Proceedings of the 2nd International Conference on Interdisciplinary Humanities and Communication Studies
DOI: 10.54254/2753-7048/33/20231497

A Dynamic Interactive Learning Interface For Computer Science Education: Programming Decomposition Tool

Fanfei Meng[1,a,*] and Chen-Ao Wang[2,b]

[1]Northwestern University, 633 Clark St, E Chen-Ao Wang vanston, IL 60208, USA
[2]Tianjin University, Weijin Road Campus: No. 92 Weijin Road, Nankai District, Tianjin, 300072, China
a. fanfeimeng2023@u.northwestern.edu, b. wangcha@tju.edu.cn
**corresponding author*

Abstract: When the coding assignments step into more complicated and structured periods for individuals, how to cultivate the skills have become the urgent needs for people who are engaged in sophisticated coding development in both academics as well as industries, especially those who have some basic knowledge in their undergraduate studies. In this paper, a state-of-art interactive tool: Programming Decomposition Tool (PDT) is proposed to help people enhance their complicated coding skills and further to broaden their CS learning horizons. The tools have two unique features: 1. The learning interface is adaptive and dynamic with respect to the detection of the knowledge depth of users. 2. Difficult learning parts will be decomposed into several easier sessions for better acquisition.

Keywords: Computer science learning, Interactive tools, Programming educations, Dynamic interface

1. Introduction And Intellectual Merits

With the rising attention paid to advanced Computer Science Learning (CSL), how to lead the mentioned groups to mastering complicated coding abilities has gradually become the focus of CSL educators and scholars. In general, although the early studies have laid certain foundations on advanced CS learners and they are able to develop programming tasks in certain difficulty levels, their professional reserves are not enriched enough when it comes to the more sophisticated frameworks and logics including the developed mathematical understanding as well as the mature programming capacities.

To be specific, some doctoral students in Chemistry/Material Engineering obtained basic CS knowledge from introductory courses and mastered some fundamental programming skills. Upon diving into higher-level studies in their domains, programming and CS knowledge are not mandatory in their daily learning. Consequently, their programming realizations gradually fade away and are hard to recall in the junior or senior stages. The obtained knowledge fails in tackling the sophisticated coding challenges in work and it is less efficient for them to take courses unrelated to their research projects. But for some of the group, programming and algorithm applications are pivotal to their doctoral research and it even forces them to be engaged in higher-level computing development, which form great practical gaps between their CS foundations and research expectations.

Regarding the issue, in this paper, the paper would like to propose a state-of-art interactive tool:

DOI: 10.54254/2753-7048/33/20231497

Programming Decomposition Tool (PDT) to help people enhance their complicated coding skills and further to broaden their CS learning horizons. The objectives of the tools are summarized as the following:

- The tool aims at constructing a dynamic knowledge boundary detection mechanism to de-compose users' coding tasks for easier development. The coding decomposition is cohesively related to the boundary score quantified by the Bayesian Classifier and Markov Process and the score is dynamics computed based on the real-time monitor on users' behaviors and accomplishments.
- The decomposition will be responsive to the score indicating the acquisition of the users' programming knowledge level upon the score available;
- The analysis on multiple users corresponds to the scores of each individual. The users with closed scores will be classified into the same groups, which effectively evicts the reliance on the large amount of prior user experiences.

There are two distinctive characteristics of the tool:

- The qualitative analysis on users is dynamic and flexible. The level of knowledge boundary score is the reflection on users programming abilities, which constitute the causal inference on knowledge levels of users. Different from the one-time judgment, the inference is adaptive in users' learning and operating process instead of the fixed coding decomposition. Once the system detects the significant change on the boundary score, the presented de- composition will be flexibly accommodated to the changes.
- The quantitative analysis on users is efficient in processing interactions between users and tools. The introduction to computing methods reduces the reliance on the constructions on prior experiences and the decomposition originates from simple and accurate mathematical representations on users where the score derives from multiple weighted components on knowledge mastery and coding concept cognitions.

2. Background

It is expected to reinforce the perceptions of CSL [1] and finally benefit the prosperity of the CSL community. The quantitative approach is promising to empower the recognitions on the knowledge, abilities, skills of users and extend the suitable decomposition on their developments, which furthers advanced programming professions and fosters the higher-level coding logistics, thoughts as well as abilities for users [2]. In addition, the tools are endowed with the time-series properties, which bolsters the real-time and responsive mechanism on the inter- actions between users and interfaces. Undoubtedly, more frequent and constructive interactions are involved in the CSL via the state-of-art tool to facilitate the explorations and realizations on Computer Science Education concerning the diverse qualitative evaluations on knowledge levels, which is likely to germinate the groundbreaking findings on both CSL and cognitive science.

It is probable that users encounter the obstacles to practice their advanced coding skills [3]. The qualitative analysis is dynamic and highly relevant to the action chain of users, to some extent, every action that user takes definitely affects the performance of the tool. If the users do not follow the tutorial and guidance provided by the tool and take subjective actions, the inaccurate analysis and feedback is generated in likelihood and leads to the learning inefficiency. What is more, users may be in dilemma for a long time even if the interface has made optimal and adaptive recommendations based on the real-time performance. In this case, the interface is unable to facilitate users assignments and the actual help should be considered to be out of the tool itself. In other words, users are expected to implement some CS/Maths knowledge to improve themselves instead of merely relying on decomposition ways.

DOI: 10.54254/2753-7048/33/20231497

3. Related Work

In retrospect to the development of Computer Science Learning (CSL) tools [4], on the one hand, most are focused on enhancing the fundamental coding grammar teaching without paying attention to guiding students to structure the coding logics in a more accessible, hierarchical and straightforward way. [5] is concerned with a game-designed environment to help advanced learners to practice mathematical coding. They develop four kinds of blocks to engage users in collaborative games with reward mechanisms to train their mathematical thinking. All mathematical logic is decomposed into a series of steps and visualized to present users its computational process. The research shed great light on block-based and game-based programming regarding the transmission from the text-based to other modality-based formats, which only concentrates on the light-layer coding logic visualization rather than the advanced thinking beyond the direct logics.

[6] introduces an effective way to quantify the boundary of knowledge indicating the depth of knowledge that learners can reach in linguistic studies. The measurement is developed from Bayesian inference, design the knowledge graph as a multidimensional pointer and discuss the assessment approaches. On the one hand, the research is enlightening to lead scholars considering the significance of quantitative methods on assessing user behaviors. On the other hand, they do not explore the dynamics of qualitative analysis on adapting the tool's accessibility to users for producing a better sense of practice according to the acquisitions on users knowledge level. They are pioneers in detection computing but not contributive to applying the computation to design a practical tool or interface. Some tools have proposed the concept that the learning process is a stepwise process and prototyped tools could be simulated as progressive modes to align with the formation process of human cognition on complex systems.

[7] demonstrates the necessity and effectiveness of constructing a real-time feedback mechanism to help users in their learning process instead of feedback only available upon the completion of their assignments. Nevertheless, their methods mostly rely on the record experiences collected from testers or past users, which means the maturity of the designed tools is tightly relevant to the abundancy [8-12], reliability and generality of original datasets. If the original collections are unable to satisfy the three principles such as incompetence in obtaining enough representative individuals and acquiring the full coverage on configurations of users behaviors, the designs are definitely supposed to fail in assisting users with achieving preset goals.

4. Design Argument

4.1. Desired Functionality

The configuration of PDT aims at assisting users to develop advanced algorithms in a more accessible way and finally abulates their programming thoughts. In general, PDT firstly reveals the knowledge deficiency between their current knowledge levels and target algorithm levels and then simplifies the difficulty of the whole assignment for users.

The decomposition acts as a bridge between the underdeveloped coding abilities and expected coding abilities for users, which emphasize on predicting, detecting and analyzing the obstacles that users encounter. Normally, users are not very clear about their foundations and have higher expectations on their target work so that the early attempts are usually unsuccessful. In order to bridge the underdeveloped issues (knowledge, skills etc.), they are likely to be subject to acquiring the less meaningful CS knowledge due to unclear realization of where they should improve. PDT, clarifies the orientation of enhancement for users and matches their ongoing projects with a simplified, friendly and transferring decomposed development style when they are engaged in their target programming. As a consequence, embedded self-learning is a real-time perception of users and

DOI: 10.54254/2753-7048/33/20231497

generates dynamic qualitative analysis on programming capacities with the help of quantitative methods.

On the other hand, the mechanism of programming assistance offers the valuable chances to obtain more sensitive, informative CS education data in mathematical representations. PDT identifies as a double functional tool: monitors and analytics. The monitoring functions are reflected in its helper roles, and for analytics, PDT precisely tracks a series of time-based actions and approaches more accurate programming qualitative inference on programming of users. For most real-time tools, the feedback and assistance is clustered based on action itself rather than the mathematical representation, which impose the restrictions on summarizing the quality classification on closed groups. With the help of the computing system, it is much easier for researchers to investigate the behavior correlationship and inference, which is enlightening to capture the trivial change on human learning as well as cognitions.

4.2. Centralized Characteristics

The feedback and adjustment mainly rely on the similar user computing instead of all users in the documents. Thus, the PDT can achieve more accurate and adaptive decomposition across the programming development. Most common real-time feedback learning tools are dependent on mindsets collected from all users or experimental testers, which do not distinguish the users with different habits, background as well as realization of their own coding issues. The PDT are more sensitive to the users with closed knowledge level through the boundary computing mechanism, reveal which users in the past can be referred for ongoing users and facilitate the construction of specific labeling on existing algorithm libraries.

The PDT will be advantageous on timesaving for users who are eager to strengthen their CS foundations. The relatively accurate analysis on the knowledge boundary will generate guidance for users when they are in a dilemma without any progress. At this point, the PDT is likely to recommend the learning materials for users and advise them to review the materials before they return to the coding interfaces. Enlightened by the recommendations, users are clearer about what they are expected to study before achieving the desired goals and navigate the most relevant knowledge to acquire and finally shorten their learning time.

5. Approach

The main prototypes of PDT can be described as two interacting parts and one self-learning session. Briefly, users are firstly encouraged to participate in a CS foundation survey and outline a rough knowledge configuration for an intelligent system firstly. After that, the dynamic feedback system is supposed to detect the behavior of users and enable the decomposition to be adaptive and intelligent. In the final stage, the system is refurbished to update its prior experience library to elaborate its own functionality.

To be specific, the adaptive decomposition derives from the knowledge boundary computing methods. Boundary computing is a quantitative approach to detecting the users' coding foundations as well as perceptions on the target mathematical knowledge. The method can assume a scenario where there is a graduate student with a desire to code Decision Tree step by step, who ever took a CS-related course before but time went through for several years. Under the circumstance, the PDT will invite the user to participate in a knowledge testing before the target algorithm coding interface, which comprises several coding questions relevant to the target algorithm as well as basic data structure. Upon the completion of the test, the tool analyzes the performance of users combining it with the saved information from past users and then calculates the coding knowledge boundary score of the user. For the contextual example, the user's knowledge boundary score is approximately

DOI: 10.54254/2753-7048/33/20231497

equivalent to a freshman undergraduate in CS major after the testing computation. In light of the boundary score, the initialized decomposition will split the Decision Tree algorithm into 5 to 6 modules com- pared with normal 4 modules to ease the programming difficulty. With the investigation on user background and the qualitative reasoning on the programming knowledge, the decomposition is likely to be more user-friendly and personal-adaptive.

Although the qualitative investigation has been completed before the generation of the decomposed coding interface, the rough derivation cannot represent the actual programming level of users. As a result, the tools will continue its learning in the operation process of users. To be detailed, the user has started to the first programming subsection of Decision Tree and encountered the bottleneck in branch pruning coding and been stuck in it for a while. PDT would like to detect the long stay and quantify the new information into the boundary computing system. The developed algorithms smartly percept the signal and dynamically update the boundary score. If the updated boundary score significantly varies from the beginning score, the decomposition system will revise the subsequent decomposition and structure the coding subsections to adjust the needs of users in a more considerate way.

Last but not least, each users' operation history will be documented by the tools and correlated with each individual. The correlation is based on two aspects: a. How to upgrade the initialized background investigation in a more accurate computing derivation and launch the decomposition in a more scientific way; b. How to update the existing library of algorithms with the help of users' practice. Take the example of the user as well, he/she has finished all subsections and his/her activities are documented by the system. The users are assumed to be stuck in the branch pruning, entropy computation, feature classification for long-run and these three parts will be labeled by the tools and given higher weights if the next users have the closed evaluating boundary score in the qualitative analysis. For instance, there is another user deploying the PDT to develop the Decision Tree algorithm and securing the similar knowledge boundary score. PDT will pay more attention to users' coding when they are proceeding to the labeled sessions and decompose the whole coding based on the previous users' experiences. Similarly, the PDT will add new data into the library to update algorithms with similar coding difficulties.

6. Conclusion

The plan is to recruit the participants for 5 groups and invite them to participate in the PDT experiments. Each group contains at least 3 members with closed knowledge boundary scores computed by constructed algorithms. After the group and preliminary background investigations have been conducted, every participant is required to code different algorithms in respect to the adaptive de- compositions. The paper would like to analyze all data collected from users and place more attention on the significant change on individual boundary scores. Once the scores are revised vastly, manual inquiry and interview will be performed on users and know about their progress as well as difficulties, which also works as auxiliary qualitative analysis.

The method also expect to prototype several plans for each algorithm in the library. For instance, it is likely that K-mean algorithms could be decomposed into 3, 6, 9 parts for different groups, respectively and analyze their development process. What is more, even if the same group has closed boundary scores, the method would like to investigate the trivial score effect and explore more possibilities for individual differences.

The method are dedicated to innovating the boundary score in a cascade mode, which aligns with the neutral structures of brains. The human cognitions usually follow the order that starts from most fundamentals and further to higher-level perceptions with deepening acquisitions and accumulated experiences. In light of this, the computing algorithms of boundary score will also start from the low-level boundary detection for users without rapid skipping to high-level detection. To be specific, in

DOI: 10.54254/2753-7048/33/20231497

the background survey, the first or first two questions will be very easy for users to make sure that they have any CS foundations before. If they fail in the most beginning questions, the PDT is supposed to recommend basic open-source CS courses for them before actual coding rather than directly introduce them to access the console interfaces.

References

[1] *Mordechai Ben-Ari. Constructivism in computer science education. Acm sigcse bulletin, 30(1): 257–261, 1998.*

[2] *Torsten Brinda, Hermann Puhlmann, and Carsten Schulte. Bridging ict and cs: educational standards for computer science in lower secondary education. Acm Sigcse Bulletin, 41(3):288–292, 2009.*

[3] *Yu Chen Fanfei Meng, Lele Zhang. Fedemb: An efficient vertical and hybrid federated learning algorithm using partial network embedding. anonymous preprint under review, 2023a.*

[4] *Yu Chen Yuxin Wang Fanfei Meng, Lele Zhang. Sample-based dynamic hierarchical transformer with layer and head flexibility via contextual bandit. anonymous preprint under review, 2023b.*

[5] *Sally Fincher and Marian Petre. Computer science education research. CRC Press, 2004.*

[6] *Fanfei Meng. Transformers: Statistical interpretation, architectures and applications. preprint under review, 2023.*

[7] *Fanfei Meng, Lalita Jagadeesan, and Marina Thottan. Model-based reinforcement learning for ser- vice mesh fault resiliency in a web application-level, 2021.*

[8] *Thomas L Naps, Guido Ro¨ßling, Vicki Almstrum, Wanda Dann, Rudolf Fleischer, Chris Hund- hausen, Ari Korhonen, Lauri Malmi, Myles McNally, Susan Rodger, et al. Exploring the role of visualization and engagement in computer science education. In Working group reports from ITiCSE on Innovation and technology in computer science education, pp. 131–152. 2002.*

[9] *Manijeh Razeghi, Arash Dehzangi, Donghai Wu, Ryan McClintock, Yiyun Zhang, Quentin Durlin, Jiakai Li, and Fanfei Meng. Antimonite-based gap-engineered type-ii superlattice materials grown by mbe and mocvd for the third generation of infrared imagers. In Infrared Technology and Applications XLV, volume 11002, pp. 108–125. SPIE, 2019.*

[10] *Andreas Scha¨fer, Jan Holz, Thiemo Leonhardt, Ulrik Schroeder, Philipp Brauner, and Martina Ziefle. From boring to scoring–a collaborative serious game for learning and practicing mathematical logic for computer science education. Computer Science Education, 23(2):87–111, 2013.*

[11] *Shuhan Wang, Fang He, and Erik Andersen. A unified framework for knowledge assessment and progression analysis and design. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, pp. 937–948, 2017.*

[12] *Lisa Yan, Annie Hu, and Chris Piech. Pensieve: Feedback on coding process for novices. In Proceedings of the 50th acm technical symposium on computer science education, pp. 253–259, 2019.*