Computational Thinking in Elementary and Secondary Teacher Education

AMAN YADAV, CHRIS MAYFIELD, NINGER ZHOU, SUSANNE HAMBRUSCH, and JOHN T. KORB, Purdue University

Computational thinking (CT) is broadly defined as the mental activity for abstracting problems and formulating solutions that can be automated. In an increasingly information-based society, CT is becoming an essential skill for everyone. To ensure that students develop this ability at the K-12 level, it is important to provide teachers with an adequate knowledge about CT and how to incorporate it into their teaching. This article describes a study on designing and introducing computational thinking modules and assessing their impact on preservice teachers' understanding of CT concepts, as well as their attitude towards computing. Results demonstrate that introducing computational thinking into education courses can effectively influence preservice teachers' understanding of CT concepts.

Categories and Subject Descriptors: K.3.2 [**Computers and Education**]: Computer and Information Science Education—*Curriculum*

General Terms: Experimentation, Measurement

Additional Key Words and Phrases: Computational thinking, teacher education, teaching/learning strategies

ACM Reference Format:
Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., and Korb, J. T. 2014. Computational thinking in elementary and secondary teacher education. *ACM Trans. Comput. Educ.* 14, 1, Article 5 (March 2014), 16 pages.
DOI:http://dx.doi.org/10.1145/2576872

1. INTRODUCTION

Computing enables and drives many technologies that are integral in today's society. In fact, its principles influence every aspect of our lives, from shopping with loyalty cards to conducting scientific research. Barr and Stephenson [2011] argued that today's students would live and work in a world that is heavily influenced by computing principles. In the midst of this trend, *computational thinking* (CT) has quickly become a prerequisite skill for many endeavors of the 21st century [Wing 2008]. Computational thinking has a long history within computer science, dating back to the 1950s and 1960s, when it was labeled as "algorithmic thinking" [Denning 2009]. Cuny and others have defined CT as "the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent" [Cuny et al. 2010]. These agents can be computers or humans, or a combination of both. The prominent features of computational thinking revolve around abstraction and automation, indicating the ability

This work was supported by the NSF CPATH program under grant CNS-0938999 and through a gift from State Farm.
Authors' addresses: A. Yadav, Department of Educational Studies, Purdue University; C. Mayfield, (current address) Department of Computer Science, James Madison University; N. Zhou, Department of Educational Studies, Purdue University; S. Hambrusch, Department of Computer Science, Purdue University; J. T. Korb, Department of Computer Science, Purdue University.

to dissect problems, abstract the high-level rules, and use technology to automate the problem-solving process. While abstraction implies the process of selecting information worthy of attention, automation is the use of tools or technology to amplify the power of abstraction [Wing 2008]. In summary, computational thinking involves "solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science" [Wing 2006, p. 33].

Computational thinking has the potential for application in a wide range of disciplines outside of computer and information sciences. Bundy [2007] suggested that computational-thinking concepts have been used in other disciplines via problem-solving techniques, and that the ability to think computationally is essential to every discipline. For example, core computational-thinking concepts could be embedded in social studies by identifying trends in population data (analysis) and deducing general principles from facts (abstraction) [Barr and Stephenson 2011]. Similarly, computational thinking could be applied to language arts by having students do linguistic analysis of sentences and identifying and representing patterns for different sentence structures. Computational thinking also has the potential to foster creativity in the classroom by allowing students to move from consumers of technology to building tools that benefit society [Mishra et al. 2013]. Such pervasiveness of computational-thinking concepts dictates the importance of exposing students to such notions early in their school years and helping them to become conscious about when and how to apply these ideas.

Wing advocated the necessity of computational thinking in K-12 education by stating, "To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability" [Wing 2006, p. 33]. A recent report on computational thinking by the National Council for Research advanced a similar idea: CT is a cognitive skill that an average person is expected to possess [NRC 2010]. The NRC report highlighted "(1) that students can learn thinking strategies such as computational thinking as they study a discipline, (2) that teachers and curricula can model these strategies for students, and (3) that appropriate guidance can enable students to learn to use these strategies independently" (p. 62). However, in order to establish a conceptual understanding of CT and the ability to transfer CT skills across different settings requires prolonged exposure to this mindset. If our goal is to nurture a generation with CT skills, we need to familiarize students with CT in K-12 education early on [Barr and Stephenson 2011].

1.1. Computational Thinking in K-12 Education

Computational thinking has the potential to advance students' problem-solving skills and abilities significantly as they begin to think in new ways. Lu and Fletcher [2009] argued that students need to learn computational thinking early and often, with an emphasis on understanding "computational processes, and not on their manifestations in particular programming languages" and "skills for abstracting and representing information" (p. 24). In order to achieve this goal, they proposed using "computational-thinking language" to incorporate computing concepts in core content areas. CT language is "not a programming language, but rather vocabularies and symbols that can be used to annotate and describe computation, abstraction, and information, and provide notation around which semantic understanding of computational processes can be hung" (p. 25). Similarly, Hemmendinger [2010] argued that the goal of teaching computational thinking "is to teach them how to think like an economist, a physicist, an artist, and to understand how to use computation to solve their problems, to create, and to discover new questions that can fruitfully be explored" and not for everyone to think like a computer scientist.

There are positive signs that students have considerable ability to comprehend computing concepts, which can be used to teach computational thinking in other disciplines [Lewandowski et al. 2007]. Lewandowski and colleagues illustrated that students from diverse backgrounds bring analytical and problem-solving skills to solve a concurrency task in a beginning CS1[1] class. The authors found that 69% of the solutions obtained were correct, which indicated that the nonprogramming students were equipped with the natural understanding of basic concepts to solve computing problems. In another study, Hambrusch and colleagues [2009] found that introducing computational thinking into the undergraduate science curriculum significantly improves students' attitude and interest in taking future computing courses, as well as an increased interest in pursuing a career that requires computing skills. Other studies have suggested that it is possible to increase student interest in computing as well educate them to use problem-solving in other disciplines [Barr and Stephenson 2011]. However, these studies have been conducted at the postsecondary level, and there are few studies on embedding CT in the core content areas of literacy, mathematics, science, and social studies at the K-12 level.

The Computer Science Teachers Association (CSTA) has emphasized the role of CT in K-12 classrooms as "a problem-solving methodology that can be automated and transferred and applied across subjects" [Barr and Stephenson 2011]. As discussed previously, Lu and Fletcher [2009] argued that using computational-thinking language in K-12 pedagogy would allow students to understand computational processes as well as develop skills for abstracting and representing information. The authors present several examples from mathematics, science, and social sciences to show how computational thinking can be integrated in primary and secondary curricula. Similarly, Qualls and Sherrell [2010] suggested introducing CT early in elementary school and nurturing the skill throughout K-12 education. Other researchers have also underscored the importance of computational thinking across disciplines at the K-16 level [Allan et al. 2010; Garcia et al. 2010; Henderson et al. 2007; Wing 2008]. Hence, in order to maximize the aforementioned benefits of computational thinking and get students interested in computing, we need to integrate CT in core content areas at the K-12 level.

The first step in this direction might be to prepare K-12 teachers to present CT ideas in explicit ways. A report by the National Research Council [2010] suggested that students could learn about computational thinking by observing teachers as they model related thinking strategies and guide students to use these strategies independently. However, most of the current efforts to educate teachers about CT have been limited to computer science teachers. For example, Blum and Cortina [2007] introduced computational thinking during a workshop for high school computer science teachers. The purpose of the workshop was to raise teachers' awareness and provide materials that they could use to show students that computer science was more than just programming. Results from the workshop demonstrated teachers' improved understanding of CT and increased awareness of the importance of CT in all aspects of life. In a similar study, Morreale and Joiner [2011] found that the introduction of computational concepts to high school computer science teachers changed their perceptions of computer science as a learning tool which can be used to solve complex problems. These findings suggest that introducing teachers to computational thinking can change their attitudes towards computing as well as raise their understanding of CT as an approach to solving problems [Barr and Stephenson 2011]. These efforts, however, need to involve content-area teachers and not just computer science teachers. In order for

[1]CS1 designates the first course in the introductory sequence of a computer science major, and typically focuses on problem-solving techniques and computer programming.

computational thinking to permeate through other content areas at the K-12 level, it is important to provide all teachers with an adequate knowledge of computational thinking and how to incorporate computational thinking in their disciplines.

One way to approach this task is to begin at the preservice teacher level and introduce computational-thinking concepts in teacher preparation programs. Exposing preservice teachers to computational-thinking concepts early in their teacher preparation might allow them to see the relevance of CT in their own disciplines. Our goal in this study was to examine the impact of computational-thinking modules on preservice teachers' understanding of computational thinking and attitudes towards computing. Our research questions were as follows.

(1) What is the influence of computational-thinking modules on preservice teachers' understanding of computational thinking?
(2) What is the influence of computational-thinking modules on preservice teachers' attitudes towards computing?

2. METHOD

2.1. Setting

This article reports the results of an experiment that assessed the impact of computational-thinking modules on preservice teachers in a required educational psychology course at a large mid-western university. The course introduces future K-12 teachers to basic concepts of classroom management, learning, motivation, and assessment. The main goals of the course include understanding current theories of learning and motivation, and the role of assessment in fostering learning. Since computational thinking is a fundamental skill that includes problem solving and understanding human behavior, it fit well with the existing course topics. Computational thinking was incorporated as one of the existing topics on problem solving and critical thinking. We used computational-thinking modules to demonstrate probabilistic reasoning, algorithmic thinking, heuristics, hypothesis testing, and problem solving.

2.2. Participants

Three hundred and fifty-seven preservice teachers participated in this study. The control group had a total of 200 students enrolled in the introductory educational psychology course during Fall 2011. Of those students, 154 completed the questionnaire, resulting in a 77% response rate. Most of the participants in the control group were either sophomores (44.8%) or juniors (35.7%). The treatment group had 157 students enrolled in the introductory educational psychology course during Spring 2012. Of those students, 141 completed the questionnaire, which resulted in a 92% response rate. Most of the participants were either sophomores (44.7%) or juniors (36.2%). The average age of control and experimental group was 20.5 years and 21 years, respectively. The GPA of control and experimental groups was also similar. Table I presents the demographic information.

2.3. Materials

We developed and implemented a one-week module (two 50-minute classes) on computational thinking in the course. Faculty and graduate students from both education and computer science met on a regular basis to develop the module. The module was taught by the first author and provided students with an overview of computational thinking and engaged them in activities that showcased computational-thinking principles. The module and supplementary materials are available online at `http://cs4edu.cs.purdue.edu/comp_think`.

Table I. Demographic Information

	Control Group	Treatment Group
Response Rate	77% (153)	92% (141)
Year in school [a]	44.8% Sophomore 35.7% Junior	44.7% Sophomore 36.2% Junior
Average GPA	3.29	3.29
Gender	71.9% Female 28.1% Male	70.9% Female 29.1% Male

[a] Two most common responses

The first class introduced students to a definition of computational thinking and five CT concepts: Problem identification and decomposition, abstraction, logical thinking, algorithms, and debugging. Given that preservice teachers had no prior computer science background, we illustrated these concepts with concrete examples from day-to-day life and related the terminologies to preservice teachers' personal experiences. The class began with the instructor asking a pair of students to develop driving directions from point A to point B. The instructor then led the students in a discussion of how they came up with the directions (knew them, sketched a map, picked the best route, asked a friend, etc.) and how extensive their directions were (number of steps, detailed turn-by-turn) using clicker questions. The discussion illustrated the various concepts of computational thinking, including algorithms (step-by-step route), efficiency (best of way to navigate), abstraction (ability to give efficient directions), and automation (using GPS).

We utilized clickers to engage preservice teachers and apply the CT concepts presented to problems presented in the class. For example, debugging was discussed by asking students to problem-solve a scenario of a desk lamp that no longer works. Students used clickers to select one of five possible steps to troubleshoot and "fix bugs" to make the lamp work again. Similarly, the concepts of "binary search" and "parallel processing" were demonstrated through the examples of looking up information from a long list and standing in lines for movie tickets, respectively. The goal of these scenarios was to help students find concepts that were personally meaningful and helped them discover the ubiquitous nature of CT in everyday life. We illustrated a number of fundamental computer science data organization concepts that also arise in day-to-day situations, including Last In–First Out (LIFO) vs. First In–First Out (FIFO), stacks vs. queues, and graphs vs. trees. Overall, the first class focused on the idea that computational thinking is ubiquitous and how computing principles have permeated to transform our lives.

The second class illustrated the application of computational thinking in educational settings, and the benefits of incorporating CT concepts in K-12 classroom. We showcased applications of computational thinking in teaching and learning about various content areas to encourage creativity and problem solving. Specifically, we discussed how computational thinking could allow students to use high-level processes, such as abstraction and reasoning skills to see patterns in data. We highlighted how computational thinking was a useful tool in dealing with ill-defined problems, where there might not be a clear-cut solution and information needed to solve the problem may be missing. The lecture also presented information on how to teach computational thinking and integrate CT concepts into everyday instruction. We also demonstrated how CT could be implemented in the core content areas, such as science and social science through various examples and activities. These examples included role-playing (e.g., acting out the interaction among variables) and simulation (e.g., demonstrating software that simulated problem-solving scenarios in science or social sciences). Finally, the lecture provided preservice teachers with resources and websites that would be helpful incorporating CT ideas into their future classrooms. We pilot-tested the CT modules and measures before implementing them in the current study (see Yadav et al. [2011] for details about the pilot study).

2.4. Measures

2.4.1. Computational-Thinking Quiz. We used three open-ended questions to assess students' understanding of computational thinking, which were developed and refined based upon our previous work on assessing preservice teachers' understanding of computational thinking [Yadav et al. 2011]. Specifically, the open-ended questions were as follows.

(1) How would you explain the concept of "computational thinking?"
(2) Can computational thinking be integrated into the classroom? If yes, how would you implement computational thinking into the content area you plan to teach? If no, why can't computational thinking be integrated into the classroom?
(3) Does computational thinking relate to other disciplines besides computer science? Please provide some examples of how computational thinking has influenced disciplines besides computer science.

2.4.2. Computing Attitude Questionnaire. In order to examine participants' attitudes towards computing, we designed a survey of 21 Likert-type scale questions. These questions were adapted from a questionnaire that was used to assess science and engineering students' attitudes towards computer science [Hoegh and Moskal 2009] and our pilot-study. Specifically, based on the open-ended responses from our pilot study, we modified existing survey items and also added additional Likert-type items to assess student attitudes towards computing [Yadav et al. 2011]. The survey items were organized thematically into five categories: Definition, Comfort, Interest, Use in classroom, and Career/future use. The "definition of the computational-thinking" category included four items that provided respondents with definitions of computational thinking. The "comfort-level" category included six statements that were oriented towards the respondents' comfort level with computational thinking and computer science. The "interest-level" category included four items that examined respondents' interest in computational thinking and computer science. The survey also included two statements focused on the respondents' view of using computational thinking in their future classrooms (use in classroom). The fifth category included five items oriented towards the respondents' view of how computational thinking will influence their future career (career/future use). Respondents were asked to rate their attitudes on the following scale: Strongly Agree, Agree, Disagree, and Strongly Disagree. Internal reliability of the survey was calculated using Cronbach's alpha ($\alpha = 0.76$), which was deemed sufficient.

2.5. Procedure

The present study builds upon and addresses methodological issues from a pilot study [Yadav et al. 2011]. Specifically, the pilot study utilized pre-post test design without a control group, which limited our ability to attribute change between the pretest and posttest to the treatment, as there is always a possibility that some extraneous variable accounted for all or part of the change [Ary et al. 2009]. Another potential issue is pretest sensitization, which suggests that the "pretest may sensitize participants so that they think about the questions and issues raised and subsequently give different response on the posttest" [Ary et al. 2009, p. 294]. Hence, we utilized a quasi-experimental design to evaluate the impact of the modules on preservice teachers' understanding of computational thinking and their attitude towards computing.

Specifically, one introductory educational psychology course served as the control group and another introductory educational psychology course served as the treatment group. The control group received the content typical for the course, which included lectures on higher-level cognitive processes, such as problem solving, transfer, critical thinking, and creativity. The treatment group, on the other hand, received the computational-thinking module. Both groups completed the same quiz and computing attitude survey during the class one week later.

2.6. Data Analysis

Participants' open-ended responses were coded into three overarching categories: Participants' view of computational thinking, Integrating computational thinking into the classroom, and relationship to other disciplines. Table II provides a description and examples of the three overarching themes. Another rater coded one-third, randomly selected, open-ended responses to establish inter-rater reliability, which yielded inter-rater agreement of 84.70%. When there was a difference in coding, the original code was used to be consistent with the remainder of the data.

The five categories from the Likert-scale items were analyzed using two-way univariate analysis of variance (ANOVA) with two factors—condition (control vs. treatment) and gender (males vs. females)—to examine the differences between control and treatment conditions as well as identify any differences based on gender. Specifically, the participants' responses on survey items within each category were aggregated and then averaged based upon the number of items in each category. (Note: reverse coding was used when needed.) We used the average score in the ANOVA.

3. RESULTS

3.1. Understanding Computational Thinking

3.1.1. Participants' View of Computational Thinking. Participants were asked to describe computational thinking and the concepts related to it. The results suggested that the treatment group could differentiate computational thinking from simply "the use of technology or computers." Specifically, treatment group participants were able to form an understanding that computational thinking was a cognitive tool that involved using computing concepts to solve complex problems with or without the use of computers [NRC 2010]. This view is highlighted by one participant's response, "Computational thinking is using basic ideas to solve problems in a logical and systematic way—sometimes it involves a computer, but it doesn't have to." Another participant responded, "Using algorithms and heuristics and a variety of other methods to think logically about a problem and the most appropriate way to solve that problem." Other participants in the treatment also responded similarly, and emphasized the fundamental elements of computational thinking, such as debugging, algorithmic thinking, problem solving, abstraction, and so on.

In contrast, the control group tended to include "the use of computers" as a necessary component in computational thinking, but in a very limited way. Specifically, control group participants reported that computational thinking involved the use of computers, but thought of computers as machines that accomplish specific tasks, not as partners and collaborators that help solve complex problems [NRC 2010]. For example, one participant responded that computational thinking was "either the use of computers or just a robotic way of computing or completing something." Other participant responses highlight similar ideas, such as "computers to do research and assignments effectively" and "[using] computers to help you solve a problem you otherwise couldn't do on your own."

Table II. Examples from Three Overarching Categories

Category	Sub-themes	Example
View of computational thinking	Problem-solving/Logic	CT involves using specific skills and strategies to solve problems in the most logical and effective way. CT involves solving problems and understanding behavior by using concepts fundamental to computer science.
	Algorithms	Computational thinking involves using algorithms and processes to solve problems.
	Use of technology/computers	CT is the process of using technology such as computers or calculators to make solving problems easier. It is using computer to do research and complete assignments efficiently and effectively.
	Critical Thinking	Using critical-thinking and problem-solving skills/strategies to solve a problem in the best way.
Integrating computational thinking into the classroom	Teach students to use problem-solving and logic	I would teach my students how to approach questions and concepts in a way that allows them to search for the best answer by using problem-solving skills.
	Teach students to use algorithms	Teach students to use algorithms and heuristics and a variety of other methods to think logically about a problem and the most appropriate way to solve that problem.
	Focus on critical thinking	Ask questions that the students have to think in depth about the subject. Ask students to explain why they choose what they do.
	Use of technology/computers	I could plan an online PowerPoint lesson that would include a quiz at the end to ensure understanding from the student.
Relationship to other disciplines	Math	I would implement CT into as much subjects as possible. Mostly in math subjects, because there is a lot of problem solving with math, such as algorithms and heuristics. Also have them use trial and error throughout problem solving.
	Everyday/Real world	You can use CT when finding the quickest route somewhere or how to plan efficient grocery store trips among other things.
	Science/Engineering	Using scientific method and testing hypotheses in lab experiments.
	English	It is a part of every subject area. English for example in sentence structure there is a systematic way of doing that, or making poems or dissecting Shakespeare's plays.

A frequency analysis of participants' responses exhibited that the majority (78%) of the treatment group explained the concept as "heuristics + problem solving," whereas only 47% of the control group conceptualized it in this same way. Similarly, more treatment participants' (46%) responded that computational thinking includes algorithms as compared to the control group (31%). Finally, more control group participants (40%)

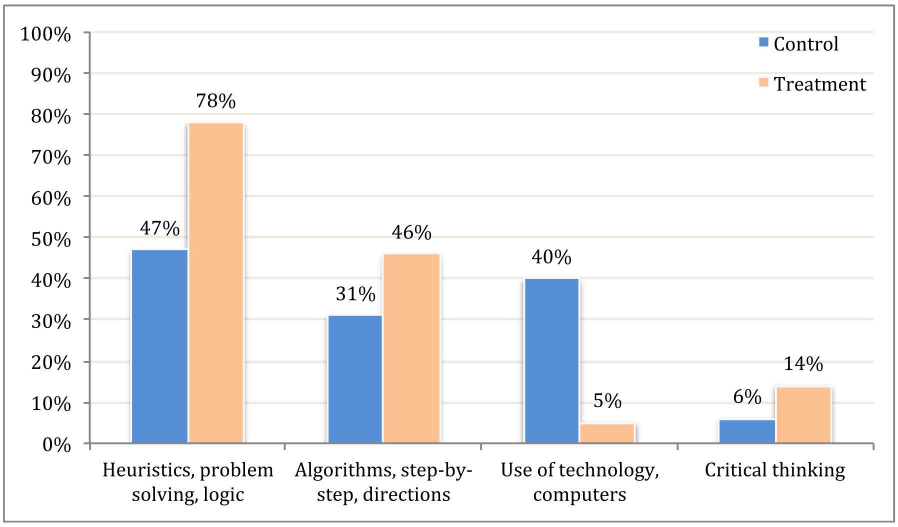

Fig. 1. Participants' definition of computational thinking.
(Note: total exceeds 100% as respondents described multiple categories).

reported computational thinking as merely including the use of technology and computers than the treatment group (5%). Figure 1 shows these results graphically.

3.1.2. Classroom Integration. Respondents were asked if computational thinking could be integrated into the classroom. More participants in the treatment condition (99%) than the control condition (93%) agreed that computational thinking could be implemented in the classroom. Respondents were additionally asked to explain how they would implement computational thinking into the content area they plan to teach. There was a great degree of difference between the results of the control and treatment groups, with the control focusing more on the use of technology, while the treatment group focusing more on teaching their students critical thinking skills and how to use algorithms and heuristics. Specifically, control group participants' ideas regarding computational thinking involved using computers and were limited to mathematics. The following responses highlight control group thinking on how to integrate CT into the classroom: "computer-based learning with solving problems"; "In my art classroom I would use computational thinking to solve, and think through processes with Adobe programs"; "[CT] is good to think about math systematically, however not as much for language arts and English. Providing students with steps to solve problems will help students understand math."

On the other hand, participants in the treatment group were better able to articulate their conceptions about integrating CT in the K-12 classroom without the use of computers and/or technology; instead involving students in problem solving, algorithmic thinking, and abstracting general principles to other situations. For example, one participant responded, "Using heuristics to comprehend plot, setting, characters in a novel, i.e. – concept map, charts." Another participant stated, "you could implement CT in the classroom, by explaining to students various ways that they can link new concepts to old ones by abstracting general rules." Frequency analysis of participants' responses showed that the majority of the control group participants (63%) thought integrating computational thinking in the classroom involved using computers/ technology, as opposed to only 11% of the treatment group participants. The treatment group was also more likely than the control group to think that implementing

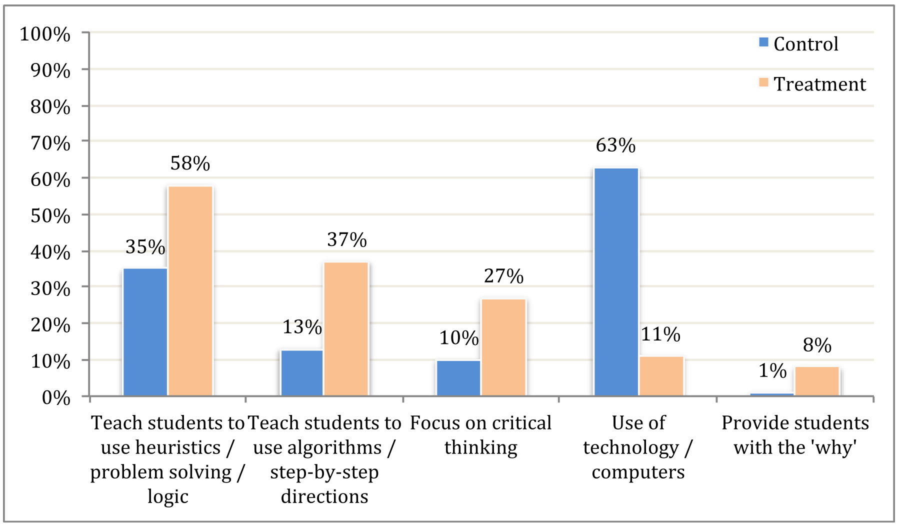

Fig. 2. Participants' view of integrating CT in the classroom.
(Note: total exceeds 100% as respondents described multiple categories).

CT in the classroom involves teaching students to problem solve (58% vs. 35%) and algorithms (37% vs. 13%). Figure 2 shows the distribution of students' ideas about integrating computational thinking in the classroom.

3.1.3. Relationship of Computational Thinking to Other Disciplines. Respondents from both groups were asked if computational thinking relates to disciplines other than computer science. Significantly more participants in the treatment condition (94%) than the control condition (75%) agreed that computational thinking relates to other disciplines. A chi-square analysis was run to see if the treatment group was more likely to answer "yes" that computational thinking does relate to other disciplines besides computer science. Results indicated the treatment group was statistically more likely to have answered "yes" as compared to the control group, $\chi^2(1 = 270) = 10.159, p = .001$. Figure 3 exhibits disciplines that participants reported were related to computational thinking.

3.2. Computing Attitude Survey

Survey results suggested that participants in the treatment group had a better understanding of computational thinking overall.

ANOVA results exhibited that the condition was a significant factor for accurately defining computational thinking [$F(1, 276) = 70.94, p < 0.00, 1 - \beta = 1.00$]. Specifically, the treatment group was more likely than the control group to agree that computational thinking involved logically solving problems and abstracting general principles that apply to other situations. The treatment group was also less likely to report that CT was simply the use of computers and how computers worked.

Condition was also a significant factor for predicting whether computational thinking could be incorporated into the classroom, [$F(1, 281) = 38.87, p < 0.00, 1 - \beta = 1.00$]. Specifically, students in the treatment condition were more likely to agree that computational thinking could be implemented in the classroom by allowing students to problem solve (and not just by using computers). Condition was, however, not a significant factor for comfort with computing [$F(1, 279) = 2.32, p > 0.05, 1 - \beta = 0.33$],

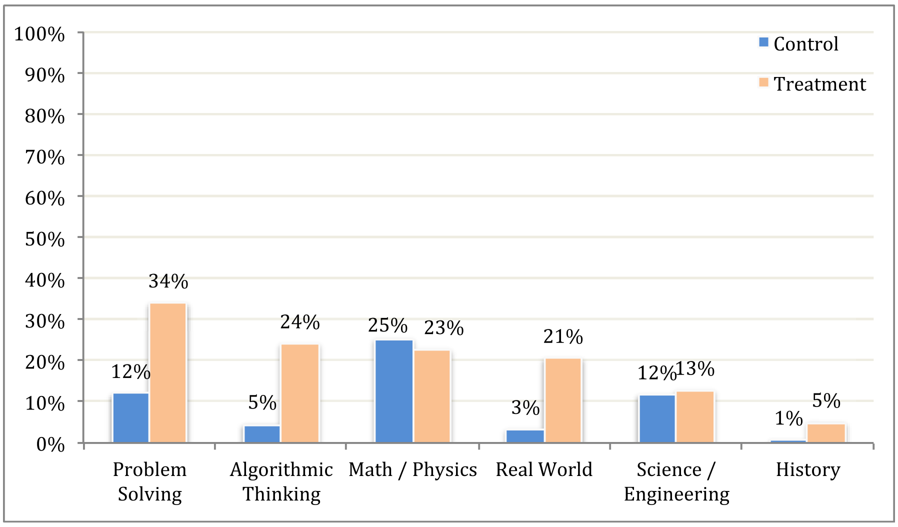

Fig. 3. Participants' view of CT's relationship to other disciplines.
(Note: total exceeds 100% as respondents described multiple categories).

Table III. Descriptive for the Survey Categories

	Definition Mean (SD)	**Comfort** Mean (SD)	**Interest** Mean (SD)	**Classroom** Mean (SD)	**Career** Mean (SD)
Condition					
Control	2.21 (0.32)	1.89 (0.37)	2.69 (0.60)	2.44 (0.34)	1.94 (0.43)
Treatment	1.82* (0.35)	1.86 (0.34)	2.72 (0.55)	2.15* (0.38)	1.94 (0.36)
Gender					
Male	1.99 (0.41)	1.90 (0.39)	2.58* (0.60)	2.34 (0.44)	2.00 (0.38)
Female	2.04 (0.38)	1.87 (0.34)	2.75 (0.55)	2.29 (0.36)	1.92 (0.39)

interest in computing [$F(1, 275) = 0.00, p > 0.05, 1 - \beta = 0.05$], and role of computing in their careers [$F(1, 274) = 0.16, p > 0.05, 1 - \beta = 0.07$].

The results also reflect the known gender imbalance in computer science with males having significantly more interest in computing than females [$F(1, 275) = 2.99, p < 0.05, 1 - \beta = 0.60$]. Gender, however, was not a significant factor for defining computational thinking [$F(1, 276) = 1.35, p > 0.05, 1 - \beta = 0.21$], comfort with computing [$F(1, 279) = 0.38, p > 0.05, 1 - \beta = 0.10$], incorporating computational thinking into the classroom, [$F(1, 281) = 1.24, p > 0.05, 1 - \beta = 0.20$], and the role of computing in their careers [$F(1, 274) = 2.48, p > 0.05, 1 - \beta = 0.35$]. There was no significant interaction between condition and gender on any of the survey categories.

Table III presents the descriptive for each of the five categories in the computing attitude survey. Table IV presents the ANOVA results. Table V displays the aggregated responses on all 21 items as compared between the treatment and control groups.

4. DISCUSSION

We conducted a quasi-experiment in order to discern the impact that computational-thinking modules have on preservice teachers' understanding of computational thinking and attitude towards computing. The treatment group received a computational-thinking module, whereas the control group did not receive the module. Results showcased that the computational-thinking module influenced preservice teachers'

Table IV. ANOVA Results

Factor		df	F-statistics	p-value	Power
Condition					
	Definition*	1,276	70.94	<0.00	1.00
	Comfort	1,279	2.32	0.13	0.33
	Interest	1,275	0.00	0.99	0.05
	Classroom*	1,281	38.87	<0.00	1.00
	Career	1,274	0.16	0.68	0.07
Gender					
	Definition	1,276	1.35	0.24	0.21
	Comfort	1,279	0.38	0.54	0.10
	Interest*	1,275	4.99	0.02	0.60
	Classroom	1,281	1.24	0.26	0.20
	Career	1,274	2.48	0.11	0.35
Condition*Gender					
	Definition	1,276	0.12	0.72	0.06
	Comfort	1,279	2.79	0.10	0.38
	Interest	1,275	0.93	0.33	0.16
	Classroom	1,281	0.23	0.62	0.08
	Career	1,274	0.55	0.46	0.11

*Indicates significant values

understanding of computational-thinking concepts as well as their thinking about incorporating computational thinking in their own classrooms. Specifically, preservice teachers who received the module thought of CT as a problem-solving approach that uses algorithmic thinking and abstraction to generate general principles.

On the other hand, the control group tended to agree more with the notion that computational thinking involves the use of computers and could not be applied without computers to solve problems. There were only a few preservice teachers in the treatment group who thought computational thinking involved the use of technology/computers, whereas a large number of the control group preservice teachers had the notion that CT was the use of computers. The results were similar when preservice teachers described ways to incorporate computational thinking in K-12 classrooms, with a significantly large number of the control group stating that it involves the use of computers. Finally, the treatment group was also more likely to report that computational thinking is something that is central to other disciplines such as art and English, and is not limited to computer science, mathematics, and other science disciplines than the control group.

Results were mixed with regards to preservice teachers' attitudes towards computing. There were no statistically significant differences between the control and experimental groups with regards to their comfort and interest in computing. A possible conjecture for this finding might be that even though the computational-thinking modules increased preservice teachers' interest in CT concepts (e.g., algorithm, abstraction), they might have realized their own lack of understanding of these computing concepts as they were able to see the relevance of those ideas to solving everyday problems. Hence, they did not feel comfortable in learning these concepts.

Given that computing principles enable and drive many of the technologies playing an integral part in today's society, it is important that students "begin to work with algorithmic problem solving and computational methods and tools in K-12" [Barr and Stephenson 2011]. The majority (97%) of all high school students already engage

Table V. Computing Attitude Survey

	Questions	Treatment M (SD)	Control M (SD)
Definition	Computational thinking is understanding how computers work	3.16 (0.58)	2.67 (0.66)
	Computational thinking involves thinking logically to solve problems	1.68 (0.58)	1.91 (0.51)
	Computational thinking involves using computers to solve problems	2.87 (0.71)	2.22 (0.60)
	Computational thinking involves abstracting general principles and applying them to other situations	1.78 (0.54)	2.14 (0.55)
Comfort	I do not think it is possible to apply computing knowledge to solve other problems	3.26 (0.52)	3.08 (0.54)
	I am not comfortable with learning computing concepts	3.03 (0.52)	2.99 (0.57)
	I can achieve good grades (C or better) in computing courses	1.96 (0.53)	1.87 (0.52)
	I can learn to understand computing concepts	1.84 (0.47)	1.85 (0.48)
	I do not use computing skills in my daily life	3.21 (0.53)	3.22 (0.64)
	I doubt that I have the skills to solve problems by using computer applications	3.15 (0.52)	3.04 (0.69)
Interest	I think computer science is boring	2.55 (0.71)	2.51 (0.77)
	The challenge of solving problems using computer science appeals to me	2.86 (0.70)	2.82 (0.71)
	I think computer science is interesting	2.60 (0.68)	2.60 (0.72)
	I will voluntarily take computing courses if I were given the opportunity	2.92 (0.67)	2.84 (0.71)
Classroom	Computational thinking can be incorporated in the classroom by using computers in the lesson plan	2.22 (0.59)	1.97 (0.53)
	Computational thinking can be incorporated in the classroom by allowing students to problem solve	1.56 (0.54)	1.87 (0.49)
Career	Knowledge of computing will allow me to secure a better job	1.87 (0.52)	1.73 (0.503)
	My career goals do not require that I learn computing skills	3.20 (0.59)	3.17 (0.57)
	I expect that learning computing skills will help me to achieve my career goals	1.88 (0.57)	1.87 (0.55)
	I hope that my future career will require the use of computing concepts	2.33 (0.62)	2.4 (0.72)
	Having background knowledge and understanding of computer science is valuable in and of itself	1.90 (0.52)	1.93 (0.61)

Respondents were asked to answer the questions with one of the following responses: Strongly Agree = 1; Agree = 2; Disagree = 3; Strongly Disagree = 4.

with computing technologies, such as social networking software, games, texting, and so forth [Lenhart et al. 2008]. However, computational-thinking concepts have not permeated through K-12 education, and there is very little research on how teachers could be prepared to incorporate CT ideas in their own teaching. Researchers have argued that successful incorporation of CT ideas in various content areas at the primary and secondary levels requires that teachers be exposed to these concepts through relevant examples [Barr and Stephenson 2011]. We have argued elsewhere that teacher education presents an ideal opportunity to introduce preservice teachers to CT concepts which they could embed in their future classrooms [Yadav et al. 2011]. The current study examined the impact of this approach on preservice teachers' understanding of computational thinking as well as their perceptions and attitudes towards the role of computing concepts in K-12 education.

Females are currently underrepresented in the entire computing education pipeline. The latest data from the College Board shows that only 22% of AP Computer Science test takers are females [Board 2012]. The latest Taulbee Survey annually conducted by the Computing Research Association shows that only 13.3% of the undergraduate degrees awarded by Ph.D. granting departments go to females [CRA 2012]. Looking at all institutions awarding undergraduate degrees, 18% of the degrees are awarded to women [CRA 2012]. The findings in our study are encouraging, as they suggest that females and males are equally comfortable with computing, and both see computing playing a role in their careers. Building on this comfort and the understanding that computing plays an important part in a broad range of careers may play a crucial role in increasing the number of females pursuing computer science.

Results suggest that introducing computational thinking into education courses was effective in increasing the preservice teachers' understanding of computational thinking. Specifically, participant responses were more sophisticated and showcased students' understanding that computational thinking was more than using computers and technology. Students also had a better grasp of how computational thinking can be integrated into their future teaching by promoting algorithmic thinking, abstraction, and problem solving (and not by merely using computers). In summary, we have shown that given relevant information in CT, preservice teachers' understanding of computational thinking and ideas about how to incorporate CT in their future classrooms increases.

4.1. Implications and Future Directions

These findings have important implications for incorporating computational thinking in education as well as other subject areas. Given that computational thinking is becoming a fundamental skill for the 21st century, it is important to introduce these concepts in content areas at the K-12 level. Specifically, computational-thinking concepts must appear as early as the primary grades, and then continue through the secondary grades and beyond [Qualls and Sherrell 2010]. We need a multidimensional approach for a systematic change to integrate computational thinking at the K-12 level [Barr and Stephenson 2011]. One way to make this change is to incorporate computational-thinking modules into core teacher education courses to expose future teachers to these ideas. Results from the current work suggest that such an approach has the potential to change future teachers' understanding of computational thinking and how it can be integrated into their classrooms. However, this approach is just one way to expose preservice teachers to computational-thinking ideas, and more work needs to be conducted in this area. Future work should involve educators and computer scientists in collaborating to develop concrete examples of how computational thinking could be embedded in the core content areas, from literacy and the arts to mathematics and science.

It is important that we develop teachers' understanding of computational thinking in the context of the subject matter they teach. Unless their knowledge is developed in that context, teachers may only gain an "abstract" understanding of CT. As a result, their knowledge will remain inert and they will be unable incorporate it into their teaching [Brown et al. 1989]. Thus, we should not only incorporate computational thinking within the framework of pedagogy (how to teach), but also content knowledge (such as math and literacy) and pedagogical content knowledge (e.g., math knowledge for teaching) [Hill et al. 2004]. In addition, efforts to develop teachers' CT toolkit should include both preservice and inservice teachers. At the preservice teacher level, researchers could develop CT modules and/or courses that are integrated and provide concrete examples within or in conjunction with their content methods courses. This approach would allow preservice teachers to form an understanding of computational thinking and how it applies to their discipline and apply the CT principles in their future pedagogical practices.

At the inservice level, Barr and Stephenson [2011] recommended providing professional development opportunities for teachers to highlight computational thinking in non-CS disciplines and providing teachers with relevant resources, activities, and curricular materials. Teachers could also participate in learning communities with educators and computer scientists, who are experienced in computational thinking. The current study did not examine how teachers would actually implement computational thinking in their classrooms after being provided with professional development. Hence, future research should examine how teachers from a variety of disciplines incorporate computational-thinking practices in their own teaching. Future research should also examine effective approaches (modules, webquests, activities, projects, etc.) to engage preservice teachers in computational thinking ideas and improving their knowledge, skills, and attitudes in computing. In summary, future work in this area needs to involve all the stakeholders (including policymakers, educators, and computer scientists) in making computational thinking an integral part of the K-12 curriculum.

REFERENCES

Vicki Allan, Valerie Barr, Dennis Brylow, and Susanne Hambrusch. 2010. Computational thinking in high school courses. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE'10)*.

Donald Ary, Lucy Jacobs, Asghar Razavieh, and Chris Sorensen. 2009. *Introduction to Research in Education*. Wadsworth.

Valerie Barr and Chris Stephenson. 2011. 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.

Lenore Blum and Thomas J. Cortina. 2007. CS4HS: An outreach program for high school CS teachers. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education (SIGCSE'07)*.

John S. Brown, Allan Collins, and Paul Duguid. 1989. Situated cognition and the culture of learning. *Educational Res. 18*, 1, 32–42.

Alan Bundy. 2007. Computational thinking is pervasive. *J. Sci. Pract. Comput. 1*, 67–69.

College Board. 2012. Program Summary Report. http://media.collegeboard.com/digitalServices/pdf/research/program_summary_report_2012.pdf.

Computing Research Association CRA. 2012. CRA Taulbee Survey. http://cra.org/uploads/documents/resources/crndocs/2012_taulbee_survey.pdf.

Jan Cuny, Larry Snyder, and Jeannette M. Wing. 2010. Demystifying computational thinking for non-computer scientists. Work in progress.

Peter J. Denning. 2009. The profession of IT: Beyond computational thinking. *Commun. ACM 52*, 628–630.

Daniel D. Garcia, Colleen M. Lewis, John P. Dougherty, and Matthew C. Jadud. 2010. You might be a computational thinker!. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE'10)*.

Susanne Hambrusch, Christoph Hoffmann, John T. Korb, Mark Haugan, and Antony L. Hosking. 2009. A multidisciplinary approach towards computational thinking for science majors. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education (SIGCSE'09)*.

David Hemmendinger. 2010. A plea for modesty. *ACM Inroads 1*, 2, 4–7.

Peter B. Henderson, Thomas J. Cortina, and Jeannette M. Wing. 2007. Computational thinking. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education (SIGCSE'07)*.

H. C. Hill, S. G. Schilling, and D. L. Ball. 2004. Developing measures of teachers' mathematics knowledge for teaching. *Elemen. School J. 105*, 1.

Andrew Hoegh and Barbara M. Moskal. 2009. Examining science and engineering students' attitudes toward computer science. In *Proceedings of the 39th IEEE International Conference on Frontiers in Education*.

A. Lenhart, J. Kahne, E. Middaugh, Rankin, C. Evans, and J. Vitak. 2008. Teens, video games, and civics. Tech. rep.

Gary Lewandowski, Dennis Bouvier, Robert McCartney, Kate Sanders, and Beth Simon. 2007. Commonsense computing (episode 3): Concurrency and concert tickets. In *Proceedings of the 3rd International Workshop on Computing Education Research (ICER'07)*.

James J. Lu and George H. L. Fletcher. 2009. Thinking about computational thinking. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education (SIGCSE'09)*.

Punya Mishra, Aman Yadav, and the Deep-Play Research Group. 2013. Rethinking technology and creativity in the 21st century: Of art and algorithms. *TechTrends 57*, 10–14.

Patricia Morreale and David Joiner. 2011. Changing perceptions of computer science and computational thinking among high school teachers. *J. Comput. Sci. Colleges 26*, 6, 71–77.

National Research Council (NRC). 2010. Report of a workshop on the scope and nature of computational thinking. The National Academies Press.

Jake A. Qualls and Linda B. Sherrell. 2010. Why computational thinking should be integrated into the curriculum. *J. Comput. Sci. Colleges 25*, 66–71.

Jeannette Wing. 2006. Computational thinking. *Commun. ACM* 49, 33–35.

Jeannette Wing. 2008. Computational thinking and thinking about computing. *Philosophical Trans. Royal Soc. A: Math., Phys. Eng. Sci. 366*, 1881, 3717–3725.

Aman Yadav, Ninger Zhou, Chris Mayfield, Susanne Hambrusch, and John T. Korb. 2011. Introducing computational thinking in education courses. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*.

Received April 2013; revised October 2013; accepted January 2014