


## Research

# The effectiveness of intelligent tutoring systems in supporting students with varying levels of programming experience

Arthur William Fodouop Kouam<sup>1</sup> 

Received: 23 July 2024 / Accepted: 12 December 2024

Published online: 18 December 2024

© The Author(s) 2024 **Abstract**

This study investigates the effectiveness of Intelligent Tutoring Systems (ITS) in supporting students with varying levels of programming experience. Through a mixed-methods research design, the study explores the impact of ITS on student performance, adaptability to different skill levels, and best practices for utilizing ITS in heterogeneous programming classrooms. The quantitative analysis reveals significant differences in the influence of ITS on improving programming skills, the usefulness of ITS feedback, and satisfaction levels with the interface based on students' programming experience levels. Advanced students show the most significant improvement in programming skills and find the feedback more valuable, while intermediate and advanced students report higher satisfaction with the ITS interface. The qualitative analysis highlights positive impacts on the learning process, challenges faced, and suggestions for improvement. This study contributes to the literature by emphasizing the importance of tailoring ITS interventions to meet the unique needs of students with varying levels of programming experience, offering insights for educators, curriculum developers, and educational technology designers.

**Keywords** Educational technology · Intelligent tutoring systems · Heterogeneous learning environments · Personalized instruction · Programming education · Student proficiency levels

## 1 Introduction

Intelligent Tutoring Systems (ITS) are computer programs that use artificial intelligence to personalize teaching [1]. They typically consist of four components: domain knowledge, learner diagnosis, tutoring, and communication modules [2]. ITS has been developed for various subjects, including computer science and software engineering, with research addressing questions about subject areas, instructional functions, student modeling, and interface features [3]. ITS have been increasingly utilized in educational settings as they can effectively diagnose knowledge gaps and misconceptions [4]. These systems have evolved to accommodate many users and are particularly valuable in e-learning environments [5, 6]. Semantic-net principles in these systems have been shown to improve query precision and user interface [7]. Furthermore, integrating teaching activity trees in intelligent tutoring systems allows for personalized education and easier monitoring of student progress [8].

---

**Supplementary Information** The online version contains supplementary material available at <https://doi.org/10.1007/s44217-024-00385-3>.

---

✉ Arthur William Fodouop Kouam, [willyfodouop@163.com](mailto:willyfodouop@163.com) | <sup>1</sup>Saxo Fintech Business School, Sanya University, Sanya City 572022, Hainan Province, China.



The emergence of ITS in programming education has been a significant development, with a focus on simplifying the learning process for novice programmers [9]. These systems have the potential to provide automated feedback and grading, addressing the limitations of existing systems [10]. A systematic review of ITS in programming education highlights the variety of systems available, each with its unique features [11]. The use of Bayesian networks in a web-based ITS for computer programming further demonstrates the potential for these systems to guide students through online course materials and recommend learning goals [12]. Prior research has explored the use of ITS in programming education and also highlights its potential benefits in personalized instruction and skill development. Dadić [13] emphasizes the importance of knowledge representation and semantic analysis in these systems, which enable them to adapt to individual student needs. Khazanchi and Khazanchi [14] further underscore the role of artificial intelligence in providing adaptive instructions, particularly for students with disabilities. Conati and Kardan [15] extend this discussion by highlighting the use of student models to support personalization in various educational activities, including problem-solving and exploratory open-ended tasks.

However, many studies have focused on homogeneous groups of students or lacked a comprehensive examination of how ITS can adapt to diverse programming experience levels. This gap in the literature calls for further investigation into the effectiveness of ITS in supporting a range of student skill levels in programming. While some students may have prior experience or advanced skills in programming, others may be novices entering the field. Addressing these differences in programming backgrounds is crucial for maximizing the impact of ITS on learning outcomes and ensuring equity in educational support.

This study aims to investigate the effectiveness of intelligent tutoring systems in supporting students with varying levels of programming experience. Specifically, this study's research question is: ***How can intelligent tutoring systems be optimized to enhance learning outcomes for students with diverse programming backgrounds?***

The objectives of this research include.

- Assessing the effectiveness of ITS in improving the performance of students from diverse programming backgrounds
- Examining the adaptability of ITS to different skill levels and
- Identifying best practices for utilizing ITS in heterogeneous programming classrooms

Understanding how ITS can effectively support students with varying programming experience levels is crucial for advancing programming education and promoting student success. By investigating the factors that contribute to ITS's effectiveness in heterogeneous learning environments, this study provides valuable insights for educators, curriculum developers, and educational technology designers.

The paper is structured as follows: The next section reviews relevant literature on intelligent tutoring systems and programming education. Section three presents the theoretical framework of the study. Subsequently, the methodology is outlined, detailing the research design and data collection methods. The study results are then presented, and the research question is discussed. Finally, the conclusions from the study's findings and recommendations for future research and educational practice are discussed.

## 2 Literature review

This part focuses on using Intelligent Tutoring Systems in education, specifically focusing on programming education. The review highlights the increasing importance of ITS in enhancing learning outcomes, providing personalized instruction, and catering to diverse user needs. Previous studies have demonstrated the effectiveness of ITS in improving student performance and guiding learning experiences in a variety of settings. However, challenges such as the need for more in-depth explanations of AI techniques and interoperability issues remain areas of concern in the field. This section provides a comprehensive overview of the existing literature on ITS in programming education, emphasizing its potential benefits and current limitations.

### 2.1 Overview of intelligent tutoring systems in education

The current body of literature discusses using ITS in education from various perspectives. Novickis and Rikure [5] highlight the increasing importance of intelligent tutoring systems, particularly in tertiary education, and their adaptability to various users. Hoppe et al. [16] present the use of an intelligent tutoring system as a formative assessment tool in

design education, demonstrating its ability to guide students to valid solutions through supportive feedback. Intelligent Tutoring Systems have been increasingly utilized in education to enhance learning experiences. Hoppe et al. [16] and Epstein et al. [17] both highlight the effectiveness of ITS in providing formative assessment and personalized instruction, respectively. Guang [7] and Smith et al. [18] discuss the technical and design aspects of ITS, emphasizing the need for user-friendly interfaces and the integration of human factors. Furthermore, [2] provide a comprehensive review of ITS, focusing on using intelligent agents in their architecture and the shift towards adaptive instructional systems.

ITS are crucial for students and significantly impact their achievement [19]. These systems, which use adaptivity to cater to a wide range of users, have the potential to enhance the learning experience significantly. They can improve query precision and user interface in web-based education [7] and are seen as a critical component of knowledge-based expert systems [20]. The development of ITS is a crucial area of research, with a focus on knowledge representation, student modeling, planning, natural language issues, explanations, and learning [20]. Moreover, Intelligent Tutoring Systems through personalized e-learning are more effective than traditional instruction methods, particularly when combined with on-screen agents and personalization [19, 21]. They have also been shown to be beneficial in programming teaching, where they can provide automated feedback and grading [10]. Furthermore, ITS can be tailored to individual student needs via their behavior and emotions to enhance the teaching–learning process [22].

## 2.2 Discussing previous research on using ITS in programming education

A range of studies have explored using Intelligent Tutoring Systems in programming education. Gavrilović and Jovanović [23] and Lee and Baba [24] both highlight the potential of ITS in addressing the challenges of teaching programming, with [23] explicitly discussing the implementation of ITS for Java programming e-learning. Schez-Sobrino et al. [25] and Xhakaj and Aleven [26] further contribute to this discussion by proposing dynamic graphic visualizations and conceptually oriented activities to enhance the learning experience. These studies collectively underscore the potential of ITS in improving the effectiveness and accessibility of programming education. Furthermore, meta-analysis shows ITS are significantly more effective than traditional instruction methods in computer science education [3]. ITS can enhance student learning by addressing issues like lack of engagement and problem-solving skills [27]. An ITS based on informative tutoring feedback has been developed to provide real-time guidance during programming problem-solving, effectively promoting computational thinking, especially for low-level learners [28].

However, challenges remain in the field, including the need for more in-depth explanations of the relationship between AI techniques and ITS data [29]. The effectiveness of ITS in delivering different types of knowledge and guiding learning experiences has been highlighted, particularly in the Science, Technology, Engineering, and Mathematics (STEM) domains [30]. Furthermore, the interoperability of ITS for programming, particularly in the context of programming exercises, has been identified as a significant issue [31].

In sum, the literature review underscores the significant impact of Intelligent Tutoring Systems on learning outcomes and the teaching process, particularly in programming education. Previous research has highlighted the potential of ITS to enhance the effectiveness and accessibility of programming instruction through personalized feedback and adaptive learning approaches. However, challenges such as the need for more detailed explanations of AI techniques and interoperability issues pose barriers to the widespread implementation of ITS in programming education. This study seeks to address these limitations by examining the optimization of ITS to support students with varying levels of programming experience, contributing to the advancement of best practices for utilizing ITS in heterogeneous programming classrooms.

## 3 Theoretical framework: cognitive load theory

Cognitive Load Theory (CLT) provides a valuable theoretical framework for understanding how Intelligent Tutoring Systems can support student learning in programming education. CLT posits that cognitive load, or the mental effort required to process information, plays a crucial role in learning and problem-solving [32]. According to CLT, learners have a limited capacity to process information, and cognitive overload can hinder learning by inhibiting the acquisition and retention of new knowledge [33].

In programming education, students must often juggle multiple cognitive tasks, such as understanding coding syntax, problem-solving logic, and debugging errors, leading to a high cognitive load. High cognitive load in programming education can present significant challenges for students, particularly novices [34, 35]. It can lead to rote learning and

memorization, hindering the development of a deeper understanding of programming concepts [35]. ITS has been shown to mitigate cognitive overload in programming students. [36] found that an ITS combined with a social network improved problem-solving performance. Müller et al. [37] explored the feasibility of using a cognitive system, IBM Watson, as a virtual tutor in an introductory programming course, indicating potential for future use. Orhun [38] discussed the importance of knowledge representation in ITS for computer programming, a critical factor in reducing cognitive load. Yousoof et al. [39] proposed a mechanism to measure and manage cognitive load in programming instruction, which could be incorporated into ITS to enhance learning.

By applying Cognitive Load Theory to the study of ITS in programming education, this research seeks to investigate how ITS can optimize cognitive load management for students with varying levels of programming experience. The study aims to identify strategies for reducing extraneous cognitive load, optimizing germane cognitive load, and promoting effective learning techniques to enhance student engagement and skill development in programming tasks. By leveraging the principles of CLT within the design and implementation of ITS, this study aims to contribute valuable insights to the field of programming education and support the effective use of technology-enhanced learning environments in fostering student success.

## 4 Methodology, data analysis, ITS's usage, and learning activities

### 4.1 Methodology and data analysis

The study utilized a mixed-methods research design [40] to investigate the effectiveness of Intelligent Tutoring Systems in supporting students with varying levels of programming. Participants were classified into three levels of programming experience: beginners, intermediates, and advanced. Beginners are students with little to no prior programming experience, typically having completed foundational coursework or introductory programming subjects. Intermediates are familiar with programming concepts and have completed one or more programming courses, demonstrating the ability to write and debug simple programs. Advanced students have a firm grasp of programming principles, have engaged in complex projects or advanced coursework, and are comfortable using multiple programming languages. This classification allows for a nuanced analysis of the effectiveness of Intelligent Tutoring Systems across different levels of experience. This research explores the effectiveness of ITS in supporting student learning and skill development in programming tasks. The epistemological stance adopted for this study is pragmatism, which acknowledges the value of both quantitative and qualitative data in understanding complex phenomena and informing educational practices [41].

The target population for this study consists of 160 computer science students from a university in Southern China who have previous experience with ITS in programming education. A purposive sampling approach is used to select participants who have engaged with ITS in their learning process. The study includes a diverse group of participants to capture a range of perspectives and experiences with ITS. Figure 1 below is one of the ITS the respondents use in programming.

In this research, all participants utilized a single ITS known as "Blackbox." This platform was specifically chosen for its robust features designed to support programming education through interactive problem-solving and immediate feedback. Participants engaged solely with Blackbox throughout the study, which allowed us to maintain consistency in the learning experience and directly assess its effectiveness. By focusing on one ITS, we could examine its impact on students' learning outcomes and experiences, facilitating a clearer understanding of how Blackbox supports various levels of programming expertise. The singular focus on this platform helps isolate the variables specific to its design and functionality.

Data are collected through a questionnaire containing a combination of closed-ended and open-ended questions. The closed-ended questions assess students' perceptions of the effectiveness, usability, and impact of ITS on their learning outcomes. The open-ended questions allow participants to elaborate on their experiences, challenges, and suggestions for improvement regarding the ITS in programming education.

Quantitative data analysis involves descriptive statistics summarizing participants' responses to closed-ended questions. Comparative analyses, such as chi-tests and ANOVAs, examine potential differences in perceptions and outcomes based on students' programming experience levels. Qualitative data from open-ended responses are analyzed using thematic analysis [42] to identify common themes and patterns in students' feedback on their experiences with the ITS. In addition, NVivo 14 is employed for qualitative data analysis. Simultaneously, the research findings are thoroughly and comprehensively analyzed using IBM Statistical Package for the Social Sciences (SPSS) Statistics 29.0.2.0, guaranteeing a solid and comprehensive quantitative analysis. Table 1 below displays the respondents' traits.

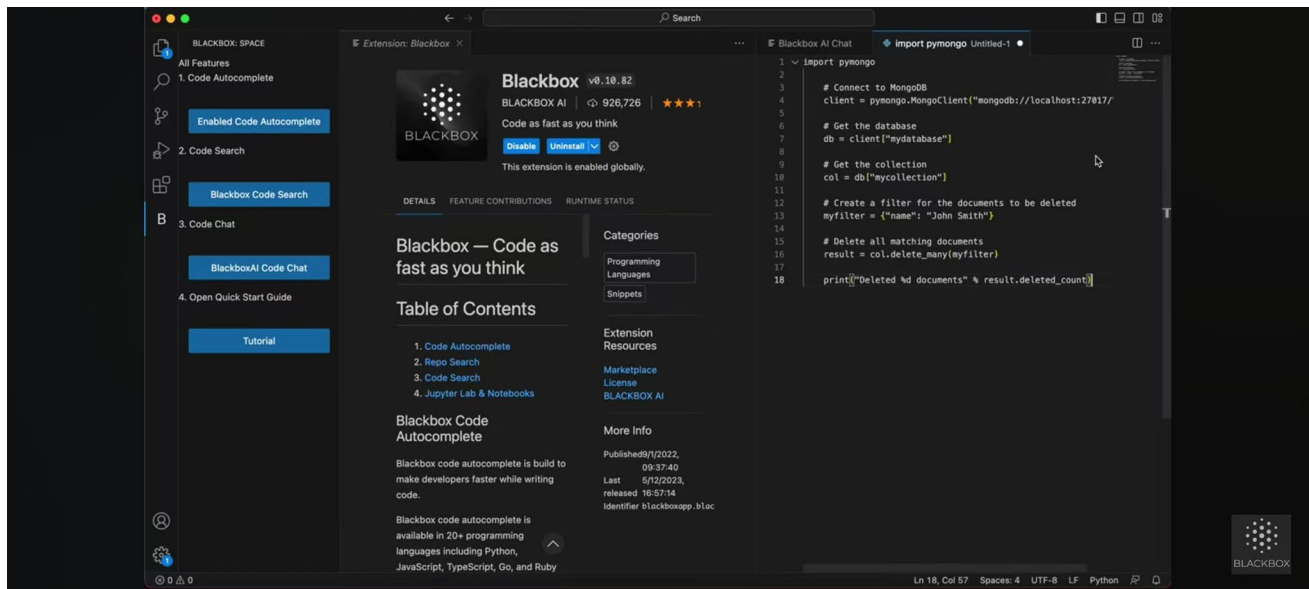


Fig. 1 ITS used by the respondents (source: the author from the respondents)

## 4.2 Usage of the ITS and learning activities

This study utilized the ITS to supplement traditional human-led teaching methods. Its integration aimed to enhance the learning experience for participants rather than replace conventional instruction. Teaching staff explicitly recommended and supported the ITS, guiding students on effectively incorporating it into their learning. This approach ensured that students understood the ITS as a valuable resource for reinforcing concepts discussed in class and accessing additional practice opportunities tailored to their needs.

Integrating the ITS into the curriculum involved structured sessions where students were encouraged to use the system following a lesson or during designated practice times. Activities included interactive exercises, quizzes, and problem-solving scenarios corresponding to the current curriculum. This methodology fostered a supportive learning environment, allowing instructors to monitor student progress and provide personalized feedback while students engaged with the ITS. The study combined structured and self-directed activities to capture a comprehensive understanding of how the ITS facilitated learning across various programming skill levels.

**Table 1** Respondents' traits

Characteristics	Options	Frequency	Percentage
Sex	Male	136	85
	Female	24	15
Level of education	Bachelor	160	160
	18–24	152	95
	25–34	8	5
Level in programming	Beginner	136	85
	Intermediate	16	10
	Advanced	8	5

**Table 2** Chi-Square Tests 1

	Value	df	Asymptotic Significance (2-sided)
Pearson Chi-Square	4.983 <sup>a</sup>	2	0.083
Likelihood Ratio	8.515	2	0.014
Linear-by-Linear Association	4.317	1	0.038
N of Valid Cases	160		

<sup>a</sup>2 cells (33.3%) have expected count less than 5. The minimum expected count is 1.20

**Table 3** Chi-square tests 2

	Value	df	Asymptotic Significance (2-sided)
Pearson Chi-Square	1.486 <sup>a</sup>	2	0.476
Likelihood RATIO	2.674	2	0.263
Linear-by-linear association	1.287	1	0.257
N of valid cases	160		

<sup>a</sup>2 cells (33.3%) have expected count less than 5. The minimum expected count is 0.40

## 5 Findings and discussions

This section presents and discusses the results of the quantitative and qualitative analysis conducted to explore the impact of Intelligent Tutoring Systems on students with varying levels of programming experience. Through chi-square and ANOVA tests, the quantitative analysis examines associations between programming proficiency levels and the effectiveness of ITS. The qualitative analysis highlights themes related to the positive impact of ITS, challenges faced by students, and suggestions for improvement. By integrating these findings, this section offers valuable insights into optimizing ITS interventions for students with diverse programming backgrounds.

### 5.1 Assessing the effectiveness of ITS in improving student performance: quantitative analysis

- Results of Chi-square tests

This study performed Chi-square tests to identify potential associations between students' programming level and the increase in their confidence and motivation in programming since using ITS. On the other hand, Chi-square tests were performed to check potential associations between students' level of programming and the usefulness of ITS in completing programming tasks. Results are displayed in Tables 2 and 3 below.

The Chi-square is 4.983, with p-values of 0.083. Given that the p-value is higher than 0.05, there is a non-significant association between students' level in programming and the increase in their confidence and motivation in programming since using ITS. In other words, there is insufficient evidence to claim a significant association between students' level in programming and the increase in their confidence and motivation in programming since using ITS. The higher p-value indicates that the difference in confidence and motivation levels between different levels of programming students may be due to random chance rather than an actual relationship.

Similarly, the following Chi-square is 1.486, with p-values of 0.476. Given that the p-value is higher than 0.05, there is a non-significant association between students' level of programming and the usefulness of ITS in completing programming tasks. Put differently, findings suggest that there is not enough evidence to suggest a significant association between students' level of programming and the usefulness of ITS in completing programming tasks. The higher p-value indicates that the difference in the perceived usefulness of ITS in completing programming tasks among different levels of programming students may be due to random chance rather than an actual relationship.



**Table 4** ANOVA 1: Influence of ITS in improving programming skills

	Sum of squares	df	Mean square	F	Sig
Between Groups	5.365	2	2.682	5.524	0.005
Within Groups	76.235	157	0.486		
Total	81.600	159			

**Table 5** Multiple Comparisons 1: Post Hoc Tests

(I) Level in programming	(J) Level in programming	Mean Difference (I-J)	Std. Error	Sig	95% Confidence Interval	
					Lower bound	Upper bound
Beginner	Intermediate	0.294	0.184	0.250	– 0.14	0.73
	Advanced	– 0.706*	0.254	0.016	– 1.31	– 0.11
Intermediate	Beginner	– 0.294	0.184	0.250	– 0.73	0.14
	Advanced	– 1.000*	0.302	0.003	– 1.71	– 0.29
Advanced	Beginner	0.706*	0.254	0.016	0.11	1.31
	Intermediate	1.000*	0.302	0.003	0.29	1.71

\*The mean difference is significant at the 0.05 level

**Table 6** ANOVA 2: Usefulness of ITS feedback on programming tasks

	Sum of squares	df	Mean square	F	Sig
Between Groups	5.835	2	2.918	3.476	0.033
Within Groups	131.765	157	0.839		
Total	137.600	159			

**Table 7** Multiple Comparisons 2: Post Hoc Tests

(I) Level in programming	(J) Level in programming	Mean difference (I-J)	Std. error	Sig	95% Confidence Interval	
					Lower bound	Upper bound
Beginner	Intermediate	0.176	0.242	0.747	– 0.40	0.75
	Advanced	– 0.824*	0.333	0.038	– 1.61	– 0.03
Intermediate	Beginner	– 0.176	0.242	0.747	– 0.75	0.40
	Advanced	– 1.000*	0.397	0.034	– 1.94	– 0.06
Advanced	Beginner	0.824*	0.333	0.038	0.03	1.61
	Intermediate	1.000*	0.397	0.034	0.06	1.94

\*The mean difference is significant at the 0.05 level

## 5.2 Examining the adaptability of ITS to different skill levels: ANOVA test results

We conducted three ANOVA tests regarding the influence of ITS in improving programming skills, the usefulness of ITS feedback on programming tasks, and satisfaction with using the interface and functionality of ITS. Tables 4 and 5 show the results of the first ANOVA test.

The significance level of 0.005 in Table 4 is lower than the set level of 0.05. Therefore, there is a statistically significant difference in the influence of ITS in improving programming skills among students of different levels of programming experience, with the advanced students showing the most significant improvement compared to beginner and intermediate students. The post hoc tests offer ample details about differences, notably between “Beginner” and “Advanced” and between “Intermediate” and “Advanced,” with significance levels of 0.16 and 0.003, respectively. Students’ proficiency in programming may impact the influence of ITS in improving their programming skills.

Tables 6 and 7 display the results of the second ANOVA test.

**Table 8** ANOVA 3: Satisfaction with using the interface and functionality of the ITS

	Sum of squares	df	Mean square	F	Sig
Between groups	5.718	2	2.859	6.813	0.001
Within groups	65.882	157	0.420		
Total	71.600	159			

**Table 9** Multiple comparisons 3: post hoc tests

(I) Level in programming	(J) Level in programming	Mean Difference (I-J)	Std. Error	Sig	95% Confidence Interval	
					Lower bound	Upper bound
Beginner	Intermediate	0.471*	0.171	0.018	0.07	0.88
	Advanced	− 0.529	0.236	0.067	− 1.09	0.03
Intermediate	Beginner	− 0.471*	0.171	0.018	− 0.88	− 0.07
	Advanced	− 1.000*	0.281	0.001	− 1.66	− 0.34
Advanced	Beginner	0.529	0.236	0.067	− 0.03	1.09
	Intermediate	1.000*	0.281	0.001	0.34	1.66

\*The mean difference is significant at the 0.05 level

The ANOVA test results for the usefulness of ITS feedback on programming tasks show a significant difference in the effectiveness of feedback based on the student's level of programming ( $F(2, 157) = 3.476$ ,  $p = 0.033$ ). The between-groups variance is 5.835, while the within-groups variance is 131.765, indicating a significant difference in mean scores between at least two of the three levels of programming.

The post-hoc tests further reveal the mean differences between the different levels of programming. The results indicate significant differences in mean scores between the beginner and advanced groups (Mean Difference =  $-0.824$ ,  $p = 0.038$ ) and intermediate and advanced groups (Mean Difference =  $-1.000$ ,  $p = 0.034$ ). At the same time, there are no significant differences between the beginner and intermediate groups (Mean Difference =  $0.176$ ,  $p = 0.747$ ).

In summary, these results suggest significant differences in the usefulness of ITS feedback on programming tasks among students of different levels of programming experience. The feedback is more effective for advanced students than beginner and intermediate students. This information can be valuable for educators and instructional designers aiming to enhance the effectiveness of ITS feedback in programming education.

Tables 8 and 9 present the results of the third ANOVA test.

The ANOVA test results for satisfaction with using the interface and functionality of the ITS show a significant difference in satisfaction levels based on the student's level of programming ( $F(2, 157) = 6.813$ ,  $p = 0.001$ ). The between-groups variance is 5.718, while the within-groups variance is 65.882, indicating a significant difference in mean scores between at least two of the three levels of programming.

The post-hoc tests (multiple comparisons) reveal the mean differences between the different levels of programming. The results indicate significant differences in mean scores between the beginner and intermediate groups (Mean Difference =  $0.471$ ,  $p = 0.018$ ) and intermediate and advanced groups (Mean Difference =  $1.000$ ,  $p = 0.001$ ). At the same time, there was no significant difference between the beginner and advanced groups (Mean Difference =  $-0.529$ ,  $p = 0.067$ ).

In conclusion, these findings suggest significant differences in satisfaction levels with using the interface and functionality of the ITS among students of different levels of programming experience. Intermediate and advanced students show higher levels of satisfaction compared to beginner students. This information can be helpful for developers and educators looking to tailor the ITS interface and functionality to different levels of programming expertise to improve user satisfaction.

### 5.3 Identifying best practices for utilizing ITS in heterogeneous programming classrooms: thematic analysis

Three themes emerged from the thematic analysis: positive impact on the learning process, challenges faced, and suggestions for improvement. This analysis offers a comprehensive understanding of the potential benefits and limitations of ITS in supporting students with varying levels of programming experience.



- *Positive impact on the learning process*

Many respondents highlighted the positive influence of the Intelligent Tutoring System on their learning process. For example, one participant stated, *"ITS helped me understand the topics I did not understand in the classes,"* while another mentioned, *"It makes me calmer."* These responses indicate that the ITS has aided comprehension and reduced anxiety during the learning process.

- *Challenges faced*

While some participants reported no significant challenges, others mentioned specific difficulties. For instance, one respondent stated, *"Sometimes I cannot find a course on the topic I want."* At the same time, another mentioned, *"Having restrictions on programming languages, tools, or project types may limit the scope and depth of my learning."* These responses suggest that there may be limitations or barriers that impact the effectiveness of the ITS in programming education.

- *Suggestions for improvement*

Participants offered suggestions for enhancing the ITS to better support their programming education needs. For example, one respondent recommended, *"There should be more cooperation with Intelligent Agent (IA),"* while another suggested, *"Provides diverse learning resources such as video tutorials, interactive experiments, and programming challenges."* These suggestions highlight the importance of incorporating additional features and resources to optimize the ITS for varied learning preferences and requirements.

The thematic analysis of the responses indicates that while the ITS positively impacts the learning process for many students, there are also challenges and areas for improvement that should be addressed to enhance its effectiveness in supporting programming education.

## 5.4 Integration of the findings

Integrating the quantitative and qualitative findings provides a comprehensive understanding of the effectiveness of Intelligent Tutoring Systems in supporting students with varying levels of programming experience. The quantitative analysis revealed significant differences in the influence of ITS on improving programming skills, the usefulness of ITS feedback on programming tasks, and satisfaction levels with the interface and functionality of the ITS based on students' programming experience levels.

The qualitative analysis further enriched these findings by highlighting the positive impact of ITS on the learning process, challenges faced by students, and suggestions for improvement. The qualitative data shed light on the benefits of ITS in aiding comprehension and reducing anxiety, as well as the limitations and barriers students may encounter when utilizing ITS in programming education. Additionally, students' suggestions for enhancing the ITS to meet their learning needs better emphasized the importance of incorporating diverse learning resources and enhancing collaboration with Intelligent Agents.

Overall, integrating quantitative and qualitative findings underscores the importance of optimizing Intelligent Tutoring Systems to cater to students' diverse programming backgrounds and preferences.

## 5.5 Discussions

- *Building on prior research findings*

The findings of the quantitative analysis align with previous research indicating the effectiveness of Intelligent Tutoring Systems in programming education. Studies like those by [19] and [10] have highlighted the benefits of personalized e-learning and automated feedback in enhancing the learning experience. The current study builds on this foundation by examining the impact of ITS on students with varying levels of programming experience, shedding light on how proficiency levels can influence the effectiveness of ITS feedback and satisfaction.

- *Deviation from anterior research trends*

In line with the studies by [23] and [24], our research deviates by uncovering the nuanced differences in the influence of ITS based on students' programming proficiency levels. Contrary to previous assumptions about ITS efficacy being consistent across all learners, our findings suggest that the effectiveness of ITS feedback and satisfaction levels varied significantly among beginner, intermediate, and advanced programming students. This departure from the general trend underscores the importance of considering students' proficiency levels in designing and implementing ITS in programming education.

- *Novelty of the study*

The novelty of this study lies in its focus on the impact of ITS on students with diverse programming backgrounds. By conducting chi-square and ANOVA tests to examine the association between programming proficiency levels and the effectiveness of ITS, the study offers valuable insights into how individual student characteristics can influence the outcomes of ITS interventions. Furthermore, the qualitative thematic analysis highlights the specific challenges and suggestions for improvement provided by students, adding a qualitative dimension to the existing literature on ITS in programming education. Overall, the study contributes to the growing body of research on ITS by emphasizing the importance of tailoring ITS interventions to meet the unique needs of students with varying levels of programming experience.

## 6 Conclusion

This study investigated the effectiveness of Intelligent Tutoring Systems in supporting students with varying levels of programming experience. The findings from the mixed-methods research design provided valuable insights into the impact of ITS on student learning outcomes and the adaptability of ITS to different skill levels. As a result, best practices for utilizing ITS in heterogeneous programming classrooms are discussed below.

The quantitative analysis revealed significant differences in the influence of ITS on improving programming skills, the usefulness of ITS feedback on programming tasks, and satisfaction levels with the interface and functionality of the ITS based on students' programming experience levels. Advanced students showed the most significant improvement in programming skills and found the feedback more advantageous than beginner and intermediate students. Additionally, intermediate and advanced students reported higher satisfaction with the ITS interface and functionality.

The qualitative analysis highlighted three key themes: the positive impact of ITS on the learning process, challenges faced by students, and suggestions for improvement. Many participants reported that the ITS helped them understand topics and reduced anxiety, while others mentioned difficulty finding specific courses or limitations in the scope of learning. Suggestions for improvement included enhancing collaboration with Intelligent Agents and providing diverse learning resources.

### 6.1 Theoretical contributions

This study contributes to the literature by emphasizing the importance of tailoring ITS interventions to meet the unique needs of students with varying levels of programming experience. The findings suggest that individual student characteristics, such as programming proficiency levels, can significantly influence the effectiveness of ITS feedback and satisfaction levels in programming education.

### 6.2 Practical implications

The research findings indicate several best practices for utilizing Intelligent Tutoring Systems in heterogeneous programming classrooms.

- First, it is crucial to implement adaptive learning strategies within the ITS to personalize the learning experience for students with varying levels of programming proficiency. This could involve adjusting the difficulty level of tasks, providing tailored feedback, and offering individualized learning pathways to meet each student's unique needs.
- Additionally, incorporating a variety of learning resources and activities within the ITS, such as interactive tutorials, coding challenges, and peer collaboration opportunities, can cater to different learning styles and preferences among students.

- Furthermore, creating a supportive and collaborative learning environment by integrating Intelligent Agents within the ITS to provide timely feedback, guidance, and assistance can enhance students' engagement and motivation in learning programming concepts.
- Lastly, continuous monitoring and evaluation of the ITS's effectiveness, as well as seeking feedback from students to identify areas for improvement, can contribute to the ongoing optimization of the learning experience in heterogeneous programming classrooms.

Educators and instructional designers can use the insights from this study to optimize ITS interventions for students with diverse programming backgrounds. By addressing challenges and incorporating suggestions for improvement, educators can enhance students' learning experience and outcomes using ITS in programming education. Tailoring ITS to individual student needs can lead to more personalized and effective learning environments.

### 6.3 Limitations and prospective research opportunities

This study has several limitations that warrant discussion. First, one notable limitation is the potential for self-report bias in the student data collected. Future research should consider incorporating objective measures of programming performance alongside self-reported data to strengthen our findings' validity. Additionally, further exploration of how demographic factors, such as gender, influence the effectiveness of ITS could yield valuable insights into personalized learning approaches within programming education. Another critical factor is the varying levels of programming experience among participants. Beginners and advanced learners may benefit differently from the Blackbox ITS. These discrepancies highlight the need for tailored interventions that address individual experience levels to optimize learning outcomes effectively.

Moreover, the exclusive use of the Blackbox ITS in this study may restrict the generalizability of our findings. Future research could explore multiple ITSs, enabling comparative effectiveness analyses across different platforms to broaden understanding. Additionally, we must consider the potential variations in ITS effectiveness due to differences in user interfaces; varied designs can significantly impact student engagement and learning.

In light of these developments, we also acknowledge the emergence of Generative AI systems like Copilot and ChatGPT, which offer enhanced programming support through improved integration and assistance. This advancement indicates that programming ITSs are evolving, presenting significant opportunities for better programming education. Future research should examine how these tools can be effectively integrated within ITSs to enrich the learning experience and support diverse programming needs.

**Acknowledgements** The author(s) is(are) grateful to Erdem Kutluk and Fadi Imad Osman Abu Raid for their assistance in the data collection. The author(s) also thank(s) everyone, particularly students, who contributed to the writing of this paper through their participation in its survey.

**Author contributions** Conceptualization, AWFK; methodology, AWFK; software, AWFK; validation, AWFK; formal analysis, AWFK; investigation, AWFK; resources, AWFK; data curation, AWFK; writing—original draft preparation, AWFK; writing—review and editing, AWFK; visualization, AWFK; supervision, AWFK; project administration, AWFK; funding acquisition, AWFK.

**Funding** This research received no external funding.

**Data availability** The datasets generated during and/or analyzed during the current study are available from the corresponding author upon reasonable request.

### Declarations

**Ethics approval and consent to participate** While preparing this work, the author(s) used Grammarly AI to proofread and improve the manuscript's language. After using this tool/service, the author(s) reviewed and edited the content as needed and took full responsibility for the publication's content.

**Competing interests** The authors declare no competing interests.

**Open Access** This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If

material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

## References

1. Alkhatlan A, Kalita JK. Intelligent Tutoring systems: a comprehensive historical survey with recent developments. *ArXiv*, abs/1812.09628. 2018.
2. Šarić-Grgić I, Grubišić A, Stankov S, Stula M. An agent-based intelligent tutoring systems review. *Int J Learn Technol*. 2019;14:125–40.
3. Nesbit JC, Liu L, Liu Q, Adesope OO. Work in progress: intelligent tutoring systems in computer science and software engineering education. In 2015 ASEE Annual Conference & Exposition. 2015; pp. 26–1754.
4. Moraes LO, Pedreira CE. Designing an intelligent tutoring system across multiple classes. *CSEDM@EDM*. 2020.
5. Novickis L, Rikure T. Intelligent tutoring systems. In Proceedings of IST4Balt International Workshop "IST 6th Framework Programme–Great Opportunity for Cooperation & Collaboration. 2005; pp. 35–40.
6. Bradác V, Kostolanyova K. Intelligent tutoring systems. *J Intell Syst*. 2016;26:717–27.
7. Guang G. Research on intelligence-based tutoring system. *J Chin Inf Proc*. 2003.
8. Zhang L, Zeng Q, Chen Y. Research on the intelligent tutoring system based on teaching activities tree. 2010 International Conference on Computer Application and System Modeling (ICCASM 2010), 2010; 14: V14-534–V14-538.
9. Al-Jumeily D, Hussain AJ, Alghamdi MI, Lamb DJ, Hamdan H. The development of an intelligent tutorial system for system development. 2014 International Conference on Web and Open Access to Learning (ICWOAL), 2014;1–6.
10. Fan Z, Noller Y, Dandekar A, Roychoudhury A. Intelligent tutoring system: experience of linking software engineering and programming teaching. *ArXiv*, abs/2310.05472. 2023.
11. Crow T, Luxton-Reilly A, Wuensche B. Intelligent tutoring systems for programming education: a systematic review. In Proceedings of the 20th Australasian Computing Education Conference. 2018; pp. 53–62.
12. Butz CJ, Hua S, Maguire RB. A web-based intelligent tutoring system for computer programming. *IEEE/WIC/ACM International Conference on Web Intelligence (WI'04)*, 159–165. 2004.
13. Dadić T. Intelligent tutoring system for learning programming. In *Intelligent tutoring systems in e-learning environments: Design, implementation and evaluation* (pp. 166–186). IGI Global. 2011.
14. Khazanchi R, Khazanchi P. Artificial intelligence in education: A closer look into intelligent tutoring systems. In *Handbook of research on critical issues in special education for school rehabilitation practices* (pp. 256–277). IGI Global. 2021.
15. Conati C, Kardan S. Student modeling: supporting personalized instruction, from problem solving to exploratory open ended activities. *AI Mag*. 2013;34:13–26.
16. Hoppe L, Gembariski PC, Lachmayer R. Intelligent Tutoring System as a Tool of Formative Assessment in Design Education. *DS 110: Proceedings of the 23rd International Conference on Engineering and Product Design Education (EPDE 2021)*. 2021.
17. Epstein D, Pinho ID, Acosta OC, Reategui E. Inquiry-based learning environment using intelligent tutoring system. *IEEE Front Educ Conf (FIE)*. 2013;2013:1072–4.
18. Smith PJ, Soloway E, Carroll JM. Special session: intelligent tutoring and help systems. *Proc Hum Factors Ergon Soc Annu Meeting*. 1987;31:280–280.
19. Swanson T. The effectiveness of using intelligent tutoring systems to increase student achievement. 2019.
20. Sokolnicki T. Towards knowledge-based tutors: a survey and appraisal of Intelligent tutoring systems. *Knowl Eng Rev*. 1991;6:59–95.
21. Rokade PS, Jawandhiya D. Online intelligent tutoring system for improvising the performance of students. *J Emerg Technol Innov Res*. 2019.
22. Rathore AS, Arjaria SK. Intelligent tutoring system. In *Utilizing educational data mining techniques for improved learning: emerging research and opportunities* (pp. 121–144). IGI global. 2020.
23. Gavrilović N, Jovanović S. Implementing ITS (Intelligent Tutoring Systems) for Java programming e-learning. In *The Sixth International Conference on e-Learning (eLearning-2015)*. 2015; pp. 24–25.
24. Lee C, Baba MS. The intelligent web-based tutoring system using the C++ standard template library. *Malaysian Online J Instr Technol*. 2005;2(3):34–42.
25. Schez-Sobrinho S, Gmez-Portes C, Vallejo D, Glez-Morcillo C, Redondo MA. An intelligent tutoring system to facilitate the learning of programming through the usage of dynamic graphic visualizations. *Appl Sci*. 2020;10(4):1518.
26. Khakaj F, Aleven V. Towards improving introductory computer programming with an ITS for conceptual learning. In *Artificial Intelligence in Education: 19th International Conference, AIED 2018, London, UK, June 27–30, 2018, Proceedings, Part II 19* (pp. 535–538). Springer International Publishing. 2018.
27. Al-Shanfari L, Abdullah S, Fstnassi T, Al-Kharusi S. Instructors' perceptions of intelligent tutoring systems and their implications for studying computer programming in Omani higher education institutions. *Int J Membrane Sci Technol*. 2023;10(2):947–67.
28. Zhong X, Zhan Z. An intelligent tutoring system for programming education based on informative tutoring feedback: system development, algorithm design, and empirical study. *Interactive Technology and Smart Education*, (ahead-of-print). 2024.
29. Francisco R, Silva F. Intelligent tutoring system for computer science education and the use of artificial intelligence: a literature review. *International Conference on Computer Supported Education*. 2022.
30. Feng S, Magana AJ, Kao D. A systematic review of literature on the effectiveness of intelligent tutoring systems in STEM. *IEEE Front Educ Conf (FIE)*. 2021;2021:1–9.
31. Hieu BT. Analysis of the interoperability of intelligent tutoring systems for programming: an aspect of programming exercises. *Int J Sci Technol Res*. 2019;8:655–9.
32. Sweller J. Cognitive load theory, learning difficulty, and instructional design. *Learn Instr*. 1994;4(4):295–312.

33. Sweller J. Discussion of 'emerging topics in cognitive load research: using learner and information characteristics in the design of powerful learning environments.' *Appl Cogn Psychol*. 2006;20:353–7.
34. Renumol VG, Jayaprakash S, Janakiram D. Classification of Cognitive Difficulties of Students to Learn Computer Programming, Department of Computer Science, Indian Institute of Technology, Madras. India, Technical Report. 2009.
35. Yousoof M, Sapiyan M. Customized instructional pedagogy in learning programming-proposed model. *J Theor Appl Inf Technol*. 2016;85(3):309.
36. Lamia M, Mohamed H, Samira A. An Intelligent Tutoring System within a Social Network (IT2SN) for logical problem solvin. In *Proceedings of the 5th Multidisciplinary International Social Networks Conference*. 2018; pp. 1–6.
37. Müller S, Bergande B, Brune P. Robot tutoring: On the feasibility of using cognitive systems as tutors in introductory programming education: a teaching experiment. In *Proceedings of the 3rd European Conference of Software Engineering Education*. 2018; pp. 45–49.
38. Orhun E. Knowledge representation in intelligent tutoring systems for computer programming. *Am J Math Manag Sci*. 1991;9:243–59.
39. Yousoof M, Sapiyan M, Kamaluddin K. Measuring cognitive load- a solution to ease learning of programming. *World Acad Sci Eng Technol Int J Soc Behav Educ Econ Bus Ind Eng*. 2007;1:32–5.
40. William FKA. Crafting a strong research design: a step-by-step journey in academic writing. *Int J Sci Res Manag*. 2024;12(3):3238–45.
41. Creswell JW. *Research design: qualitative, quantitative, and mixed methods approaches*. 4th ed. United States of America: SAGE Publications Inc.; 2014.
42. William FKA. My data are ready, how do I analyze them: navigating data analysis in social science research. *Int J Sci Res Manag*. 2024;12(3):1730–41.

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.