Exploring the Role of ChatGPT in Undergraduate Programming Education: A Fine-Grained Analysis of Students' Behaviors and Inquiry Patterns

Dan Sun
Chinese Education Modernization Research Institute
Hangzhou Normal University
Hangzhou, China
dansun@hznu.cn
https://orcid.org/0000-0003-2467-7406

Yi Cao
Department of Engineering Education
Virginia Tech
Blacksburg, US
caoyi@vt.edu
https://orcid.org/0000-0003-3988-837X

Fan Xu
College of Education and Human Ecology
The Ohio State University
Columbus, US
xu.3849@osu.edu
https://orcid.org/0000-0002-2518-4075

Chengcong Zhu
Zhejiang Xiaoshan High School
Hangzhou, China
m18969036832@163.com

***Abstract*— This research full paper describes the exploration of ChatGPT-supported programming among undergraduate students. Generative artificial intelligence (AI) can provide efficient and personalized intelligent services and technical support for education. As a typical generative AI language model, the application of ChatGPT in programming has gained widespread attention in the industry. However, limited research has provided empirical evidence on how learners utilize ChatGPT in learning to program. In this study, we have dived into the programming process of 36 college students by analyzing their programming behaviors and knowledge inquiry questions at a fine-grained level. The assessment of student's final projects, computer screen recordings (2160 minutes in total), and ChatGPT log data were collected. The study results show that learners first used ChatGPT as a useful resource to guide their learning process. They tended to copy codes or error messages to ChatGPT for its feedback and directly copy the code or solutions given by ChatGPT. Secondly, learners' knowledge inquiry with ChatGPT was mainly at the superficial- and medium-level. Lastly, the interview revealed the pros and cons of using ChatGPT in learning to program. The study puts forward suggestions on how to utilize ChatGPT to assist college students in programming education, intending to provide insights for improving learning outcomes and efficiency.**

Keywords—Generative AI, programming education, learning behavior, learning analysis, undergraduate students

I. INTRODUCTION

In recent years, with the rapid development of information technology, generative artificial intelligence (AI) technology has been widely used in various fields. The launch of the generative AI tool ChatGPT (Chat Generative Pre-trained Transformer) has aroused heated discussions in the field of education. As an intelligent dialogue system based on a large language model, ChatGPT can assist in learning and teaching through input, interaction, and the generation of natural language texts [1-3].

These AI advancements have greatly impacted programming education in particular. Learning to program at the university level poses considerable challenges, encompassing the intricacies of coding, debugging, and computational thinking. It is difficult for college-level learners to debug errors in their code, so they often need support and guidance throughout the process [4]. With its adept code generation and problem-solving capabilities, ChatGPT offers a powerful support system to assist students in navigating the complexities of programming concepts and completing various programming tasks [5]. Seamlessly integrated into programming environments, ChatGPT boosts developers' productivity in programming [6].

However, how ChatGPT reshapes student learning behaviors and cognitive approaches in programming courses has not yet been fully explored. An in-depth analysis of learners' programming processes can help educators and learners adopt more effective teaching and learning strategies and approaches [7]. Therefore, by employing comprehensive frameworks and learning analytics methods to analyze programming behaviors and knowledge inquiry questions, this study delves into the nuanced ways learners interact with ChatGPT. The following research questions (RQs) were addressed:

RQ1. What are the behavioral patterns of college students in ChatGPT-assisted programming?

RQ2. What is the quality of learners' inquiry questions in ChatGPT-assisted programming?

RQ3. What are the pros and cons of ChatGPT-assisted programming?

The study offers valuable insights into the dynamics of learning programming in an AI-augmented environment and ChatGPT's potential for enhancing learning outcomes. The findings enrich our understanding of generative AI's role in education, particularly programming education, paving the way for more effective teaching and learning strategies under this trend.

II. Literature Review

A. Generative AI in Programming Education

Generative AI heralds a paradigm shift in the application of AI systems in education, surpassing the conventional tasks of classification, clustering, or prediction performed by traditional machine learning models [8]. By training on vast datasets, generative AI systems can generate new data outputs such as text, images, audio, or computer code, resembling human creative expression. This paradigm holds significant implications for the creation, delivery, and personalization of educational content. For instance, with appropriate prompts, generative AI models could create high-quality instructional content tailored to specific disciplines or learning objectives, significantly reducing the time teachers need to spend on preparation activities [8]. Furthermore, generative AI could offer truly adaptive instructional content and personalized learning experiences based on student's learning patterns, knowledge gaps, and preferences identified through machine analysis of their demographic and learning data.

Traditional programming tools typically comprise a software development environment, a programming language, libraries, and other related components. These tools often demand learners to possess specific prior programming knowledge. For example, the Python IDE is designed exclusively for the Python language and requires learners to master its basic concepts and operations. Although cross-platform integrated development environments (IDE) like Flutter or Microsoft Visual Studio support multiple languages, learners still need to be familiar with specific syntax and concepts to operate within these environments. Based on a large language model, ChatGPT can communicate with users, understand their needs, and provide corresponding responses through natural language processing [9]. Compared to traditional programming tools, ChatGPT can be utilized by learners with limited prior programming experience.

Researchers have highlighted various advantages that ChatGPT offers over other programming tools, such as its ease of access and prompt responses [10]. Additionally, it supports multiple languages and provides clear explanations and programming examples to enhance learners' cognitive understanding [11]. Moreover, it facilitates querying and searching, offers advanced subject resources, and promotes personalized learning [12]. With AI tools like ChatGPT primarily aiding in code explanations and debugging, students can effectively enhance their programming skills, computational thinking, programming self-efficacy, and learning motivation through practice.

However, the application of ChatGPT in programming education also presents limitations, including unstructured learning, excessive reliance on assistive tools or environments, and limited support for data structures and algorithms [6]. Researchers have also noted that current generative AI tools face challenges in complex reasoning and processing visual information [13]. Meanwhile, ethical concerns related to ChatGPT, such as bias, discrimination, privacy, security, technology abuse, transparency, and social impact, are emphasized in this field [14].

B. Analysis of Programming Learning Process

While learning performance typically remains the primary focus in programming education, the evaluation of programming processes using a variety of learning analytics methods has also garnered increasing attention [15]. Evaluating learning processes can elucidate the essence of enhancing learners' programming quality through practice [16]. Research emphasizes that formative and summative evaluations complement each other, offering a comprehensive insight into programming learning and aiding educators in better understanding potential factors that influence the programming process [17]. Thus, the evaluation of programming education should not only focus on outcomes but also pay attention to learners' performance in the learning process and employ a range of analytical methods to assess students' programming abilities.

Previous empirical studies have employed various analysis methods to illustrate different aspects of the programming learning process. Wu et al. categorized research on programming process analysis into three main categories: research on cognitive models of the programming process, research on programming environments and supporting resources, and research on programming education to enhance learners' abilities [7]. Turkle et al. observed that in the process of programming learning, "tinkers" typically develop solutions gradually through iterative debugging and code modification in small increments, while "planners" tend to design a more comprehensive solution before coding and make larger edits to the code each time [18]. Sun et al. utilized clickstream analysis, lag sequence analysis, and quantitative content analysis to examine the behaviors, discourses, and cognitive levels of students in pair programming [19]. They suggested that students exhibited different characteristics in terms of social interaction, cognitive engagement, and final programming performance [19]. The aforementioned studies demonstrate that employing various learning analytics to analyze students' programming learning processes is beneficial in illustrating different aspects of learning outcomes.

III. Methods

A. Context and Participants

The study was conducted in an Object-Oriented Programming course at a typical university in southeastern China during the Spring semester of 2023. A total of 36 sophomores majoring in Educational Technologies participated in this study. The course comprised five sessions. The learning objectives include understanding fundamental programming concepts such as variables, loops, and functions, and gaining proficiency in the Python language. Additionally, this course aims to help students develop skills in using AI for specific tasks like generating code, debugging, providing feedback, etc.

PyCharm served as the IDE for Python in this study. We deployed the open-source project ChatGPT Next Web (NextChat), utilizing the open API provided by OpenAI to access the GPT-3.5-turbo model in the form of an independent website with full functionality. On NextChat platform, students can initiate thematic conversations with ChatGPT by typing their questions in the main window.

The course was designed in five instructional sessions, each lasting 80 minutes. During the first four sessions, the instructor taught basic concepts of Python programming, including the introduction of Python (e.g., IDLE, input(), eval(), print()), data structure (e.g., int, float, set, list, dictionary), control structure (e.g., if, for, while), functions, and methods (e.g., Recursion, Lambda). The instructor introduced the usage of NextChat and demonstrated it in PyCharm. In the last session, students individually completed a programming project "Radar Chart" within 100 minutes in PyCharm.

B. Data Collection

Participants' performance on the final project, the behavioral data collected as students learned to program using ChatGPT, and their interactions with ChatGPT were all gathered in this study. The assessment of students' final projects was based on quality criteria, including the correctness of the code, the completeness of the functionality, and the clarity and effectiveness of the visualization. Participants' interactions and inquiries with ChatGPT were recorded through the platform's logs. Additionally, students' programming behaviors were captured by computer screen recordings (without audio) during the Radar Map Task in the last course session.

The final session was chosen to collect behavioral data for two reasons. First, the participants completed a programming project from start to finish in this session, allowing us to comprehensively understand how the students incorporated ChatGPT into their problem-solving process and better demonstrate their programming skills. Second, the students became more familiar with the use of ChatGPT through their experiences in previous sessions, leading to increased engagement in the last session. We collected 60 minutes of recordings per participant, resulting in a total of 2160 minutes of recordings across all participants. In addition, to answer the last research question, we gathered data from students through semi-structured interviews.

C. Data Analysis

We used clickstream analysis to analyze the recordings and determine students' programming behaviors. Two researchers utilized an iterative coding process based on a validated coding framework by Sun et al. [19]. They first watched the recordings to establish the initial coding of programming behaviors and reached an agreement on the initial coding framework (as shown in Table I) through discussions. Then, the two researchers independently coded segments every 10 seconds in a recording (about 30 minutes), according to the coding framework. The coding results showed an inter-coder reliability of 0.80, indicating that the coding scheme was clear and consistent. Finally, the researchers used the coding scheme to analyze the remainder of the recordings of the programming process.

Upon completion of coding, descriptive statistics were employed to analyze patterns in learners' behaviors when learning to program with the assistance of ChatGPT. This study also adopted the Lagged Sequence Analysis (LsA) method to explore transitions across programming learning behaviors. R Studio and the LagSeq package were used to conduct LsA.

TABLE I. CODING FRAMEWORK FOR PROGRAMMING LEARNING BEHAVIORS

Category	Code	Behavior	Description of Behavior
Resources	UT	Understanding Task	Viewing programming task details through the task window
	RAM	Referring to Additional Materials	Referring to resources provided by the instructor
Code Editing	CP	Coding in Python	Writing code in PyCharm software
	DP	Debugging in Python	Debugging code in PyCharm software
	UPC	Understanding Python Codes	Understanding code by moving the mouse back and forth over the codes
	CRC	Checking Radar Chart	Examining output radar chart in PyCharm software
	RCM	Reading Console Message	Reading error messages in the PyCharm console
ChatGPT-Assisted Coding	ANQ	Asking New Questions	Asking new questions in ChatGPT independently
	PCM	Pasting Console Message	Pasting error message from console in ChatGPT
	PPC	Pasting Python Codes	Pasting Python code in the ChatGPT
	RF	Reading Feedback	Reading feedback in the ChatGPT
	CPC	Copy and Paste Codes	Copying code from the ChatGPT and pasting to PyCharm
Others	FC	Failure in ChatGPT	Being unable to get real-time feedback due to a technical failure of the platform (server, network, etc.)
	IO	Idle Operation	No operational behavior

For students' interactions with ChatGPT, we adapted an established framework for open coding of knowledge inquiry by Ouyang and Dai [20]. We analyzed the quality of learners' initial questions (Knowledge Inquiry) and their subsequent questions about the ChatGPT feedback (Feedback Inquiry) at three levels: shallow, medium, and deep. Data from each participant were organized in an Excel sheet. The questions asked by the participants were identified and divided into units in chronological order, resulting in a total of 446 coding units that were coded.

Then, two researchers manually coded the questions based on the coding framework in Table II to determine the final codes. Comparing the consistency between the two researchers, we achieved an inter-coder reliability of 0.83. Finally, all the data were transformed into the format required for cognitive network analysis to visualize learners' knowledge inquiry activities in ChatGPT [21] and analyzed using the online cognitive network analysis tool named webENA.

Final, a thematic content analysis was conducted by two researchers to identify common themes across the interview data, aiming to find common perceptions of ChatGPT's pros and cons.

TABLE II. CODING FRAMEWORK FOR KNOWLEDGE INQUIRY QUESTIONS

Dimension	Category	Code	Description
Knowledge inquiry (KI)	Irrelevant Knowledge Exploration	IKI	A participant explores information unrelated to the discussion topics.
	Superficial-level knowledge inquiry	SKI	A participant explores information related to the discussion topics without explicitly stating his/her own ideas, arguments, or perspectives.
	Medium-level knowledge inquiry	MKI	A participant presents his/her own ideas, arguments, or perspectives without a detailed explanation or supporting resources, statistics, or personal experience.
	Deep-level knowledge inquiry	DKI	A participant explicitly elaborates his/her own ideas, arguments, or perspectives with a detailed explanation or supporting resources, statistics, or personal experience.
Inquiry Content (IC)	Superficial-level Inquiry Content	SIC	A participant simply presents (dis)agreement, asks questions, or seeks clarification without explicitly stating his/her own ideas, arguments, or perspectives.
	Medium-level Inquiry Content	MIC	A participant extends another participant's ideas, arguments, or perspectives with a detailed explanation or supporting information, resources, statistics, or personal experience.
	Deep-level Inquiry Content	DIC	A participant extends, contends, and deepens the ideas, arguments, or perspectives proposed by other participants with detailed explanations or supporting information, resources, statistics, or personal experience.

IV. RESULTS

A. Descriptive Analysis of Programming Behaviors

A total of 3985 behaviors were generated by 36 students, and the frequency and distribution of each behavior are shown in Table III. In general, Code Editing and ChatGPT-assisted Coding are the most frequent behaviors, although other behaviors like Failure in ChatGPT and Idle Operation also occur frequently. Resources were the least frequent category of behaviors.

Among the Code Editing behaviors, the one with the highest frequency is Coding in Python (CP), followed by Understanding Python Codes (UPC) and Reading Console Messages (RF). This indicates that learners are constantly understanding and writing code during the programming process and seeking help by interacting with ChatGPT. Among the behaviors of ChatGPT-Assisted Programming, besides Reading Feedback (RF), the behaviors that occur more frequently are Asking New Questions (ANQ) and Pasting Console Messages (PCM).

TABLE III. FREQUENCY OF PROGRAMMING LEARNING BEHAVIORS

Category	Code	Frequency	Percentage
Resources (8.76%)	UT	91	2.28%
	RAM	258	6.47%
Code Editing (62.76%)	CP	1569	39.37%
	DP	240	6.02%
	UPC	498	12.50%
	CRC	79	1.98%
	RCM	115	2.89%
ChatGPT-assisted Coding (19.67%)	ANQ	146	3.66%
	PCM	115	2.89%
	PPC	62	1.56%
	RF	403	10.11%
	CPC	58	1.46%
Others (8.81%)	FC	338	8.48%
	IO	13	0.33%

B. Results of Lagged Sequence Analysis

The network visualization of the LsA results is shown in Fig. 1. The node size represent coding behaviors with their respective frequencies. Yule's Q is used to indicate the strength of transition associations, ranging from -1 to +1, where 0 means no association. The direction of behavior transitions is determined by the presence of an arrow pointing from one node to another. Fig. 1 displays some significant behavioral transition sequences as follows:

a) Sequence 1: Copy and Paste Codes -> Debugging in Python -> Reading Console Message (CPC -> DP:Yule's Q=0.96; DP -> RCM: Yule's Q = 0.94)

Learners often copy and paste code from ChatGPT, then test the code through debugging, and finally read the messages in the output console window to understand its effectiveness.

b) Sequence 2: Pasting Python Codes -> Pasting Console Message (Yule's Q = 0.85)

Learners tend to copy Python code and error messages from the console to ChatGPT to get help quickly, especially when they cannot understand or modify the code or when they encounter error messages they cannot fix.

c) Sequence 3: Reading Feedback -> Copy and Paste Codes (Yule's Q = 0.85)

Learners spend time reading the feedback given by ChatGPT, including code samples and copying them into PyCharm software for testing.

d) Sequence 4: Pasting Console Message -> Failure in ChatGPT (Yule's Q = 0.79)

Learners may encounter some technical problems when using ChatGPT to resolve error messages in their programs.

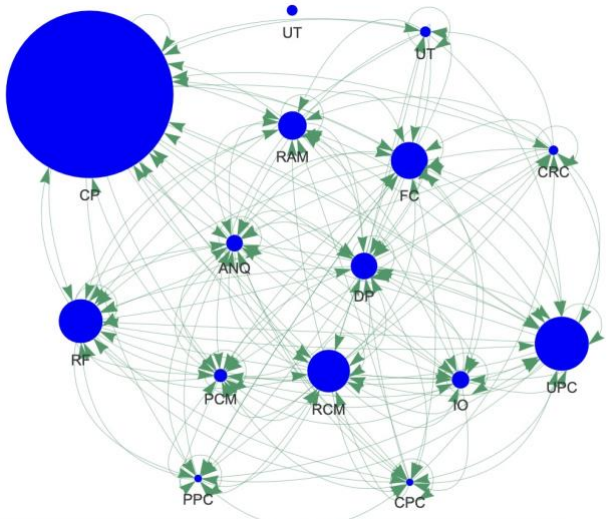

Fig. 1. Network Visualization of the Lagged Sequence Analysis.

C. *Analysis of Learners' Knowledge Inquiry in ChatGPT*

As shown in Fig. 2, learners mainly establish connections between superficial and medium-level knowledge inquiry (SKI/MKI) and connections between superficial-level knowledge inquiry and superficial-level inquiry content (SKI/SIC). While the connections to deep-level knowledge inquiry (DKI) and deep-level inquiry content (DIC) are relatively weak.

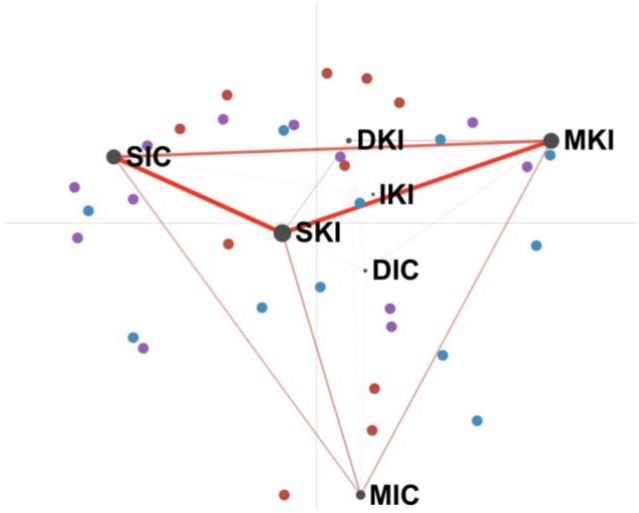

Fig. 2. Cognitive Networks of Learners' Knowledge Inquiry Activities.

D. *Student-percieved Pros and Cons of ChatGPT*

According to Table IV, among the various advantages highlighted by students, several key points emerged. First of all, ChatGPT made reference to "extensive programming knowledge" 35 times, encompassing providing coding solutions as well as helping with coding duties such as writing, understanding, and analyzing code, 35 times. Second, there were 26 mentions of the "contextualized and varied responses" that ChatGPT offered, which included contextualized comments and a variety of focused responses. Students claim that ChatGPT outperforms conventional search engines like Bing or Baidu by supplying not just rapid programming solutions but also taking into account prior inquiries and presenting several solutions to a particular issue. Thirdly, there were fifteen mentions of "accurate and efficient feedback" that resulted from combining extensive resources and cutting down on search time. Compared to well-known programming websites like CSDN or GitHub, a student found ChatGPT to be more beneficial. Fourthly, "human-like interaction" was mentioned four times. A student perceived ChatGPT as a chatbot with a high degree of humanization. On the other hand, a student pointed out an exceptional aspect of ChatGPT that sets it apart from other tools: its willingness to acknowledge incorrect answers, apologize, and accept suggestions for correction.

The second aspect highlighted the disadvantages of ChatGPT. First, 17 mentions were made of problems with erroneous codes resulting from outdated Python libraries, inadequate code compatibility, and inadequate decomposition of huge projects. A student stated, for instance, that it is impossible to expect ChatGPT to provide perfect and comprehensive code for a significant programming project. Second, the topic of "limited input and output" came up 10 times. This includes limiting the kinds of data that may be entered and generated (mostly text responses), not fully comprehending "human language", misinterpreting long-form conversational exchanges, and imposing strict guidelines on the kind of questions that can be asked (e.g., prompts). As a student pointed out, "forming questions in ChatGPT can be difficult because you need to ask logical, clear questions to get the desired response."

TABLE IV. THEMES OF CHATGPT'S PROS AND CONS EXTRACTED FROM STUDENT SEMI-INTERVIEWS

Themes	Sub-themes	*N*
Pros of ChatGPT	Expansive programming knowledge	35
	Contextualized and varied response	26
	Accurate and efficient feedback	15
	Human-like interaction	4
Cons of ChatGPT	Inaccurate codes	17
	Limited input and output	10
	Technical problems	9

V. DISCUSSIONS AND IMPLICATIONS

A. *ChatGPT-assisted Programming Learning*

The results of the programming behavior analysis show that learners tend to copy and paste code from ChatGPT and verify the correctness of the codes by debugging and reading error messages. This finding echoes Chen et al.'s conclusion that learners view ChatGPT as a useful programming learning resource and rely on it to guide their learning process [22]. Learners preferred to copy and paste Python codes and error messages from the console window directly into ChatGPT for

support and solutions. This shows that learners have some trust in ChatGPT.

In addition, learners spent a lot of time reading the feedback given by ChatGPT and copying it into the IDE to test its effectiveness. It is worth noting that there may be some issues with this approach, such as the accuracy of the feedback [6]. Therefore, learners should pay attention to the following aspects when using ChatGPT for programming learning. In the early stage of programming, learners can make full use of ChatGPT to deepen their understanding of programming problems and generate ideas for solutions. As learning progresses, learners should gradually write and edit code by themselves to improve programming ability and proficiency. Finally, it is necessary to avoid learners being overdependent on ChatGPT. Educators should monitor and correct learners' irrelevant behaviors and provide scaffolding for correctly using ChatGPT to assist learning and problem-solving.

B. Problems and Suggestions in ChatGPT-assisted Programming Learning

First, copying and pasting codes or error messages to ChatGPT may prevent learners from developing abilities in problem-solving and proficient programming. The powerful computational capability of ChatGPT may be used as a convenient substitute for student thinking and lead to "overreliance on technology" among students [23]. When learners just copy and paste codes without thinking, they may not be able to understand the logic and functionality of the code to make the necessary modifications or debugging [5]. Therefore, to use ChatGPT more effectively when learning programming, students must learn to (1) carefully read and understand the logic and operation of the code; (2) think beyond ChatGPT's answers and solve problems independently; (3) actively modify and debug code to gain a deeper understanding of its functionality; and (4) ask ChatGPT specific questions with detailed descriptions of the problem and contextual information to get more accurate solutions.

Furthermore, learners may encounter technical problems when using ChatGPT to assist in programming education. ChatGPT may not always provide accurate or complete answers, indicating that there is still room for improvement in the stability of the ChatGPT platform [6]. ChatGPT suffers from the hallucination problem common to NLP technologies, which needs to be improved by the developers of large language models through continuous iterations [24].

Students should leverage their knowledge to think about problems and create their code. They can also make use of a variety of learning resources to make up for the lack of knowledge, such as books and tutorials provided by the instructor [25]. At the same time, they can actively communicate with their peers or instructors to get timely support [19]. When using ChatGPT to assist programming education, learners should focus on the quality of their inquiries and stimulate ChatGPT to provide in-depth answers by posing challenging and open-ended questions [13]. They can also try to follow up with ChatGPT for more detailed explanations of the knowledge. Learners can enhance the quality and accuracy of ChatGPT responses by using prompts that clearly state their questions, needs, or desired responses [26].

C. Teaching strategies for ChatGPT-assisted Program Learning

The thematic analysis of student-perceived pros and cons of ChatGPT reveals that while the tool offers significant benefits, such as extensive programming knowledge, quick solutions, and engaging, human-like interactions, it also presents notable challenges. Students often highlighted ChatGPT's efficiency and the breadth of contextualized feedback, which enhanced their ability to tackle coding tasks. However, there is a concerning tendency for students to rely on copying and pasting code without fully understanding the underlying logic, which can hinder the development of deeper problem-solving skills and programming proficiency. Additionally, issues like outdated libraries, incomplete solutions, and the difficulties in crafting precise inquiries underscore the risks associated with overdependence on ChatGPT.

In addition, it is crucial to develop teaching strategies that harness the strengths of ChatGPT while mitigating its limitations. Educators should encourage active learning by guiding students to use ChatGPT as a supplementary tool rather than as a primary source. Assignments should emphasize the importance of understanding code logic and promote the practice of writing and debugging code independently before consulting ChatGPT.

Scaffolded inquiry exercises can be introduced, gradually increasing in complexity to help students develop the skill of crafting precise and logical questions for ChatGPT. This approach will enhance their ability to engage with the tool meaningfully and understand its responses within the appropriate context. Moreover, educators should promote critical evaluation by teaching students to assess the accuracy and relevance of ChatGPT's outputs. Comparing ChatGPT's suggestions with traditional resources or peer-reviewed documentation can foster a more analytical approach to learning.

To reduce overreliance on ChatGPT, integrating peer collaboration into the learning process is essential. Collaborative activities where students work together to solve problems before turning to ChatGPT can enhance understanding and provide diverse perspectives on programming challenges. Furthermore, regular monitoring of how students use ChatGPT, coupled with timely feedback, can help prevent over-dependence on the tool. Educators should offer guidance on the effective use of ChatGPT as part of a broader learning strategy, ensuring that it complements rather than replaces the critical thinking and problem-solving skills essential for programming success. By implementing these strategies, educators can help students maximize the benefits of ChatGPT while also fostering the development of the necessary skills for long-term success in programming.

VI. CONCLUSIONS

In this study, 36 sophomore students from a normal university learned to program with the assistance of ChatGPT. Their learning process for the programming tasks was collected through video recordings of class sessions, ChatGPT logs, and student interviews. Different learning analytics methods were used to analyze learners' programming learning behaviors and knowledge inquiry. Through iterative coding, a coded list of

programming behaviors and knowledge inquiry was developed, which was used to analyze the patterns of learners' programming behaviors and the quality of knowledge inquiry. A thematic content analysis on interview data was conducted and revealed the pros and cons of using ChatGPT in learning to program. Based on the findings, this study provides experience and insights on how to better utilize ChatGPT to improve programming performance and promote the development of higher-order thinking skills. However, there are some shortcomings in this study, such as the small sample size and the homogeneity of participants in terms of age and major. Future research will further explore and address these limitations to better leverage the power of generative AI to support students' programming learning.

REFERENCES

[1] Yang, Z., Wang, J. Wu, D., & Chen, X. (2023). Exploring the Impact of ChatGPT/AIGC on Education and Strategies for Response. *Journal of East China Normal University (Educational Science), 41* (7), 26-35.

[2] Zhang, R. (2023). The Impact of Generative Artificial Intelligence Technology on Education -An Interview on ChatGPT. *E-Education Research, 44* (2), 5-14.

[3] Dong, Y., Xia, L., Li, X., & Hou, Y. (2023). Analysis of the Path to Empowering Student Learning with ChatGPT. *E-Education Research, 44* (12), 14-20.

[4] Sun, L., & Hu, L. (2021). Can Programming Really Promote the Individual Development of Children? A Meta-analysis of 28 Experimental and Quasi-experimental Studies. *Journal of East China Normal University (Educational Science), 39* (11): 45-58.

[5] Lu, Y., Y, J., Chen, P., & Li, M. (2023). Application and Prospect of Generative Artificial Intelligence in Education: A Case Study of ChatGPT System. *Chinese Journal of Distance Education, 43* (4), 24-31.

[6] Yilmaz, R., & Yilmaz, F. G. K. (2023). Augmented intelligence in programming learning: Examining student views on the use of ChatGPT for programming learning. *Computers in Human Behavior: Artificial Humans, 1*(2), 100005. https://doi.org/10.1016/j.chbah.2023.100005

[7] Wu, L., Liu, Q., Bian, J., Zhang, S. & Zhang, Y. (2020). Analysis of Learners' Programming Process from the Perspective of Learning Analysis. *Modern Distance Education*, (2), 68-75.

[8] Hickey, S., Correia, A.-P., & Xu, F. (2023). *The Role of Artificial Intelligence in Learning & Development: Understanding ChatGPT – A Quick Reference*. The Ohio State University. https://books.apple.com/us/book/the-role-of-artificial-intelligence-in/id6446716738

[9] Correia, A. P., Hickey, S., & Xu, F. (2024). Beyond the virtual classroom: integrating artificial intelligence in online learning. *Distance Education*, 1-11. https://doi.org/10.1080/01587919.2024.2338706

[10] Hasanein, A. M., & Sobaih, A. E. E. (2023). Drivers and Consequences of ChatGPT Use in Higher Education: Key Stakeholder Perspectives. *European Journal of Investigation in Health, Psychology and Education, 13*(11), 2599-2614. https://doi.org/10.3390/ejihpe13110181

[11] Aikenhead, G. S., & Jegede, O. J. (1999). Cross-cultural science education: A cognitive explanation of a cultural phenomenon. *Journal of Research in Science Teaching, 36*(3), 269-287. https://doi.org/10.1002/(SICI)1098-2736(199903)36:3<269::AID-TEA3>3.0.CO;2-T

[12] Chen, C. M. (2008). Intelligent web-based learning system with personalized learning path guidance. *Computers & Education, 51*(2), 787-814. https://doi.org/10.1016/j.compedu.2007.08.004

[13] Rahman, M. M., & Watanobe, Y. (2023). ChatGPT for education and research: Opportunities, threats, and strategies. *Applied Sciences, 13*(9), 5783. https://doi.org/10.3390/app13095783

[14] Wu, X., Duan, R., & Ni, J. (2023). Unveiling security, privacy, and ethical concerns of chatgpt. *Journal of Information and Intelligence, 2* (2), 102-115. https://doi.org/10.1016/j.jiixd.2023.10.007

[15] Sun, D., Ouyang, Fan., Li, Y., Zhu, C. C., & Zhou, Y. (2024) Using multimodal learning analytics to understand the effects of block-based and text-based modalities on computer programming. Journal of Computer Assisted Learning, 40(3), 1123-1136.

[16] Pereira, F. D., Oliveira, E. H., Oliveira, D. B., Cristea, A. I., Carvalho, L. S., Fonseca, S. C., ... & Isotani, S. (2020). Using learning analytics in the Amazonas: understanding students' behaviour in introductory programming. *British Journal of Educational Technology, 51*(4), 955-972. https://doi.org/10.1111/bjet.12953

[17] Sun, D., Ouyang, F., Li, Y., & Zhu, C. (2021). Comparing learners' knowledge, behaviors, and attitudes between two instructional modes of computer programming in secondary education. *International Journal of STEM Education, 8*, 1-15. https://doi.org/10.1186/s40594-021-00311-1

[18] Turkle, S., & Papert, S. (1992). Epistemological pluralism and the revaluation of the concrete. *Journal of Mathematical Behavior, 11*(1), 3-33.

[19] Sun, D., Ouyang, F., Li, Y., & Chen, H. (2021). Three contrasting pairs' collaborative programming processes in China's secondary education. *Journal of Educational Computing Research, 59*(4), 740-762. https://doi.org/10.1177/0735633120973430

[20] Ouyang, F., & Dai, X. (2022). Using a three-layered social-cognitive network analysis framework for understanding online collaborative discussions. *Australasian Journal of Educational Technology, 38*(1), 164-181. https://orcid.org/0000-0002-4382-1381

[21] Shaffer, D. W., Collier, W., & Ruis, A. R. (2016). A tutorial on epistemic network analysis: Analyzing the structure of connections in cognitive, social, and interaction data. *Journal of Learning Analytics, 3*(3), 9-45. https://doi.org/10.18608/jla.2016.33.3

[22] Chen, E., Huang, R., Chen, H.-S., Tseng, Y. H., & Li, L.-Y. (2023). GPTutor: A ChatGPT-Powered Programming Tool for Code Explanation. In N. Wang, G. Rebolledo-Mendez, V. Dimitrova, N. Matsuda, O. C. Santos (Eds.), *Artificial Intelligence in Education. Posters and Late Breaking Results, Workshops and Tutorials, Industry and Innovation Tracks, Practitioners, Doctoral Consortium and Blue Sky* (pp. 321–327). Springer, Cham. https://doi.org/10.1007/978-3-031-36336-8_50

[23] Zheng, Y., Zhou, D., Zhang, Y., Tian, X., Wang, J. & Zheng, Y. (2023). ChatGPT from the Perspective of Computational Education: Connotation, Theme, Reflection, and Challenge. *Journal of East China Normal University (Educational Science), 41* (7), 91-102.

[24] Goddard, J. (2023). Hallucinations in ChatGPT: a cautionary tale for biomedical researchers. *The American Journal of Medicine, 136*(11), 1059-1060. https://doi.org/10.1016/j.amjmed.2023.06.012

[25] Zhu, G., & Wang, X. (2023). Operation Mode, Key Technologies, and Future Prospects of ChatGPT. *Journal of Xinjiang Normal University (Edition of Philosophy and Social Sciences), 44* (4), 113-122.

[26] Fulford, I., Ng, A. Y. (n.d.). *ChatGPT Prompt Engineering for Developers.* DeepLearning.AI. https://www.deeplearning.ai/short-courses/chatgpt-prompt-engineering-for-developers/