One of them responded with " [...] the studies overwhelmingly show that pair programming is beneficial. It would be in the teachers best interest to be mindful of the groups that are paired up. During introductory courses, I think its better to bring students together that have the same or similar level of coding skill." Another teacher shared " I think that paired programming is an idea that I will try to include in my math classroom as well allowing opportunities for the students to work with multiple different partners over the course of the school year." Almost all teachers expressed excitement about pair-programming and were interested in trying it out in their respective classrooms.

The other pedagogical strategy that we discussed was Team-Based Learning [49]. Team-Based Learning (TBL), as established by Michaelsen, is unique from other forms of team-based pedagogy strategies. Specifically, the components of pre-class preparation for students, individual Readiness Assurance Test (iRAT), team Readiness Assurance Test (tRAT) using IF-AT (Immediate Feedback Assessment Technique) cards, and application exercises are unique to TBL. The formation of teams also follows a specific procedure to distribute talent more equally across the teams. TBL was developed to provide a local community among students in class, especially for large class settings. While the strategies of in-class group work are ideal for large classrooms, they work perfectly well for smaller (30-40 students) class sizes too. The benefits of working in a team and applying your knowledge towards solving a problem provide the students with an increased sense of self-efficacy and a positive experience with respect to team work. The IF-AT cards used by teams lead to more discussion and stronger learning and retention by students. Teachers in our program practice TBL in their teacher preparation courses so they have direct experience with it. Several teachers have adopted TBL in their high school courses as a result of their exposure in our teacher preparation program.

D. Curriculum matters

In today's day and age with students always dependent on their phones for information, communication, entertainment, and so much more, it is crucially important to have a curriculum that resonates with students. Students are excited to learn concepts related to mobile apps, the world-wide web, online privacy and security, artificial intelligence, and so on. These concepts can help them understand the world around them and help them appreciate why it is crucial to learn about the fundamentals of computer science.

With these goals in mind, the Exploring Computer Science course [35] was developed. The curriculum focuses on understanding that problem solving is at the heart of computer science and that programming languages and algorithms are tools that can help us solve problems. It also introduces students to data analysis and showcases robotics and web design as interesting and relevant applications of computer science.

The AP Computer Science Principles course was designed to provide teachers with the flexibility to design a curriculum given a set of *principles* and *big ideas*. These principles focus on creating and analyzing computational artifacts as well as learning to communicate the results of your analysis to others using the appropriate terminology. The big ideas show students that creativity, abstraction, and the ability to use data are at the heart of computer science and that with the ubiquity of the Internet it is critical to be mindful of power and perils that it presents to today's software developers.

Due to the fact that the AP CS Principles course was mostly a framework, there are many *implementations* of the course available to a teacher to choose from. Mobile CS Principles [51] is one such implementation that uses the App Inventor [69] program to create Android apps in their first programming class. Similarly, the Beauty and Joy of Computing [31] implementation provides students with a blocks-based environment to get familiarized with computing in a web-based environment. Code.org also provides a similar implementation that uses Javascript and App Lab. There are many more such implementations for teachers to choose from. Students do have to take an exam at the end of the year which tests general concepts in addition to their computational artifacts that they submit for their final grade.

Our in-service teachers were interested in knowing more about the various options and we discussed the various offerings in class as well as online. Both the Exploring CS (ECS) course as well as the AP CS Principles course have emerged from the CS Education literature and has benefited from years of hard work from the experts in the field. This has made the courses exciting as well as relevant to the students who may consider taking a programming intensive course at their respective universities. In fact, Ryoo et al. [62] provide a success story related to ECS that is based on inquiry, a culturally relevant curriculum, and equity-oriented pedagogy. Their paper debunks the myth that females and students of color are inherently uninterested in rigorous CS learning.

E. Growth Mindset

In the field of Computer Science, there if often talk of the "geek" gene [44] and some teachers look at their bimodal grades (particularly for the first programming class [2]) and agree with their prejudice. This unfairly biases the educators about the ability of the other students who do not have the same level of exposure to computer science as their peers. Robins [61] proposed a Learning Edge Momentum theory that states students who gain a concept easily are more willing to work hard to understand the next concept that in turn they end up mastering. This *momentum* helps students do well in class by mastering the material over time through persistence and *grit*.

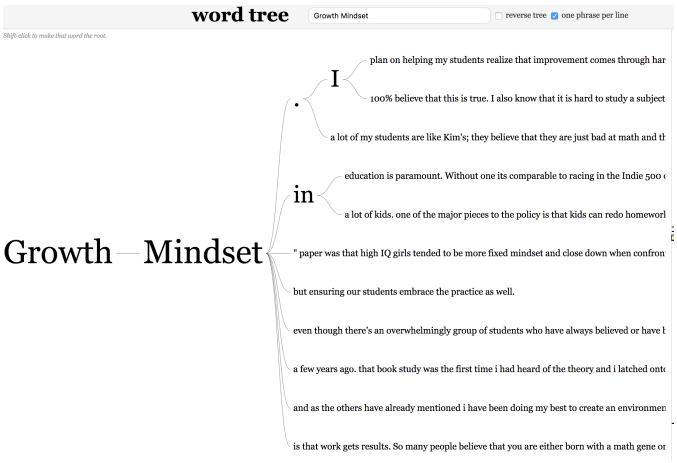

Fig. 2. A Word Tree [68] of all the discussions on the online forum pertaining to the growth mindset shows that the teachers were aware of the growth mindset and were eager to use it in their classroom to remind students that they are not inherently bad at something and that they can get better with practice.

While the teachers in our classes are well aware of the **growth mindset** philosophy and **grit**, we remind them about the fact that the students who have less experience with computer science may be feeling self-doubt due to in-class dynamics and that as teachers they need to provide these students more encouragement. We discuss ways in which teachers can provide feedback to the students using a growth mindset [54]. Figure 2 shows a Word Tree [68] that highlights the discussions surrounding growth mindset as it pertains to computer science and the misconceptions of women and minorities that they are not good at it.

Research by Murphy and Thomas [53] highlights the dangers of a fixed mindset as it relates to computer science education. The researchers found that students with a fixed mindset frequently end up "having a helpless response to challenges that seems to also lead to reduced self-esteem during college". On the other hand, students who have a growth mindset seem to understand that their failures are due to a lack of effort on their part rather than a lack of ability, which is crucial for women and minorities taking computer science courses.

F. Perceptions and Misconceptions

Zimmerman et al. [71] present a success story that highlights the significant benefits of an after-school program on a predominantly Hispanic school. They found that males **and** females were equally interested in pursuing a career in computer science and their motivations were related to earning more money, having a role model, and having exposure to computers and coding at their after school program.

Based on a survey of 836 students by Carter [17], she found most students incorrectly assume a career in computer science involves sitting in front of a computer all day. She recommends new interdisciplinary courses need to be offered that showcase the applications of computer science in a variety of fields to engage a wide range of students.

In the article by Philip Guo [38], he mentions the privilege that came with being an Asian student taking Computer Science courses at MIT. He mentions how his presence at MIT was never questioned even though he had only one year of programming experience compared to his peers. Here is an excerpt from the article - "For instance, whenever I attended technical meetings, **people would assume that I knew what I was doing** (regardless of whether I did or not). "

While biases and prejudices are part of our lives, as educators we need to remember they can affect the way we deal with our students and we need to remind our students that they have certain biases and false expectations from their peers too based on their past experiences.

G. Teachers as Change Agents

As this course was designed to introduce in-service teachers to inclusive pedagogy with respect to computer science, the module on "Teachers as Change Agents" was particularly engaging to the teachers. In addition to the relevant chapter from the course textbook [47], the teachers also read an article from Goode et al. [33] that discusses how teachers can have a significant impact on historically underrepresented students. Their study found that frequently teachers do not have enough resources to teach computer science, but if you provide appropriate training to the teachers they can "increase opportunities for underrepresented students to study computer science. " A student in the class responded to the reading on the classroom discussion forum by saying "The Goode article reinforces all other research that shows the best way to have a successful classroom is to have an effective and qualified teacher."

Interestingly, the paper by Baker [5] that discusses the need for better curriculum and pedagogy to engage women in STEM fields. While the students in the various offerings of the course agreed that the strategies in the paper such as early science experiences/exposure, hands-on activities curriculum that addresses student's interests, activities that build self-efficacy, discussing role models, and student-centered teaching are important, many students expressed that those were just good teaching strategies and were not specific to engaging women in STEM fields. Some of the students in the class were disturbed by seemingly condescending comments in the

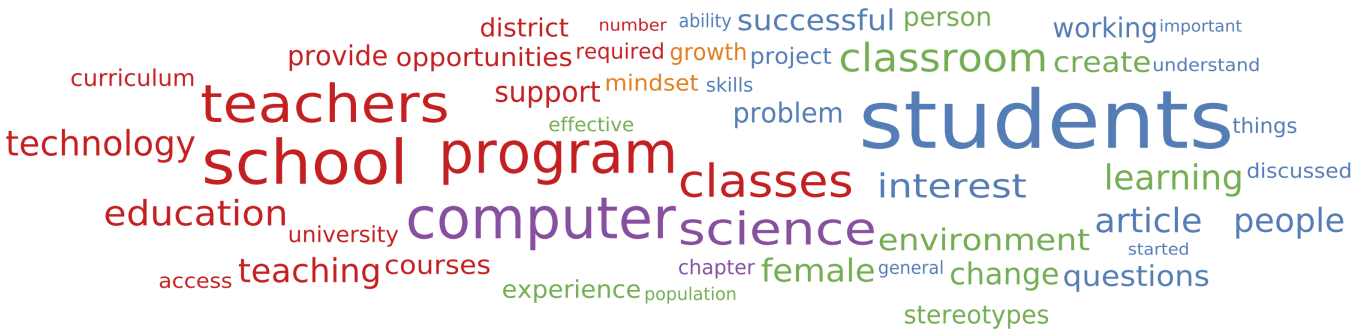

Fig. 3. Word Cloud of all the discussions based on the reading in all the sections of the course. The main themes of teachers, students, computer science come through as being widely discussed. Notably, the text in green shows that the environment in a classroom was discussed as they had read that stereotypical cues can affect self-identity for women and minorities in the paper by Cheryan et al. [18].

paper such as "one way to increase participation in computer programming classes is to have single-sex classes that emphasize programming with *girl-friendly design and drawing activities*." One student further expanded on this theme with "I believe that students rise to the expectations we have for them. A girl who works hard and 'gets it', builds a greater sense of self-confidence."

IV. Reflection

As part of the assigned reading for the course, the students in the class had to submit a reading response to every reading. Some of the comments from the reading responses have been mentioned before. Figure 3 shows a word cloud that was generated based on all the discussions on the online forum that we used for the course. The Word Cloud was generated using a Context-Preserving Layout with a Cosine Coefficient for Similarity and we used the Lexical Centrality measure to rank the words. Similar words were grouped together and stop words were removed to allow for interesting themes to emerge. As can be seen in the figure, themes in red pertain to teachers, technology, and school related topics whereas themes shown in green are related to the environment in the classroom. This may be due to the fact that early in the course students read the ambient belonging paper by Cheryan et al. [18]. It seems to resonate with the students and comes up throughout the course when discussing strategies to include women and minorities in their classes.

A. Survey results

We conducted a survey of the teachers who have taken the course over the last three years and received responses from 11 of the 39 teachers. Not all of the 39 teachers are currently teaching CS or had an opportunity to teach CS classes due to a variety of school/school-district related reasons, which may partially account for the low turnout for the survey.

The survey contained six questions that asked the participants about the strategies that

1) Which of these topics resonated with you from CS 518?
   - Pair Programming
   - Stereotypical cues affecting gender participation
   - Using role models to engage female students
   - Curriculum matters - Using a better curriculum to engage interest in CS
   - The role of teachers in maintaining an inclusive culture in the classroom
2) Which of the following, if any, collaborative learning strategies have you used in your class?
   - Pair Programming
   - Team-based learning
   - Peer-led discussions
3) Which of the strategies for inclusion discussed in the CS 518 class have you incorporated in your classroom?
4) What change, if any, have you brought about in your courses/classroom based on the discussions in CS 518?
5) Are there specific strategies you loved from the class, but haven't been able to implement? If so, what is the strategy and what has been the roadblock to implementing it?
6) Are there any specific readings from the course that stuck with you?

Based on the survey, we found that the teachers most remembered discussions related to "Pair Programming" and "The role of teachers in providing an inclusive environment" and course to all the students in their class. 10 out of 11 teachers reported to using a combination of team-based learning, pair programming, and peer-led discussions in their classrooms. Teachers reported that "Learning and using student's names during class" made a big difference in the classroom and they have "made their classroom environment more inclusive for girls, used TED talks that feature women in STEM, and used pair programming strategies."

Based on the discussions related to ambient belonging due to stereotypical cues [18], teachers reported that they paid more attention to "How my classroom is decorated and focusing the first couple of days [on] developing classroom culture and maintaining it throughout the year." Regarding programming assignments and homework, one teacher reported that "After designing a lesson or project I reflect to make sure it is interesting to both genders."

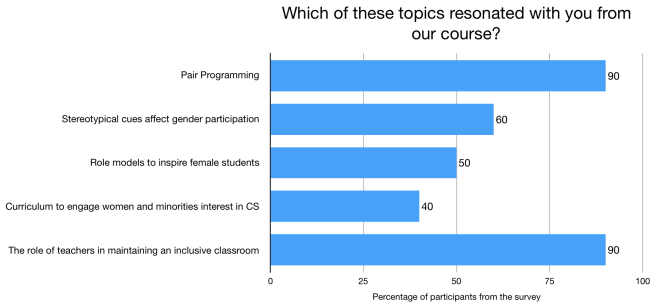

Fig. 4. Here is a bar graph that shows that *Pair Programming* and the *Role of teaching in maintaining an inclusive classroom* most resonated with the participants even two to three years, in some cases, after they took the course.

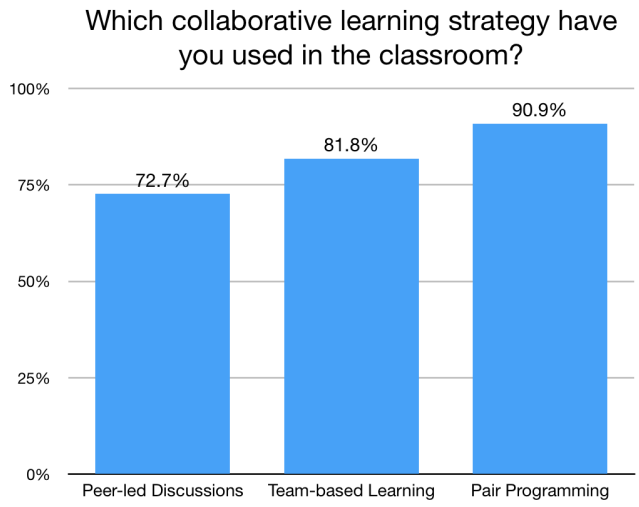

Fig. 5. Among the collaborative learning strategies discussed in class, the teachers reported to have used all of them at some point, but the pair programming strategy seems to be the most popular of them all.

For a question that asked teachers to recall a topic/reading that most resonated with them and has made a *long-lasting impact* on their teaching, they responded with reading about the harmful effects of harboring a "Defensive climate in the classroom." When questioned about which of the various collaborative learning strategies have they incorporated into their respective classrooms, the participants reported that most of them had used all of them at some point. It does seem like the pair-programming paradigm not only resonated with the participants, but was also incorporated into the student learning experiences.

Teachers responded very positively about the book for the course - "Stuck in the shallow end [47]" and one of the teachers responded with "the reading began an inquiry process into the nature of teaching and learning that was universally applicable, but focused on programming as its lens."

Most notably one of the more experienced in-service teachers provided this feedback at the end of the course - "I went into this class hoping I could glean a little insight into methods I could improve underrepresented group retention/recruitment in my program. I have to admit I got a whole lot more than I was planning for. To date, I have restructured, repainted, and remove elements in my classroom to make a more physical friendly environment. I am still in the process of reviewing each lesson I teach to remove what I now can clearly see as a gender bias. I have invited a diverse group of local guest speakers from local companies to talk with my students attempting to bring CS out of the "basement". And I have become so deeply taken by the concept of pair programming that I am attempting to develop a tool that will help find the best match for pairs based off of factors we learned in this class and character traits."

B. Lesson Plans

As mentioned in Section III-A, the students turned in three lesson plans at the beginning of the course. Since the students were in-service teachers, they had lesson plans that they had used before and so that was minimal work on their part. At the end of the course they modified the original lesson plans such that they incorporated strategies that they found appropriate depending on the course, level of the student audience, and subject matter.

In most cases, there was significant improvement with respect to collaboration and working with a partner. This is in line with the "pair programming" paradigm that seems to have resonated well with the students. One of the teachers mentioned that she had started thinking about the composition of the groups/teams as well as specified in the modified lesson plan - "Here I would make sure my groups are purposely chosen to make sure there is cross-cultural and cross-gender communication in the groups, and minimize grouping of the same color/gender/language peers together, feeling isolated from the rest of the class."

To facilitate engagement and creativity so as to appeal to a wider audience, another teacher modified her lesson plan in such a way that she would "Offer students to use their own symbols for their cards, not just shapes - maybe animals, flowers, sport items or cultural symbols." She mentioned that the "Element of creativity, and opportunity to incorporate part of their culture or interest will help students to have a feeling of belonging, and have a feeling of ownership of final product. Some creative students go a long ways to express themselves, and it helps them to connect to concepts that seem foreign."

Overall, there was a wide range of lesson plans but all the teachers had incorporated at least a couple of the strategies discussed in class with the goal to create and sustain a more inclusive culture in their own classroom.

V. Conclusion and Future Work

In this paper, we presented an overview of our course on "Inclusive Strategies for Teaching Computer Science" and reflect upon its impact on the various in-service teachers. Many of these teachers who took the course have taught computer science classes at their school and have found strategies such as "pair programming" to be useful. They have also embraced the need to be mindful of the culture in the classroom and truly

make it inclusive by even changing the physical environment of the classroom including posters, figurines, and so on. Based on the conducted survey, we found that the teachers have incorporated collaborative learning strategies in their teaching of computer science courses.

While our courses incorporated strategies for women and minorities for now, we would like to incorporate principles from the *AccessCSForAll* program for disabled students [15]. We can only create a truly inclusive classroom when we include everyone in it.

Additionally, in future, we plan to conduct interviews with all of our in-service teachers (who are currently teaching a computer science class) and some of their students to see what impact the pedagogical strategies have had towards creating an inclusive classroom.

While the CS 10K program has led to teachers being trained nationally, there is still much that needs to be done to make sure that the trained teachers can teach one of the CS course at their respective schools. Counselors need to be educated to be unbiased and informed [27] so as to not act as gatekeepers to computing classes at their school. Barr and Stephenson [7] discuss all the challenges associated with bringing computer science to various school districts. Specifically for school administrators, they recommend "Provide materials that will help school administrators understand computational thinking so that they can see why this knowledge and skills are important for todays students."

ACKNOWLEDGEMENTS

We would like to thank the National Science Foundation for funding the research under the NSF grant #1339403 - "CS 10K: IDoCode: A Sustainable Model for Computer Science in Idaho High Schools." We would like to thank the Idaho Technology Council, the Idaho State Board of Education, the Idaho Digital Learning Academy, and Local High School Principals, Counselors, and Teachers who continue to support the IDoCode program.

REFERENCES

[1] W. R. Adrion, S. Biskup, D. Boisvert, L. Clarke, J. Fountain, P. Grocer, S. Mackler, A. R. Peterfreund, K. A. Rath, A. Smith, D. D. Snyder, and A. Wiens. Broadening participation in computing: K12-community-college-university-graduate pathways. In *2008 38th Annual Frontiers in Education Conference*, pages S4F–15–S4F–20, Oct 2008.

[2] A. Ahadi and R. Lister. Geek genes, prior knowledge, stumbling points and learning edge momentum: parts of the one elephant? In *Proceedings of the ninth annual international ACM conference on International computing education research*, pages 123–128. ACM, 2013.

[3] O. Astrachan, T. Banres, D. D. Garcia, J. Paul, B. Simon, and L. Snyder. Cs principles: piloting a new course at national scale. In *Proceedings of the 42nd ACM technical symposium on computer science education*, volume 42, pages 397–398. ACM, 2011.

[4] O. Astrachan, J. Cuny, C. Stephenson, and C. Wilson. The cs10k project: mobilizing the community to transform high school computing. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, pages 85–86. ACM, 2011.

[5] D. Baker. What works: Using curriculum and pedagogy to increase girls' interest and participation in science. *Theory Into Practice*, 52(1):14–20, 2013.

[6] L. J. Barker, K. Garvin-Doxas, and M. Jackson. Defensive climate in the computer science classroom. *ACM SIGCSE Bulletin*, 34(1):43–47, 2002.

[7] V. Barr and C. Stephenson. Bringing computational thinking to k-12: what is involved and what is the role of the computer science education community? *Acm Inroads*, 2(1):48–54, 2011.

[8] K. Beck. *Extreme programming explained: embrace change*. Addison-Wesley Professional, 2000.

[9] D. Bernier and J. Margolis. The revolving door: Computer science for all and the challenge of teacher retention. ECS report. http://www.exploringcs.org/wp-content/uploads/2014/04/The-Revolving-Door-CS-for-All-and-the-Challenge-of-Teacher-Retention-Final.pdf, 2014.

[10] J. Black, P. Curzon, C. Myketiak, and P. W. McOwan. A study in engaging female students in computer science using role models. In *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*, pages 63–67. ACM, 2011.

[11] A. C. Board. Ap computer science principles. Online, 2015. Retrieved June 16, 2018 from https://apstudent.collegeboard.org/apcourse/ap-computer-science-principles.

[12] A. Briggs and L. Snyder. Computer science principles and the cs 10k initiative. *ACM Inroads*, 3(2):29–31, 2012.

[13] Q. Brown and A. Briggs. The cs10k initiative: progress in k-12 through exploring computer science part 1. *ACM Inroads*, 6(3):52–53, 2015.

[14] A. Bruckman, M. Biggers, B. Ericson, T. McKlin, J. Dimond, B. DiSalvo, M. Hewner, L. Ni, and S. Yardi. Georgia computes!: Improving the computing education pipeline. In *ACM SIGCSE Bulletin*, volume 41, pages 86–90. ACM, 2009.

[15] S. Burgstahler and R. Ladner. An alliance to increase the participation of individuals with disabilities in computing careers. *ACM SIGACCESS Accessibility and Computing*, (85):3–9, 2006.

[16] CAITE. Commonwealth allication for information technology education. http://caite.cs.umass.edu/index.html, 2008.

[17] L. Carter. Why students with an apparent aptitude for computer science don't choose to major in computer science. *ACM SIGCSE Bulletin*, 38(1):27–31, 2006.

[18] S. Cheryan, V. C. Plaut, P. G. Davies, and C. M. Steele. Ambient belonging: How stereotypical cues impact gender participation in computer science. *Journal of personality and social psychology*, 97(6):1045, 2009.

[19] A. Cockburn and L. Williams. The costs and benefits of pair programming. *Extreme programming examined*, 8:223–247, 2000.

[20] Code.org. Girls set ap computer science recordskyrocketing growth outpaces boys, 2017. Retrieved May 3rd, 2018 from https://medium.com/@codeorg/girls-set-ap-computer-science-record-skyrocketing-growth-outpaces-boys-41b7c01373a5.

[21] code.org. Rethinking perkins to expand access to k-12 computer science. https://code.org/files/RethinkingPerkins.pdf, 2017. Accessed: 2017-02-09.

[22] College Board. AP Computer Science Principles. http://www.apcsprinciples.org/, 2017. Accessed: 2017-4-11.

[23] O. S. Crutchfield, C. D. Harrison, G. Haas, D. D. Garcia, S. M. Humphreys, C. M. Lewis, and P. Khooshabeh. Berkeley foundation for opportunities in information technology: A decade of broadening participation. *ACM Transactions on Computing Education (TOCE)*, 11(3):15, 2011.

[24] J. Cuny. Transforming high school computing: a call to action. *ACM Inroads*, 3(2):32–36, 2012.

[25] J. Cuny. Transforming k-12 computing education: an update and a call to action. *ACM Inroads*, 6(3):54–57, 2015.

[26] J. Denner, L. Werner, S. Campe, and E. Ortiz. Pair programming: Under what conditions is it advantageous for middle school students? *Journal of Research on Technology in Education*, 46(3):277–296, 2014.

[27] P. J. Denning and A. McGettrick. Recentering computer science. *Communications of the ACM*, 48(11):15–19, 2005.

[28] M. desJardins and S. Martin. CE21-Maryland: the state of computer science education in maryland high schools. In *Proceeding of the 44th ACM technical symposium on Computer science education*, pages 711–716. ACM, 2013.

[29] C. Fletcher. Strategies for effective and inclusive cs teaching, 2018. Retrieved May 3rd, 2018 from https://utakeit.stemcenter.utexas.edu/media/syllabi/WeTeach_CS__Strategies_for_Effective_and_Inclusive_CS_Teaching_Syllabus_1.pdf.

[30] J. Freeman, B. Magerko, T. McKlin, M. Reilly, J. Permar, C. Summers, and E. Fruchter. Engaging underrepresented groups in high school introductory computing through computational remixing with earsketch. In *Proceedings of the 45th ACM technical symposium on Computer science education*, pages 85–90. ACM, 2014.

[31] D. Garcia, B. Harvey, and T. Barnes. The beauty and joy of computing. *ACM Inroads*, 6(4):71–79, 2015.

[32] D. D. Garcia, O. Astrachan, B. Brown, J. Gray, C. Lin, B. Beth, R. Morelli, M. desJardins, and N. Sridhar. Computer science principles curricula: On-the-ground; adoptable; adaptable; approaches to teaching. In *Proceedings of the 46th ACM technical symposium on Computer Science Education*, pages 176–177. ACM, 2015.

[33] J. Goode. If you build teachers, will students come? the role of teachers in broadening computer science learning for urban youth. *Journal of Educational Computing Research*, 36(1):65–88, 2007.

[34] J. Goode, G. Chapman, and J. Margolis. Beyond curriculum: the exploring computer science program. *ACM Inroads*, 3(2):47–53, 2012.

[35] J. Goode and J. Margolis. Exploring computer science: a case study of school reform. *ACM Transactions on Computing Education*, 11(2):1–16, 2011.

[36] J. Gray, H. Abelson, D. Wolber, and M. Friend. Teaching cs principles with app inventor. In *Proceedings of the 50th Annual Southeast Regional Conference*, pages 405–406. ACM, 2012.

[37] K. Gunion, T. Milford, and U. Stege. Curing recursion aversion. In *ACM SIGCSE Bulletin*, volume 41, pages 124–128. ACM, 2009.

[38] P. Guo. Silent technical privilege. *Slate. Retrieved on February*, 21:2014, 2014.

[39] M. Guzdial. Get cs into schools through math and science classes: What we might lose. Computing Education Blog - https://computinged.wordpress.com/2014/08/11/get-cs-into-schools-through-math-and-science-classes-and-what-we-might-lose/, 2014.

[40] M. Guzdial. Learner-centered design of computing education: Research on computing for everyone. *Synthesis Lectures on Human-Centered Informatics*, 8(6):1–165, 2015.

[41] Illinois State University. Teacher education in computer science (tecs). http://tecs.illinoisstate.edu, 2016. Accessed: 2017-5-21.

[42] J. Kolligian Jr and R. J. Sternberg. Perceived fraudulence in young adults: Is there an'imposter syndrome'? *Journal of personality assessment*, 56(2):308–326, 1991.

[43] M. Leake and C. M. Lewis. Recommendations for designing cs resource sharing sites for all teachers. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, pages 357–362. ACM, 2017.

[44] R. Lister. Computing education research geek genes and bimodal grades. *ACM Inroads*, 1(3):16–17, 2010.

[45] A. L. Luehmann. Identity development as a lens to science teacher preparation. *Science Education*, 91(5):822–839, 2007.

[46] M. Marder. Uteach: Teacher preparation at the university of texas at austin. *Bulletin of the American Physical Society*, 2005.

[47] J. Margolis, R. Estrella, J. Goode, J. J. Holme, and K. Nao. *Stuck in the shallow end: Education, race, and computing*. MIT Press, 2010.

[48] J. Margolis and A. Fisher. *Unlocking the clubhouse: Women in computing*. MIT press, 2003.

[49] L. K. Michaelsen and M. Sweet. Team-based learning. *New directions for teaching and learning*, 2011(128):41–51, 2011.

[50] I. T. Miura. The relationship of computer self-efficacy expectations to computer interest and course enrollment in college. *Sex Roles*, 16(5-6):303–311, 1987.

[51] R. Morelli, T. De Lanerolle, P. Lake, N. Limardo, E. Tamotsu, and C. Uche. Can android app inventor bring computational thinking to k-12. In *Proc. 42nd ACM technical symposium on Computer science education (SIGCSE'11)*, pages 1–6, 2011.

[52] B. B. Morrison, L. Ni, and M. Guzdial. Adapting the disciplinary commons model for high school teachers: improving recruitment, creating community. In *Proceedings of the ninth annual international conference on International computing education research*, pages 47–54. ACM, 2012.

[53] L. Murphy and L. Thomas. Dangers of a fixed mindset: implications of self-theories research for computer science education. In *ACM SIGCSE Bulletin*, volume 40, pages 271–275. ACM, 2008.

[54] NCWIT. 8 ways to give students more effective feedback using a growth mindset. Online, 2010. Retrieved April 22, 2018 from https://www.ncwit.org/resources/ncwit-tips-8-ways-give-students-more-effective-feedback-using-growth-mindset/.

[55] NCWIT. Top 10 ways to engage underrepresented students in computing. Online, 2010. Retrieved April 22, 2018 from https://www.ncwit.org/resources/top-10-ways-engage-underrepresented-students-computing/.

[56] L. Ni. Building professional identity as computer science teachers: supporting secondary computer science teachers through reflection and community building. In *ICER*, pages 143–144, 2011.

[57] L. Ni and M. Guzdial. Prepare and support computer science (cs) teachers: Understanding cs teachers professional identity. In *American Educational Research Association (AERA) Annual Meeting*, 2011.

[58] L. Ni, M. Guzdial, A. E. Tew, B. Morrison, and R. Galanos. Building a community to support hs cs teachers: the disciplinary commons for computing educators. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, pages 553–558. ACM, 2011.

[59] M. Papastergiou. Are computer science and information technology still masculine fields? high school students perceptions and career choices. *Computers & education*, 51(2):594–608, 2008.

[60] C. Reas and B. Fry. *Processing: a programming handbook for visual designers and artists*. Number 6812. Mit Press, 2007.

[61] A. Robins. Learning edge momentum: A new account of outcomes in cs1. *Computer Science Education*, 20(1):37–71, 2010.

[62] J. J. Ryoo, J. Margolis, C. H. Lee, C. D. Sandoval, and J. Goode. Democratizing computer science knowledge: Transforming the face of computer science through public high school education. *Learning, Media and Technology*, 38(2):161–181, 2013.

[63] E. M. Skaalvik and S. Skaalvik. Dimensions of teacher self-efficacy and relations with strain factors, perceived collective teacher efficacy, and teacher burnout. *Journal of educational psychology*, 99(3):611, 2007.

[64] C. M. Steele. A threat in the air: How stereotypes shape intellectual identity and performance. *American psychologist*, 52(6):613, 1997.

[65] R. Taub, M. Ben-Ari, and M. Armoni. The effect of cs unplugged on middle-school students' views of cs. *ACM SIGCSE Bulletin*, 41(3):99–103, 2009.

[66] L. Thomas, M. Ratcliffe, and A. Robertson. Code warriors and code-a-phobes: a study in attitude and pair programming. In *ACM SIGCSE Bulletin*, volume 35, pages 363–367. ACM, 2003.

[67] US Congress. H.r.1020 - stem education act of 2015. https://www.congress.gov/bill/114th-congress/house-bill/1020, 2015. Accessed: 2017-02-09.

[68] M. Wattenberg and F. B. Viégas. The word tree, an interactive visual concordance. *IEEE transactions on visualization and computer graphics*, 14(6), 2008.

[69] D. Wolber. App inventor and real-world motivation. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, pages 601–606. ACM, 2011.

[70] A. Yadav, S. Gretter, J. Good, and T. McLean. Computational thinking in teacher education. In *Emerging Research, Practice, and Policy on Computational Thinking*, pages 205–220. Springer, 2017.

[71] T. G. Zimmerman, D. Johnson, C. Wambsgans, and A. Fuentes. Why latino high school students select computer science as a major: Analysis of a success story. *ACM Transactions on Computing Education (TOCE)*, 11(2):10, 2011.