Can Majoring in Computer Science Improve General Problem-solving Skills?

Shima Salehi*
Stanford University
salehi@stanford.edu

Karen D. Wang*
Stanford University
kdwang@stanford.edu

Ruqayya Toorawa
Stanford University
ruqayya@alumni.stanford.edu

Carl Wieman
Stanford University
cwieman@stanford.edu

ABSTRACT

Teaching students to become skillful problem solvers is a goal of university education, but it has been difficult to measure such skill or demonstrate the benefits of particular educational experiences. This paper presents a study of college students solving a problem unrelated to their academic majors. The analysis suggests that the educational experiences of Computer Science (CS) students may better train them in problem-solving than the experiences of other majors. In this study, students from a variety of undergraduate majors and grade levels were given a 15-minute problem-solving task embedded in an interactive science simulation. The complex task calls upon many problem-solving practices needed by scientists and engineers in their professions. Although this task has little resemblance to the problems encountered in a computer science course, CS students performed significantly better than students in any other major. In addition, only for CS students was there an indication of improvement in problem-solving from lower to upper grade levels. We propose that general problem-solving and computational thinking share some common practices, such as problem decomposition and comprehensive data collection. Furthermore, we present preliminary evidence that training in computational thinking is transferable to problem-solving tasks across domains and discuss how the unique features of CS programming assignments could be generalized to other science and engineering courses to foster students' general problem-solving skills.

KEYWORDS

computational thinking, problem solving, simulations, computer science education, student assessment

ACM Reference Format:
Shima Salehi, Karen D. Wang, Ruqayya Toorawa, and Carl Wieman. 2020. Can Majoring in Computer Science Improve General Problem-solving Skills?. In *The 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20), March 11–14, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3328778.3366808

*Both authors contributed equally to this research.

SIGCSE '20, March 11–14, 2020, Portland, OR, USA

ACM ISBN 978-1-4503-6793-6/20/03…$15.00
https://doi.org/10.1145/3328778.3366808

1 INTRODUCTION

Although universities emphasize the importance of problem-solving, there has been little progress in demonstrating how to measure or teach general problem-solving practices, or the specific practices that go into solving problems. We have been carrying out a research program to first identify, and then to teach and evaluate productive problem-solving practices that apply across a variety of disciplines. To do this, we have developed a tool for analyzing problem-solving based on a short activity embedded in an interactive simulation environment. The detailed description of this tool and our extensive studies will be discussed in longer publications currently in process [23].As part of a more extensive study on the teaching and evaluation of specific problem-solving practices, we used this tool to measure the problem-solving practices of university students from a variety of different majors and grade levels. We were intrigued by the qualitative observation that the students who performed well in the task were frequently majoring in computer science.

The observation led us to pursue the following research questions in this paper: 1) is there indeed a difference in problem-solving performance between CS and non-CS majors? and 2) how does the CS educational experience lead to improved problem-solving performance? This paper first presents the results of the post hoc analyses of students' performance on a 15-minute problem-solving task, followed by a discussion exploring the link between general good problem-solving practices and computational thinking using evidence from this analysis.

2 BACKGROUND

Effective science and engineering education goes beyond teaching content knowledge and encompasses training problem-solvers who can use their knowledge in practice to solve complex problems. Over the past decade, the science and engineering education community [1, 5, 17, 25, 27] has acknowledged the significance of training good problem solvers. However, there remain essential questions about teaching and learning of problem solving that are under investigation, including "What are the characteristics of good problem-solving?"; "How can problem-solving be measured?"; and "How can problem-solving be taught?" [6]. The work presented here is a small part of our larger research project investigating these three questions.

In the literature of problem-solving, a problem has been defined as a goal-oriented task for which a set of required actions to reach the desired goal is not known in advance [7, 15, 19, 20]. Extending this definition, we characterize problems in science and engineering domains as goal-directed tasks that: 1) require employing relevant scientific and engineering knowledge; 2) do not come with a prescribed set of actions to reach the goal; and 3) encompass multiple parts and have multiple possible solution paths.

We have identified eight specific practices important for solving problems with the listed characteristics [23]. These practices include:

(1) problem definition and decomposition of the problem into suitable sub-problems
(2) data collection
(3) data recording
(4) data interpretation
(5) reflection on problem definition and assumptions
(6) reflection on knowledge (what is known and what needs to be known)
(7) reflection on effectiveness of strategy used to solve the problem
(8) reflection on solution, including testing and verification of the solution

Although we derived these from empirical observations, many of these practices have been also noted in previous works related to engineering education [2, 22, 28, 30].

These practices also have some overlap with elements of computational thinking. Computational thinking (CT) refers to a set of thinking practices that are fundamental to computer scientists and can be used to solve complex problems in many disciplines [31]. Research has been undertaken in the past decade to identify and operationalize computational thinking in practice. Identified CT practices include systematic processing of information, problem abstraction and decomposition, data collection and manipulation, and testing and debugging solutions [3, 5, 8, 11, 29].

3 METHODS

3.1 The Black Box Problem

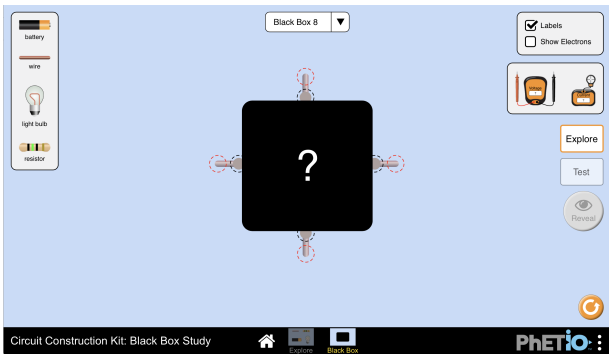

Figure 1: The user interface of a black box problem with four terminals ©PhET Interactive Simulations

To measure individual's problem-solving performance, we designed a set of complex problems named "CCK black box" embedded in the circuit construction kit of PhET Interactive Simulations (http://phet.colorado.edu) [18]. The black box problem consists of a black box hiding a circuit structure with four wires ("terminals") protruding from the box (Figure 1). Solvers are asked to figure out the circuit structure behind the black box by using electrical components and measurement tools available in the simulation interface to collect data and interpret them. The richness of the interactive environment provides capabilities for evaluating problem-solving practices far beyond what is possible with traditional paper-and-pencil tests. The data log of problem solvers' interaction with the black box problem yields detailed information on their chosen practices.

The black box problem features all the characteristics of problems in science and engineering domains listed in a previous section of the paper. Specifically, the problem requires scientific knowledge about electrical circuits, including Ohm's law, structural characteristics of circuits, as well as characteristics of different electrical components. The set of actions required for solving the problem is not clear in advance. In addition, the problem is complex with multiple parts and numerous possible solution paths. While the problem is different from the typical assignment problems that students encounter in science or engineering courses, its deep structure resembles the authentic problem-solving endeavors of scientists and engineers. They have to decide what data are needed, how to collect the data effectively, how to interpret the collected data and construct a model of the scientific / engineering phenomenon at hand, and how to evaluate and test their solution [12, 13, 23, 24].

Although the black box problem is complex and difficult enough to capture many important scientific problem-solving practices, it is constrained enough be carried out within the limited time (15 minutes) allocated in the study. Our goal is to make the black box problem a challenging, but not overwhelming, task for all participants. To this end, we first pre-screened study participants to ensure that they possess the basic content knowledge about electrical circuits required for solving the problem. Second, we adjusted the difficulty level of the problem to be suitable for this population by appropriately choosing the number of connections and the type of electrical components (wire, battery or resistor) in the black box. Generally speaking, the more connections behind the black box, the harder the problem. Nearly all the participants could meaningfully engage with the problem and come up with a proposed solution. This allows us to evaluate their problem-solving performance, even though most did not achieve a fully correct answer in the allotted time.

3.2 Participants

74 undergraduate students (43 female, 31 male) were recruited for this study via email listservs at a highly-selective R1 university. To be included in the study, students needed to (a) have taken a high school physics course (usually AP Physics) and/or a university course covering electric circuits; and (b) not be advanced majors in physics or electrical engineering. The inclusion criteria ensured that participants have some, but not extensive, background in electric circuits. Seven participants were excluded from this analysis: four

due to extremely low engagement level and/or lack of electricity knowledge; one due to prior exposure to the black box problem, and two freshmen who did not report any intended major.

The remaining 67 students were categorized into three groups according to their reported major: quantitative natural science and engineering ("*quant S&E*"), computer science ("*CS*"), and all others ("*nonquant*") (Table 1). This distribution of majors is roughly consistent with the school's overall undergraduate student body major distribution [16]. To examine the problem-solving differences across major and year, the three categories of majors were further split according to whether students are first-year or upper-level (predominately second, third, and fourth year, with two fifth-year students). First-year students would only be in their second quarter of college classes and have very limited exposure to the courses in a specific discipline, while upper-level students would have taken at least a few courses in their major.

Table 1: Major and year of participants (N = 67)

Group	CS	Quant S&E	Nonquant
N (first-year)	11	7	11
N (upper-level)	11	16	11
Majors	CS	Chemical Eng. Chemistry Civil Eng. Electrical Eng. Environmental Systems Eng. Management Science & Eng. Materials Science & Eng. Math Mechanical Eng. Product Design unspecified Eng.	African American Studies Architectural Design Biology Classics Economics Human Biology International Relations Philosophy Political Science Psychology Science Technology & Society

In a one-on-one lab setting, students first read and signed a consent form, then completed a short tutorial to become familiar with features of the simulation. This included building a complete circuit and being reminded of Ohm's law (V =IR, or voltage = current times resistance). Upon finishing the tutorial, students were given 15 minutes to figure out the hidden circuit structure behind the black box and record their solutions on paper (Figure 2). In the full study, students received experimental interventions after attempting this black box problem and proceeded to solve more black box problems to study their improvements in problem-solving practices. Here we only consider their performance on the first black box problem before any intervention.

3.3 Solution Scores

The quality of students' problem-solving performance is assessed by a rubric measuring the degree of similarity between their solutions and the correct answer. The rubric evaluates a proposed solution in three dimensions: 1) identification of the correct circuit structure; 2) identification of the correct components of the structure (e.g., two resistors); and 3) identification of the correct values of the components (e.g., 15 ohms). A solution receives a score in each of the three dimensions of zero for incorrect, one for partially correct and two for correct, making the total score range from zero to six.

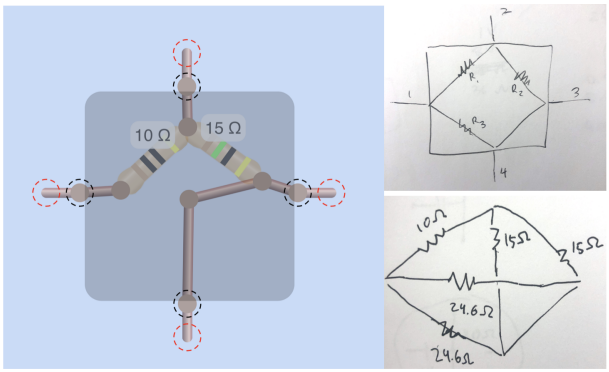

Figure 2: Circuit behind the black box and samples of solutions submitted by students

4 ANALYSIS AND RESULTS

4.1 Differences in Problem-solving Performance

Overall, the black box was a challenging problem for most participants. Students' solution scores ranged from 0 to 6, with a mean of 2.67 and a standard deviation of 1.76. Only 18% of the solutions received a score of 5 or higher. As shown in Figure 3, upper-level CS students reached substantially higher solution scores than all other groups. Non-parametric Mann-Whitney U tests were used in lieu of t-tests to examine whether there were significant differences in solution scores across the six groups because of the non-normal distribution of solution scores (Shapiro-Wilk = 0.91, $p = 0.0002$). The upper-level CS majors performed significantly better than upper-level students in both Quantitative science and engineering and Non-quant groups (CS upper vs. Quant upper: $p = 0.05$, CS upper vs Non-quant upper: $p = 0.008$).

The best argument for a causal relationship between studying a discipline and developing problem-solving skill would come from the comparison of the differences of scores with grade level. Only for CS was there an indication of a difference between first-year and upper-level students. Although the sample size was small, the difference in solution scores between CS first-year and upper-level students was still marginally significant (CS first-year vs. CS upper: $p = 0.09$). In contrast, there was clearly no difference between first-year and upper-level students in non-CS majors (Quant first-year vs. Quant upper: $p = 0.97$, Non-quant first-year vs. Non-quant upper: $p = 0.87$). These results suggest that only in CS major are students learning something that improves their problem-solving performance.

A possible mundane explanation for the change in solution scores with level could be that upper-level CS students have taken courses

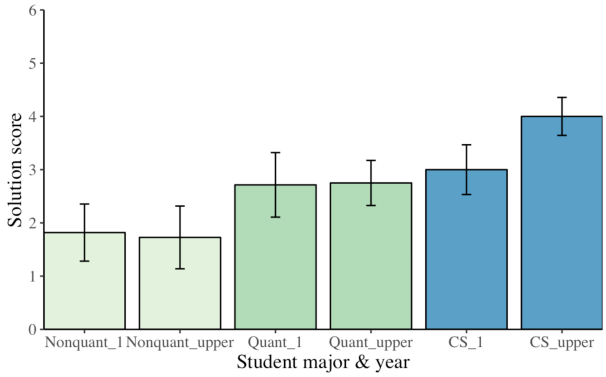

Figure 3: Average solution scores by major and year with error bars representing standard error of the mean

in which they learn more about electric circuits. This can be ruled out for two reasons. First, for this experiment, we selected our sample and designed the tutorial session to ensure that participants have comparable knowledge about circuits, further minimizing the effects of physics content knowledge. Second, there was no significant difference between Quant S&E and CS upper-level students in terms of their amount of coursework covering electric circuit (Fisher's exact test, $p = 0.66$).

We conclude that CS students probably have learned problem-solving practices in the study of their discipline and were able to apply these practices in solving the black box problem. Yet the nature and details of the problem-solving practices involved remains a question. In the next section, we present an examination of the identified problem-solving practices listed above, which contributed to CS students' problem-solving success.

4.2 Differences in Problem-solving Practices

To better understand the problem-solving processes that participants went through, a JSON file of their interaction data with the black box problem was obtained from the developers of the simulation platform. The file was first parsed into a time-stamped sequence of actions taken by individual participants when investigating the circuit configuration behind the black box (e.g., at Time 0: connect a wire to Terminal 1, at Time 1: connect a battery to Terminal 3). The Python NetworkX package [9] was then employed to analyze the structural properties of the circuits built by participants based on their time-sequenced actions. Four types of circuits were identified (Figure 4):

(1) simple circuits connecting two terminals (i.e., wires protruding from the black box)
(2) complex circuits connecting more than two terminals at a time
(3) naive circuits connecting only one terminal
(4) external circuits built separately from the black box

Drawing on existing literature in computational thinking and our own research on scientific problem solving [23], we highlight the following two practices: problem decomposition and comprehensive data collection. The interaction data log provided evidence as to the quality of the these two practices for each student.

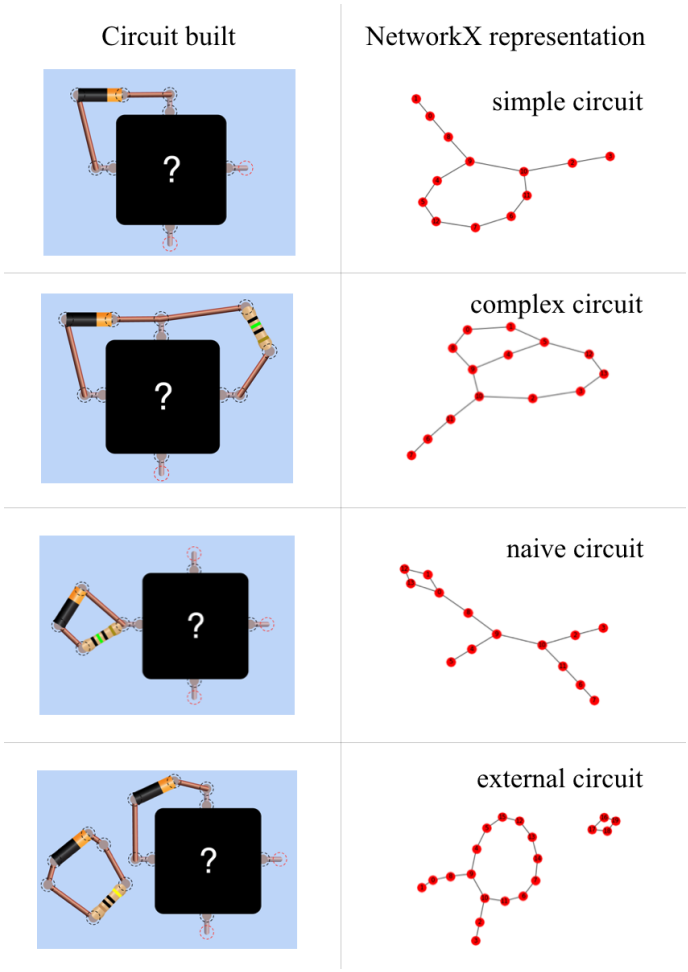

Figure 4: Different types of circuits built by students and their respective NetworkX representation in Python

Problem Decomposition For complex problems, an effective problem-solving practice is to break down the problem into smaller sub-problems that are easier to solve. In the context of the black box problem, the practice of problem decomposition is manifested in the construction of simple circuits connecting only two terminals at a time. The building of complex circuits, which connects more than two terminals, is evidence of poor decomposition practice as the resulting data from such circuits is far more difficult to interpret. A decomposition index was calculated based on the percentage of simple circuits built (out of the total number of circuits built) for individual students (Table 2). Linear regression analysis was conducted to examine the relationship between the decomposition index and a categorical variable combining year and major. The year-major variable had the following six levels: first-year nonquant (baseline), upper-level nonquant, first-year quant S&E, upper-level quant S&E, first-year CS, and upper-level CS. Upper-level CS students significantly outperformed first-year nonquants in problem decomposition ($p = 0.03$), while there was no significant difference between the other groups and first-year nonquants (upper-level

nonquant: p = 0.24, first-year quant S &E: p = 0.23, upper-level quant S &E: p = 0.15, first-year CS: p = 0.13).

Table 2: The Decomposition Index for different year & majors (percentage of simple circuits built)

	first-year	upper-level
CS	78%	89%
Quant	76%	75%
Nonquant	56%	73%

Comprehensive Data Collection Given the interconnected nature of the circuit configuration behind the black box, students need to collect data from all six pairs of terminals in order to fully investigate the problem. To measure the comprehensiveness of data collection, we assigned a binary variable "comprehensive_data_collection" to each student indicating whether they have constructed simple circuits between each of the six pairs of terminals. Table 3 presents the percentage of students who collected all the data in each major & year category. Logistic regression analysis was used to test the effects of major and year on the likelihood of comprehensive data collection. The upper-level CS students were the only group who significantly outperformed the first-year nonquant students in comprehensive data collection (p = 0.02). The difference between first-year nonquant and upper-level quant S&E student was marginal (p = 0.06), and insignificant for the other groups (upper-level nonquant: p = 1.00, first-year quant S &E: p = 0.10, first-year CS: p = 0.18).

Table 3: Percentage of students practicing comprehensive data collection for different year & majors

	first-year	upper-level
CS	45%	73%
Quant	57%	50%
Nonquant	18%	18%

Problem decomposition and comprehensive data collection are both practices that we have identified as important problem-solving practices in our previous work, and they are related to the skills emphasized in the computational thinking literature. As discussed below, these practices are explicitly called upon in many programming assignments, so it is plausible that CS students improve their problem-solving practices as they take multiple courses involving programming.

5 DISCUSSION

The results presented in this paper support the crosscutting nature of computational thinking, and provide initial evidence that computational thinking skills that are practiced in CS courses improve problem-solving in other contexts. What makes computer science education a powerful context for the development of problem-solving skills? To answer this question, we examined the instructional materials and homework assignments of several CS courses, and we identified a few particular features. First, CS instructional materials often include explicit teaching and modeling of effective problem-solving practices. For instance, students are taught to thoroughly cover the search space and avoid off-by-one errors when they learn to use loop statements to solve the Fencepost problem [4]. Furthermore, CS programming assignments are highly effective in helping students develop specific problem-solving practices. Assignments of introductory programming courses commonly take the form of creating an entire program/game "from scratch". Students are expected to come up with appropriate requirements, constraints and assumptions to make the goal of the problem better specified. These assignments also require decomposing the problem into parts/subproblems that are easier to solve, often with explicit guidance on how to do this. The following example (Table 4) illustrates how problem decomposition is expected and modeled for students in an introductory CS class assignment to create the classic arcade game of Breakout [21]:

Table 4: An excerpt from a programming assignment modeling problem decomposition

> Success in this assignment will depend on breaking up the problem into manageable pieces and getting each one working before you move on to the next. The next few sections describe a reasonable staged approach to the problem:
>
> (1) set up the bricks
> (2) create the paddle
> (3) create a ball and get it to bounce off the walls
> (4) check for collisions
> (5) finish up

The interactive programming environments also enable students to practice collecting and interpreting real-time data and iteratively construct and test proposed solutions. When debugging a program, for instance, students must learn to seek and interpret various outputs of the program in order to determine what is not working and how it could be fixed. In this way, CS programming assignments are explicitly and repeatedly supporting students developing good problem-solving practices that we see are relevant to solving a wide variety of complex problems.

6 CONCLUSION

In this study, we used an interactive simulation to examine the problem-solving performance of undergraduate students. A post hoc analysis indicates that upper-level CS students possess better problem-solving performance than students in other majors as well as first-year CS students. Unlike CS majors, non-CS majors show no problem-solving improvements from first-year to upper-level. This supports the causal link between taking CS courses and improved problem-solving performance. This interpretation is further supported by analyses of specific problem-solving practices, showing that the same practices regularly required in CS assignments are also valuable for solving the CCK black box problem. The results support Wing's claim that computational thinking "represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use"[31].

The nature of computer science assignments contrasts with many procedurally "well-structured" problems used in other STEM courses[26, 30]. Those "well-structured" problems short-circuit many of the identified authentic problem solving practices and the decisions they involve [14]. In these problems, all information needed to solve the problem is specified in the problem statement, and students simply have to choose one from a limited number of rules and principles to plug in the numbers [10]. An over-reliance on these problems in education leaves students ill-prepared for solving problems that they will encounter in the real world. However, this does not have to be the case. Just as CS programming assignments help students learn and practice computational thinking and good problem-solving practices, assignments in other STEM courses can be changed to make effective problem-solving practices explicit and required. This would likely improve students general problem-solving performance.

We note there are several limitations with this work. First, as we mentioned above, this study was not originally designed to establish causal relation between studying CS discipline and developing problem-solving skills. Future studies should more directly examine the causal relation between CS education and problem-solving performance. Longitudinal studies that track problem-solving of CS and non-CS students over the course of their undergraduate education would be better suited for this goal. Second, not all practices used in CS assignments will be applicable more generally, and similarly, for most other types of problems there are likely to be some practices that are not helped by a CS education. Those are likely to be practices that are heavily dependent on the knowledge in the field, something our black box problem was explicitly created to minimize. Third, the number of participants in the various categories is obviously small, and the participants are selected from a highly selective institution, which limits the generalization of such claims across majors and broader student populations. In addition to expanding the study populations, future work should examine more carefully the effects of background knowledge (i.e. electricity and circuits) on CCK black box problem performance and compensate for this in studying the problem-solving of students with a broader range of backgrounds. Finally, although we see that the practices effective for solving the black box problems are useful in solving other types of problems [23], the evidence that such practices are broadly transferable is still limited. Nevertheless, this work provides encouraging preliminary results on the impacts of different forms of instruction on broadly useful problem-solving practices.

ACKNOWLEDGMENTS

This research was supported by the Gordon and Betty Moore Foundation.

REFERENCES

[1] ABET. 2000. Criteria for Accrediting Engineering Programs. Retrieved July 30, 2019 from https://www.abet.org/accreditation/accreditation-criteria/

[2] Cynthia J Atman, Robin S Adams, Monica E Cardella, Jennifer Turns, Susan Mosborg, and Jason Saleem. 2007. Engineering design processes: A comparison of students and expert practitioners. *Journal of engineering education* 96, 4 (2007), 359–379.

[3] Karen Brennan and Mitchel Resnick. 2012. New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada*, Vol. 1. 25.

[4] CodeHS. [n. d.]. Fencepost Problem. Retrieved July 30, 2019 from https://codehs.com/glossary/term/38

[5] National Research Council et al. 2012. *A framework for K-12 science education: Practices, crosscutting concepts, and core ideas*. National Academies Press.

[6] Peggy Cuevas, Okhee Lee, Juliet Hart, and Rachael Deaktor. 2005. Improving science inquiry with elementary students of diverse backgrounds. *Journal of Research in Science Teaching: the Official Journal of the National Association for Research in Science Teaching* 42, 3 (2005), 337–357.

[7] Karl Duncker and Lynne S Lees. 1945. On problem-solving. *Psychological monographs* 58, 5 (1945), i.

[8] Shuchi Grover and Roy Pea. 2013. Computational thinking in K–12: A review of the state of the field. *Educational researcher* 42, 1 (2013), 38–43.

[9] Aric Hagberg, Pieter Swart, and Daniel S Chult. 2008. *Exploring network structure, dynamics, and function using NetworkX*. Technical Report. Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

[10] David H Jonassen. 2000. Toward a design theory of problem solving. *Educational technology research and development* 48, 4 (2000), 63–85.

[11] Irene Lee, Fred Martin, Jill Denner, Bob Coulter, Walter Allan, Jeri Erickson, Joyce Malyn-Smith, and Linda Werner. 2011. Computational thinking for youth in practice. *Acm Inroads* 2, 1 (2011), 32–37.

[12] Richard Lehrer and Leona Schauble. 2007. Scientific thinking and science literacy. *Handbook of child psychology* 4 (2007).

[13] Richard Lesh and Helen M Doerr. 2003. Foundations of a models and modeling perspective on mathematics teaching, learning, and problem solving. *Beyond constructivism: Models and modeling perspectives on mathematics problem solving, learning, and teaching* (2003), 3–33.

[14] Nathan J McNeill, Elliot P Douglas, Mirka Koro-Ljungberg, David J Therriault, and Ilana Krause. 2016. Undergraduate students' beliefs about engineering problem solving. *Journal of Engineering Education* 105, 4 (2016), 560–584.

[15] Allen Newell, Herbert Alexander Simon, et al. 1972. *Human problem solving*. Vol. 104. Prentice-Hall Englewood Cliffs, NJ.

[16] Stanford Registrar's Office. 2018. Enrollment Statistics. https://registrar.stanford.edu/everyone/enrollment-statistics/enrollment-statistics-2018-19

[17] Honor J Passow and Christian H Passow. 2017. What competencies should undergraduate engineering programs emphasize? A systematic review. *Journal of Engineering Education* 106, 3 (2017), 475–526.

[18] Katherine Perkins, Wendy Adams, Michael Dubson, Noah Finkelstein, Sam Reid, Carl Wieman, and Ron LeMaster. 2006. PhET: Interactive simulations for teaching and learning physics. *The physics teacher* 44, 1 (2006), 18–23.

[19] George Polya. 2004. *How to solve it: A new aspect of mathematical method*. Number 246. Princeton university press.

[20] Frederick Reif. 1995. Understanding basic mechanics. *Understanding Basic Mechanics, by Frederick Reif, pp. 288. ISBN 0-471-11624-6. Wiley-VCH, January 1995.* (1995), 288.

[21] Eric Roberts and Jerry Cain. 2017. Assignment3 Breakout! Retrieved July 30, 2019 from https://web.stanford.edu/class/cs106j/handouts/18-Assignment3.pdf

[22] Moshe F Rubinstein. 1974. *Patterns of problem solving*. Prentice-Hall.

[23] Shima Salehi and Carl Wieman. 2019. *Science in Practice: Improving Scientific Problem-solving through Reflection*. Ph.D. Dissertation. Stanford University, Stanford, CA.

[24] Christina V Schwarz, Brian J Reiser, Elizabeth A Davis, Lisa Kenyon, Andres Achér, David Fortus, Yael Shwartz, Barbara Hug, and Joe Krajcik. 2009. Developing a learning progression for scientific modeling: Making scientific modeling accessible and meaningful for learners. *Journal of Research in Science Teaching: The Official Journal of the National Association for Research in Science Teaching* 46, 6 (2009), 632–654.

[25] Sheri Sheppard, Anne Colby, Kelly Macatangay, and William Sullivan. 2007. What is engineering practice? *International Journal of Engineering Education* 22, 3 (2007), 429.

[26] Namsoo Shin, David H Jonassen, and Steven McGee. 2003. Predictors of well-structured and ill-structured problem solving in an astronomy simulation. *Journal of research in science teaching* 40, 1 (2003), 6–33.

[27] NGSS Lead States. 2013. Next Generation Science Standards: For States, By States. Retrieved July 30, 2019 from http://www.nextgenscience.org/

[28] Paul S Steif, Jamie M Lobue, Levent B Kara, and Anne L Fay. 2010. Improving problem solving performance by inducing talk about salient problem features. *Journal of Engineering Education* 99, 2 (2010), 135–142.

[29] David Weintrop, Elham Beheshti, Michael Horn, Kai Orton, Kemi Jona, Laura Trouille, and Uri Wilensky. 2016. Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology* 25, 1 (2016), 127–147.

[30] Kristen Bethke Wendell, Christopher G Wright, and Patricia Paugh. 2017. Reflective decision-making in elementary students' engineering design. *Journal of Engineering Education* 106, 3 (2017), 356–397.

[31] Jeannette M Wing. 2006. Computational thinking. *Commun. ACM* 49, 3 (2006), 33–35.