Adaptive Scaffolding in an Intelligent Game-Based Learning Environment for Computer Science

Wookhee Min, Bradford Mott, and James Lester
North Carolina State University, Raleigh, North Carolina, USA
{wmin, bmott, lester}@ncsu.edu

Abstract. Recent years have seen growing interest in intelligent game-based learning environments that simultaneously provide intelligent tutoring systems’ adaptive pedagogical functionalities and promote learners’ engagement through the rich narratives and situated challenges of games. A key research question posed by game-based learning is how to deliver effective adaptive scaffolding to support tailored learning processes for individual students. This paper discusses adaptive pedagogical strategies in the context of introducing computer science principles to middle school students. We explore two adaptive scaffolding strategies: (1) adaptive task selection based on students’ problem-solving performance, and (2) adaptive hint generation by exploring the problem’s solution space. Furthermore, we present specific approaches that leverage dynamic Bayesian networks and solution spaces to achieve each of these.

Keywords: Game-based learning environments, narrative-centered learning environments, adaptive scaffolding.

1 Introduction

Recent years have witnessed growing interest in teaching computer science principles to K-12 students, with a goal of fostering computational thinking and encouraging participation in computer science. Several computer science education interventions have been developed for K-12 students in the U.S. under the proposed course objectives of the AP Computer Science Principles course [1]. These middle school computing interventions include introducing visual programming toolkits such as Kodu [2], Alice [3], and Scratch [4–5] that enable students to learn fundamental programming concepts while designing and developing games and interactive stories. In addition, simplified programming languages, such as Logo [5] and Squeak [6], as dialects of formal programming languages, have been introduced for teaching programming fundamentals to K-12 students.

As a novel support to learning this subject, intelligent game-based learning environments hold great potential as a tutoring framework for middle grade computer science education. Since these learning environments leverage intelligent tutoring systems’ adaptive pedagogical functionalities, it is possible to create dynamically tailored learning experiences that are simultaneously effective and engaging. Key
benefits include their support for enhancing student motivation and engagement, their capacity as a platform for complex problem-solving scenarios, and their ability to deliver adaptive hints, feedback, and scaffolding to support students’ learning processes. A broad range of game-based learning environments are under investigation in both classrooms [7–9] and for training [10].

Challenges in building intelligent game-based learning environments fall into two categories. First, while they provide active and self-paced learning by allowing considerable autonomy, it is possible for students to unintentionally spend time on problem-solving tasks for which they already have mastery, and inadvertently skip problem-solving tasks where they possess gaps in knowledge. A key step in addressing this inefficiency is devising computational models that predict students’ knowledge, and adaptively suggesting targeted problem-solving tasks that help remediate students’ misconceptions or gaps in knowledge [11–14]. Second, it is important to avoid undesirable situations in which students might struggle within problem-solving tasks for an extended period of time resulting in frustration and disengagement in learning. Intelligent game-based learning environments should be able to overcome these challenges by suggesting tailored hints and feedback at relevant times [15].

In this paper, we present the ENGAGE game-based learning environment for teaching computer science principles to middle school students. The curriculum underlying ENGAGE’s narrative is derived from the CS Principles course [1]. This paper makes the following contributions: (1) proposing a game-based learning environment augmented with adaptive scaffolding to effectively teach computer science principles to middle grade learners, and (2) devising computational approaches to accomplishing intervention strategies for adaptive scaffolding. With predictive models in place, it will be possible to achieve adaptive pedagogical planning that (1) dynamically tailors narrative plots, problem-solving tasks, and curricular content, and (2) provides personalized hints and feedback, based on the knowledge and learning activities of individual learners.

2 Related Work

In game-based learning environments, recent research has investigated techniques to model students’ knowledge, abilities, plans, and goals in order to provide adaptive scaffolding. Dynamic Bayesian networks have been utilized to predict students’ self-regulated learning abilities [16], and to predict student knowledge to inform interactive narrative adaptations [12]. Semi-supervised learning has shown potential to improve accuracy rates over supervised learning when predicting students’ problem-solving performance [14]. Collaborative filtering techniques have been investigated to predict scores on a sequence of embedded assessments and dynamically personalize event sequences preventing students from reaching suboptimal solutions in an open-ended learning environment [13]. Ha et al. have leveraged Markov logic networks to recognize students’ goals from observed sequences of player actions and thus individualize pedagogical scaffolding [17].

Several classic projects examined diagnostic and tutoring plans in the context of programming problem-solving tasks. Meno-II was an error-oriented program analysis
system that diagnoses non-syntactic errors that students make, and remediating students’ misconception related to the errors [18]. Bridge introduced iterative plans from a natural language description to a formal programming description based on the assumption that novice programmers tend to approach programming tasks with natural language constructs [19]. Proust addresses intention-based analysis, in which students’ programs are interpreted at three levels of diagnosis: goals, plans, and code [20]. More recently, hint generation systems have utilized (1) representative solutions and non-solutions that are specified by instructors or based on student solutions [21–22], and (2) a sequence of actions toward solutions represented in interaction graphs [15]. Our work investigates approaches to both adaptive task selection and adaptive hint generation in the context of a game-based learning environment for teaching computer science principles to middle school students.

3 ENGAGE Learning Environment

ENGAGE (Figure 1) is a game-based learning environment for middle school computer science education leveraging a CS Principles-based curriculum [23]. Drawing on curriculum design activities and recommendations from teachers and students, ENGAGE consists of three levels: Introduction, Digital World, and Big Data.

![ENGAGE game-based learning environment](image)

**Figure 1.** ENGAGE game-based learning environment.

Students play the role of the protagonist who has been sent to an underwater research station to determine why all communication with the station has been lost. Unbeknownst to them, the facility has been taken over by a rogue scientist. As students explore the station, they must progress through the three levels of the game consisting of a series of rooms. Each room presents students with a set of computational problems such as programming a moving platform to cross a pit, and they solve the problems with a visual programming interface in order to advance to the next room. Throughout their investigation, students interact with a cast of non-player characters offering clues and relevant details via dialogue. The overarching narrative is advanced through cinematics and character dialogue while learning is scaffolding through on-screen hints and animated vignettes.
The Introduction level introduces students to the game mechanics (e.g., controlling their character, using the visual programming interface), and several computational problem-solving tasks that can be completed by programming devices (e.g., moving platform, crane). The CS Principles Objective Statement focus of this level is, “Programming languages are a tool through which people implement algorithms to solve problems using their creativity and skills.” In the Digital World level, the concept of binary numbers that can represent various types of data such as decimal numbers, alphabetical characters, and colors are presented to students. The CS Principles Objective Statement focus of this level is, “Binary is an abstraction that computers use to communicate, and the meaning of any binary sequence will depend on its interpretation and use.” In the Big Data level, students analyze diverse data sets by calculating simple descriptive statistics and visualizing the data sets (e.g., histograms, scatter plots), and discover empirical findings from the data analysis. The CS Principles Objective Statement focus of this level is, “People use computers to analyze data and discover new information with practical applications to real-world problems.” Through a series of highly interactive learning activities, students uncover the mystery of the station, and eventually solve it.

4 Adaptive Scaffolding

In ENGAGE, students complete a series of computational problem-solving tasks situated in the game world. A task requires multi-step procedures, which are associated with a set of concepts such as conditional and iterative statements. To successfully complete a task without guessing, students are required to clearly understand what the current task’s goal is, and have some understanding of the concepts since there could be numerous ways to solve a problem due to its ill-defined nature.

We propose two adaptive scaffolding strategies that in the future we plan to investigate within the ENGAGE learning environment, one that adaptively selects the next problem-solving task and one that provides step-by-step targeted guidance (e.g., hint, feedback) within a problem-solving task [24]. On the one hand, it is possible that students may find a sub-optimal solution, and inadvertently skip mastering advanced concepts in which they may possess gaps or misconceptions; on the other hand, it is possible that students may struggle to identify the next step within a problem-solving task, which could cause them to be distracted. In order to cope with these undesirable situations, intelligent game-based learning environments must be capable of supporting adaptive scaffolding satisfying instructional objectives. In the following subsections, we describe two adaptive scaffolding strategies: adaptive task selection and adaptive hint generation, along with an example problem-solving task drawn from the ENGAGE learning environment.

4.1 Example of Problem-Solving Task

As students progress through ENGAGE, they encounter computational challenges that they complete using programmable devices within each room. For example, students are prompted to write a program to operate moving platforms in order to get across
pits or move from one ledge to another. In an earlier phase of the game, students start with a relatively simple task where they can complete the task by writing a program with 4 move forward blocks; later, students encounter more complex tasks, for which they have to use multiple rotate blocks along with multiple move forward blocks, or more efficiently with repeat blocks. Figure 2 depicts an in-game problem-solving task that has a programmable moving platform and pipe obstacles that block the path towards the other ledge. Additionally, Figure 3 presents three sample solutions to this problem-solving task; from left to right, solutions become more efficient in terms of the number of programming blocks used.

Figure 2. A moving platform problem-solving task with obstacles that block paths through the center or the right.

Figure 3. Representative answers to the moving platform task illustrated in Figure 2. (Left) using all command blocks, (Middle) using three single loops, (Right) using a nested loop.

4.2 Outer Loop: Adaptive Problem-Solving Task Selection

To provide outer loop scaffolding [24], ENGAGE must predict whether a student needs extra problem-solving tasks to complement gaps in knowledge or remediate misconceptions in knowledge. Previous work has addressed similar challenges by modeling
students’ knowledge with machine learning formalisms such as probabilistic graphical models [11–12]. We propose a dynamic Bayesian network-based approach to supporting student modeling as illustrated in Figure 4, which for illustrative purposes is limited to the example in Figure 2 and Figure 3, but is scalable as needed. In Figure 4, an observation variable denotes a feature vector based on a student’s submitted answer that holds the following information: (1) whether she solved the problem with only move and rotate blocks (e.g., Figure 3, left), single loops (e.g., Figure 3, middle), or a nested loop (e.g., Figure 3, right), and (2) whether the submitted answer successfully completes the task. Latent variables consist of four binary variables that represent the student’s mastery of different concept knowledge (Move Command, Rotate Command, Single Loop, Nested Loop), a binary variable (Goal) that indicates whether the student correctly understands the goal for the current task, and a binary variable (Interface Using Skill) that quantifies how skillful the student is in using the programming interface.

![Figure 4](image-url) Conceptual illustration of student modeling leveraging a dynamic Bayesian network.

It is noteworthy that there is a directed link from the Single Loop concept knowledge to the Nested Loop concept knowledge because mastering the latter strongly depends on mastering the former. The Goal and Interface Using Skill variables can be utilized to detect cases where a student submits a wrong answer, even though the student has demonstrated mastery of every concept. For example, the Goal variable for moving platform tasks is to program moving platforms to reach specified destinations.

Once a new observation becomes available, the student model is updated, and the learning environment can predict which concepts she has and has not successfully mastered by calculating the probability on each of the hidden variables. A scaffolding strategy to cope with these unmastered features is to introduce intermediate problem-solving tasks between the current task and the task that is supposed to be the next task if the outer loop scaffolding does not exist. Intelligent game-based learning environ-
ments can satisfactorily deal with this uncertainty. Once they decide to introduce new tasks, they can dynamically relocate the player to a new room, and smoothly advance the game along with individualized narratives. In these intermediate tasks, learning environments might need to restrict some abilities that the programming interface originally possesses, in order to guide students to practice and learn unmastered features. For example, if a student failed to master the single loop concept, then the next task can limit the number of available Move and Rotate Command blocks as a way to enforce using Repeat blocks to complete the task. In another situation where it appears that the student does not understand the goal of the task, the systems can provide a reminder. This strategy is used in conjunction with one of the inner loop scaffolding strategies, the goal-setting hint [25].

4.3 Inner Loop: Adaptive Hint Generation

The inner loop [24] provides personalized hints, especially on steps when students get stuck in a problem-solving task. Challenges posed by generating automatic hints include (1) how to efficiently build an effective automatic hint generation system, (2) when to provide hints, (3) what needs to be corrected in the current student’s answer, and (4) what types of hints to provide to enhance students’ learning outcomes. In this paper, we focus on the third and fourth item when tutoring computer science principles for middle grade learners.

Several approaches have addressed these challenges by leveraging a data-driven technique, such as constructing solution spaces for individual tasks based on previous collected corpora and reasoning about next steps in various types of machine learning frameworks [15]. We plan on leveraging an automatic hint generation system for ENGAGE, based on the data-driven solution space approach. A plausible design for

![Figure 5. Abstract of a hand-authored solution space to the second solution in Figure 3; the light-colored rectangle (State3) indicates a subgoal state.](image-url)
solution spaces is leveraging a graph structure [15, 26], where a node denotes an intermediate program in a parse tree form (e.g., abstract syntax tree) and an edge denotes a programming action (e.g., dragging and dropping a programming block) that makes a state transition. Once spaces are constructed, the next step is locating a student’s answer on the closest node in the solution space by calculating distances between states (e.g., graph edit distance). Finally, a hint can be suggested based on a scaffolding policy (e.g., the first state in the shortest path to the goal state). Once a final solution is submitted, it can be collaboratively utilized to assess the student’s evolving performance (e.g., mastery of concept knowledge), update the student model, and adaptively select the next problem-solving task in the outer loop phase.

Figure 5 depicts part of a hand-authored solution space on the example problem-solving task. This specifically illustrates part of the students’ answers that can be derived from the middle and right approaches of Figure 3, in which students solve the task using repeat blocks. The mechanics for visual programming in ENGAGE are inspired by Scratch [4], where subsequent blocks are dragged together with a currently dragged block, and thus some edges (e.g., State_3–State_4, State_3–State_7) are unidirectional, while most of the edges are bidirectional.

Types of hints can be differentially selected based on the policy that a learning environment employs. One type of hint is to scaffold conceptual knowledge that the student is predicted to have not mastered based on the current student model. Another type of hint is to direct students to the next state toward the solution. With a single goal-oriented policy, the learning environment can suggest a hint that leads to the shortest path to the final state, while with a multi-sub-goal oriented policy, it can suggest a hint that leads to the next subgoal state instead of the final goal state. The latter procedural assistance might be helpful, particularly for low performing students, by periodically setting intermediate subgoals to a problem-solving task. For example, in Figure 5, State_8 is set as a subgoal state, which is a pivotal state to reach a final solution. In this case, by leveraging the sub-goal oriented policy, a student who is currently in State_3 can get a detailed instructional hint on how to move on to State_2 or State_8, so that she can successfully progress one step forward to the sub-goal state. Because intelligent game-based learning can serve as a framework to effectively deliver hints through animated vignettes and narratives from non-player characters, it is anticipated that ENGAGE will help students persist even when confronted by significant challenges.

5 Conclusion

Intelligent game-based learning environments hold great potential for providing adaptive scaffolding because of their support for rich personalized narratives and situated game challenges that promote student motivation and engagement in learning. We have presented two adaptive scaffolding strategies in the context of the ENGAGE game-based learning environment that introduces computer science principles to middle school students. To address challenges in the “outer” loop, we have proposed a dynamic Bayesian network-based approach to adaptively selecting the next task based on the predicted concept mastery, and for the “inner” loop, we have presented a solu-
tion-space based approach to provide targeted guidance to students within a problem-solving task. Key characteristics (e.g., dynamic placement of the player, tailored narratives, animated vignettes) of game-based learning environments are particularly well suited to effectively deliver both kinds of adaptive scaffolding, and thus can create dynamically tailored learning experiences that are simultaneously effective and engaging.

It will be important to conduct classroom studies of learners using the ENGAGE environment, and studies are currently underway. In the future, it will also be important to incorporate dynamic Bayesian networks and solution spaces into the learning environment to evaluate the impact of individualized pedagogy in run-time intelligent game-based learning environments. Additionally, it will be useful to design a broad array of student modeling and adaptive scaffolding strategies. Together, it is hoped that these endeavors will yield intelligent game-based learning environment designs that enhance middle school students’ knowledge of computer science and computational thinking.

Acknowledgements. The authors wish to thank colleagues from North Carolina State University for their assistance. This research was supported by the National Science Foundation under Grant IIS-1138497. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors.

6 References