

Enhancing Computational Thinking and Problem-Solving in Programming Education Through Generative AI: A Scoped Review

Courage Matobobo
Business and Application Development
Walter Sisulu University
East London, South Africa
0000-0002-7125-5989

Prince Daughin Ngqabutho Neube
Networking and IT Support
Walter Sisulu University
East London, South Africa
0000-0002-9262-3779

Nomputumo Linah Ngesimani
Business and Application Development
Walter Sisulu University
East London, South Africa
0000-0002-8385-1243

Godwin Pedzisai Dzvapatsva
School of Technology Business and
Arts
University of Suffolk
Ipswich, United Kingdom
0000-0001-5534-2647

Edmore Chinhamo
Business School
University of Greenwich
London, United Kingdom
0000-0003-4684-7517

Abstract—This study assesses how generative artificial intelligence tools enhance computational thinking and problem-solving skills in the context of programming education. Generative AI (GenAI) has ushered in a new era in programming education, offering immediate, personalised support through tools like ChatGPT and GitHub Copilot. Although generative AI tools have shown promise in enhancing immediate problem-solving abilities, there is a lack of research on their long-term effects on students' computational thinking and professional programming skills development. This study conducted a scoped evaluation of previously published papers using the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) standard, and the data was analysed using a thematic approach. The findings from the study indicate that integrating GenAI can potentially enhance computational thinking and problem-solving for programming students. On the flipside, our study highlighted some significant ethical challenges associated with using GenAI in academia, particularly regarding issues of originality in student work. Contrary to expectations on how GenAI tools enhance learners' decomposition, abstraction, and algorithm design skills, most of the findings concentrated on students' completion of tasks. From a practical perspective, it is evident that GenAI has changed the learning landscape therefore, there is a need from a policy perspective to start thinking about the transformational roles of educators. Future studies should be carried out over a long period and should start by assessing students' levels of problem-solving at a particular age before the immersive use of GenAI and then check the results after the use of these tools.

Keywords—*Computational Thinking, Generative Artificial Intelligence, Programming Education, Problem Solving Skills, ChatGPT*

I. INTRODUCTION

Programming education equips students with problem-solving skills and computational thinking, both essential in today's technology-driven world [1]. However, students often struggle with complex programming concepts and syntax, leading to frustration and disengagement [2]. Traditional teaching approaches sometimes fail to provide personalised support to foster meaningful learning experiences [3]. These challenges highlight the need for innovative pedagogical tools to enhance programming education and better engage students.

Generative Artificial Intelligence (AI) is emerging as a transformative tool in this context. Tools like OpenAI, Codex,

and GitHub Copilot provide real-time code suggestions, automate debugging, and offer context-aware programming solutions. By reducing cognitive load and providing immediate feedback, these tools encourage students to experiment with different approaches, deepening their problem-solving abilities and enhancing their computational thinking skills [4]. Furthermore, the adaptive nature of these AI-driven platforms enables personalised learning, offering tailored guidance that caters to individual students' needs [5]. This personalisation fosters technical proficiency and the development of critical thinking, which is vital for success in higher education and beyond.

Although generative AI tools have shown promise in enhancing immediate problem-solving abilities, their long-term effects on students' computational thinking and professional programming skills development remain underexplored. While the potential of these tools is significant, their integration into programming education also raises critical challenges [6]. While the potential of generative AI tools is significant, their use in programming education also raises critical challenges [6]. It's not just about the benefits but also the responsible use of these tools. There is a concern that students may become overly dependent on AI-generated solutions, which could limit the development of independent problem-solving skills [7]. Additionally, ethical considerations, such as plagiarism, algorithmic bias, and data privacy, must be addressed to ensure that AI tools are used responsibly. Balancing the benefits of AI assistance with the need for students to engage deeply with programming problems on their own is not just essential, but it is a moral imperative for educators aiming to integrate these tools effectively [8], [9], [10].

Given the potential and challenges of generative AI in programming education, this study aims to explore the role these tools play in enhancing students' problem-solving abilities and computational thinking. Specifically, the research focuses on how generative AI tools are used in higher education to help students overcome programming challenges. The primary research question guiding this study is:

How do Generative AI tools assist students in solving programming problems in higher education contexts?

This scoped review synthesises findings from the high-quality Scopus database to comprehensively understand

current practices and trends. The study offers practical insights for educators, policymakers, and curriculum developers by identifying the benefits and challenges of integrating generative AI into programming courses. The study offers practical insights for educators, policymakers, and curriculum developers by identifying the benefits and challenges of integrating generative AI into programming courses needless to say, software houses producing IDE have already started incorporating AI into their programs. Although the review may not cover all existing tools due to the field's rapid growth, it focuses on the most influential examples to provide actionable recommendations for the future of programming education.

The findings from this study contribute valuable perspectives on integrating generative AI tools in programming education, helping inform teaching practices and curriculum design. By addressing the potential benefits and limitations, this research aims to provide a balanced understanding of how these tools can be leveraged to foster computational thinking and problem-solving in students, shaping the next generation of programmers and problem-solvers.

II. LITERATURE REVIEW

A. Computational Thinking and Problem-Solving in Programming Education

Computational thinking (CT) is foundational to programming education, equipping students with critical skills such as decomposition, abstraction, and algorithm design. These skills are essential for managing complex computational challenges and enable students to engage in structured, strategic problem-solving [11]. As programming tasks become more complex, CT has become increasingly relevant, serving as a vital framework supporting both introductory and advanced programming [12]. People should, however, be cautious about CT following structure, which can hamper creativity. Many early programmers developed languages like Assembly Language, LISP, FORTRAN, C, and MATLAB, on which most of the latest technological developments are based, all without any traces of CT. Exhaustive application of CT might not equate to the ability to solve real-life scenarios as they are not structured. While acknowledging this rebuttal, research illustrates how interactive learning tools rooted in CT principles improve students' programming abilities, fostering algorithmic thinking and translating abstract concepts into concrete learning experiences [13]. Educational games, for instance, have shown promise in making CT accessible to students by breaking programming problems into manageable, step-by-step tasks [14].

B. Generative AI's Unique Contributions to Programming Education

Generative AI (GenAI) has ushered in a new era in programming education, offering immediate, personalised support through tools like ChatGPT and GitHub Copilot. Unlike traditional AI applications that provide feedback and guidance, GenAI actively generates code, suggests solutions, and offers step-by-step support. This unique capability allows students to shift their focus from syntax to problem-solving, a critical shift for developing higher-order thinking skills in programming [15]. The multifaceted role of GenAI tools in programming education is evident, as they provide distinct support that other educational tools cannot fully replicate,

particularly in assisting with complex, iterative programming tasks. These tools facilitate higher-order thinking by allowing students to tackle more complex programming challenges, making them valuable resources for fostering algorithmic and reflective thinking in programming education [15].

The use of GenAI tools in programming is particularly impactful due to their unique ability to scaffold complex programming concepts for students at various levels of expertise. A study [16] suggests that GenAI has been shown to help students overcome code comprehension challenges in introductory programming, providing contextual guidance that strengthens their understanding of intricate programming concepts. The real-time, tailored support from GenAI tools like GitHub Copilot and ChatGPT empowers students to explore programming structures and logic with an added support layer, fostering more profound engagement with the problem-solving process. The downside, however, is that GenAI rarely offers adaptive scaffolding, which can result in cognitive overload depending on the stage at which a learner is.

Generative AI supports critical thinking and analysis in structured educational settings by facilitating rapid prototyping and code generation tasks. This isn't just about making tasks easier but about fostering a higher level of engagement with CT principles. The real value of GenAI lies in its ability to enable students to concentrate on refining and critically assessing their solutions, promoting a higher level of engagement with CT principles. A study in [17] introduces the AI-Lab framework to integrate GenAI in programming courses, emphasising how GenAI-driven debugging and feedback promote active learning and self-directed problem-solving. Such active learning environments encourage students to explore programming concepts more thoroughly, enhancing their computational thinking and technical proficiency.

A study in [18] describes GenAI as a "supportive companion" in programming education, providing a non-judgmental space where students can experiment and develop creativity alongside essential CT skills. GenAI's non-directive assistance fosters a sense of independence while helping students build ethical considerations and critical thinking skills within a safe, exploratory environment. However, engagement with GenAI is not uniform across all student populations. A study in [19] highlights accessibility and cultural perceptions as factors influencing student engagement with GenAI, underscoring the need for supportive and inclusive programming environments where students can use GenAI as a collaborative partner in problem-solving.

C. Challenges of Generative AI Tools in Programming Education

GenAI tools also present significant challenges within programming education despite their many advantages. A primary concern is the potential for over-reliance, which could inhibit students' ability to develop independent computational thinking skills essential for programming. As GenAI tools provide instant solutions, students may become less inclined to engage in the iterative problem-solving processes required to build a strong foundation in CT. A study in [20] cautioned that excessive dependence on GenAI might prevent students from engaging with the complexities of programming tasks, potentially hindering the development of the problem-solving skills they will need in real-world scenarios. A study in [21]

adds that students who rely heavily on GenAI may adopt a passive approach to learning, which can limit their ability to troubleshoot and debug independently, key skills for programming proficiency.

Researchers emphasise balancing GenAI-assisted tasks with manual coding and problem-solving exercises to mitigate these challenges. A study in [22] proposes integrating “no-GenAI” assignments to encourage independent problem-solving while incorporating GenAI-guided tasks that reinforce critical thinking without reducing CT skills. Such a balanced curriculum can help students develop resilience and computational thinking independence while benefiting from GenAI’s capabilities.

Another challenge lies in the ethical and accessibility considerations associated with GenAI use in education. While tools like ChatGPT and GitHub Copilot provide significant support, they raise concerns about data privacy, equitable access, and potential biases in AI-generated outputs. Existing literature highlights the need to address these ethical issues to ensure that GenAI fosters a responsible and inclusive learning environment [23]. Without careful consideration of these challenges, integrating GenAI in programming education may inadvertently disadvantage certain student groups or promote biased learning outcomes. There is also a potential drawback: it can provide misleading solutions when used by novices in a field who cannot tell whether the provided output is incorrect, especially if the Zone of Actual Development (ZAD) is blurred.

D. How GenAI Tools Help Students in Practice

The use of Generative AI (GenAI) by students in practice has revolutionized the application of knowledge and skill development in both academic and professional contexts. By offering personalized learning, GenAI tools empower students to take control of their learning journey, generating tailored study materials, adaptive exercises, and language support that cater to their individual needs and learning styles. Recent studies highlight that AI-powered adaptive learning technologies can increase student performance and engagement by up to 30%, particularly for language acquisition and STEM subjects [24], [25].

These tools foster skill development through practical simulations, coding assistance, and role-play scenarios. For example, AI-powered medical simulations are being used to train healthcare students in diagnostics and surgical procedures, providing immersive, risk-free environments for practice [26]. Similarly, platforms like Codex or GitHub Copilot assist students in learning programming by offering instant code generation and debugging tips, significantly reducing the learning curve for new coders [27].

Furthermore, GenAI enhances creative problem-solving by providing brainstorming support, enabling the development of prototypes, and gamifying learning activities to encourage critical thinking and innovation. AI-driven gamification strategies have been shown to enhance creative engagement, with students reporting a 40% improvement in motivation when such tools are integrated into their curricula [28]. Prototyping tools that utilize AI have also become invaluable for design students, enabling rapid ideation and iteration of projects [29].

In collaborative settings, AI facilitates teamwork by organizing tasks, co-authoring documents, and delivering

constructive feedback to improve project outcomes. Tools such as Microsoft Teams and Google Workspace now leverage AI features to foster a sense of community and shared learning experience, reduce redundancy, and increase productivity in group projects [30]. Collaborative writing platforms with real-time AI feedback further streamlined academic and professional teamwork, ensuring polished and cohesive outcomes [31].

While the benefits of GenAI are substantial, it is crucial to consider the ethical implications of its use. Over-reliance on AI can hinder student autonomy, and issues such as privacy concerns and algorithmic bias must be addressed. This is essential to ensure these technologies' equitable and responsible deployment [32].

By integrating GenAI into their practice, students not only deepen their understanding of theoretical concepts but also develop essential skills in adaptability, collaboration, and innovation, preparing them for future challenges in an increasingly tech-driven world.

E. Theoretical Framework

We have adopted the scaffolding theory, a sub-theory of Lev Vygotsky’s constructivist approach, as foundational in using GenAI to solve complex problems. Scaffolding is “a way of operationalising Vygotsky’s 1987 concept of working in the zone of proximal development” [33]. Study [33] cited in [34] identified three important aspects of scaffolding as:

- i) co-construction of knowledge in a discourse that is fundamentally dialogic.
- ii) the significance of the kind of activity in which knowledge is embedded and,
- iii) the role of artifacts that mediate knowing

For the purposes of this study, we support the role of GenAI for interactive scaffolding, allowing students to work within their Zone of Proximal Development [34], [35]. The student needs to delineate when to work independently from GenAI less the Zone of No Development is extended. GenAI allows dialogue with the learner to enhance computational thinking and problem-solving as he/she attempts to complete the task. GenAI as an artifact will propel the student in the Zone of Proximal Development with the potential of reducing support [36], [37] as students will have developed the needed computational thinking and problem-solving skills. We will use the scaffolding lens to view our results and link them with the viewpoint of the study discussion.

III. METHODOLOGY

This study adopted a scoped literature review to explore the role of generative AI tools in supporting the solving of complex programming problems. The authors searched the Scopus database to identify articles that met the inclusion criteria. The Scopus database was used because it offers high-quality literature that is suitable and adequate for addressing the research question for this study. To find previously published publications, the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) standard was followed [38]. The following inclusion and exclusion criteria guided the study:

Inclusion Criteria

- Studies on generative AI applications in education

- Studies focusing on computer programming
- Studies written in English
- Empirical studies

Exclusion Criteria

- Studies that do not focus on programming education
- Studies discussing traditional or non-generative AI applications
- Studies written in other languages except English
- Non-empirical studies

Search string used for the Scopus database:

Using "Generative AI" OR "generative artificial intelligence" AND solving AND programming

This search string retrieved articles about generative AI in teaching programming. The authors did not include specific keywords like computational thinking and problem-solving skills to broaden the search. This is because some articles may not explicitly mention these keywords, but the concepts are implicitly discussed when you go through them.

The search for articles was done on the 5th of November 2024, and 46 articles were retrieved from the initial search. A preliminary analysis of the title and abstract was done, and 22 articles were excluded because they fitted the exclusion criteria. A detailed analysis was conducted, and 4 articles were excluded after full analysis of the article. Only 20 articles met the inclusion criteria and were analysed in this study.

Figure 1 illustrates the PRISMA flowchart diagram used to conduct a scoped review on enhancing computational thinking and problem-solving in programming education through generative AI.

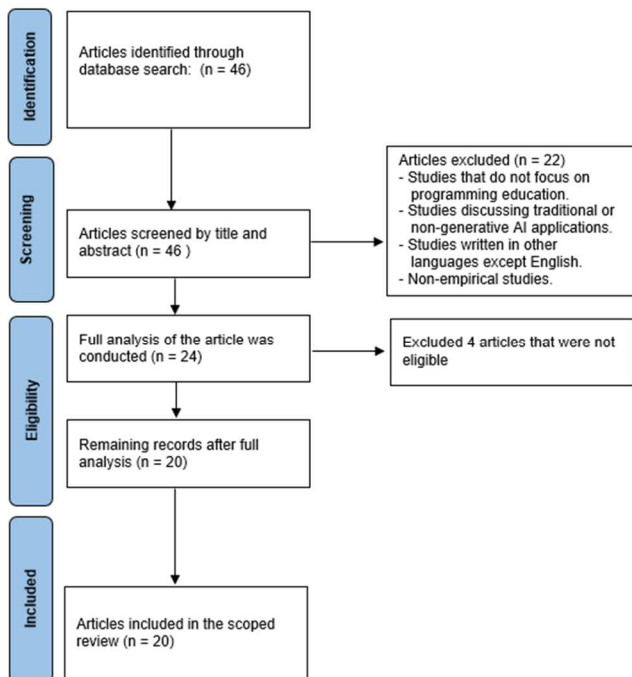


Fig. 1. PRISMA flowchart diagram for included studies

Data Analysis

The articles included in the scoping review were imported into ATLAS.ti V9.0, a Computer Assisted Qualitative Data Analysis software (CAQDAS) [39]. The six steps of thematic analysis listed below guided the data analysis [40]:

- Step 1: Getting acquainted with data, comprehending it and noting down initial ideas.
- Step 2: Developing initial codes by coding interesting points and merging data relevant to each code.
- Step 3: Looking for themes by merging codes and collecting data relevant to each potential theme.
- Step 4: Revising themes by checking their suitability for the coded extracts.
- Step 5: Describing and naming themes by generating distinct meaning and names for each theme as well as continuing to refine and analyse each one.
- Step 6: Creating the report by selecting concise, convincing examples of extracts, analysing them in detail, connecting them to the literature and producing an academic report on the analysis.

IV. RESULTS

This section presents the findings of the study. Data analysis led to 62 initial codes from the included articles in the scoped review. Subsequently, similar codes were merged and transferred to a new code that formed code clusters resulting in the decrease of the initial codes from 62 to 29. The code clusters were then transferred to the appropriate code groups where they underwent further refinement and ultimately emerged as four themes. The findings are presented in the following themes:

- Generative AI tools
- Usage of Generative AI tools
- Benefits of Generative AI tools
- Challenges of Generative AI tools

A. Generative AI Tools

Generative AI tools have proved helpful in programming education. The findings show that students use generative AI tools such as automatic program repair (APR), ChatGPT, Iris, and Copilot. The findings show that students use generative AI tools to generate, repair, and evaluate codes [8], [41], [42]. The findings show that students use tools like Automatic program repair (APR) to assist with debugging tasks [42]. APR helps students achieve the correct output of their codes, thereby making ARP a supportive mechanism in programming when students such as beginners are overwhelmed. A study in [42] shows the use of Contextual error model repair (CEMR). CEMR is an automated technique students use in the introductory programming course to repair incorrect student programming codes [42]. This technique corrected mistakes in students' code before they fully understood the solution, allowing them to learn from the corrected model.

Several studies have highlighted the use of ChatGPT in programming education [41], [43]. A POLITEHNICA University of Bucharest (UPB) study showed that students used ChatGPT to solve their programming homework [43]. A

class observation in a study in [44] showed that students found that using ChatGPT made assignments engaging and rewarding. A study in [43] showed that ChatGPT improved students' experience compared to Copilot due to challenges in comprehending, modifying, and debugging code snippets, which impacted the overall effectiveness in solving tasks.

Besides ChatGPT, Iris was used to support computer science students. Iris acts as a tutor who guides students in their programming exercises by providing calibrated guidance that encourages independent thinking, offering hints or counter-questions instead of complete solutions [8]. In a study in [8], students perceive Iris as an effective tool in programming exercises and homework because it provides immediate and personalised support, understands their questions, provides relevant support, and contributes to the learning process. Furthermore, Iris enabled students to ask questions without feeling judged by peers.

Generative AI tools transform self-directed learning (SDL) by enabling new interventions to enhance self-regulated learning strategies [45]. As computational thinking (CT) becomes vital in education, SDL is recognized as a key method for developing CT skills [46]. Generative AI-powered conversational agents are particularly impactful, creating new possibilities for SDL and successfully supporting the development of CT.

In a study in [47], the survey findings revealed that most students ($n=79\%$) felt confident programming with the generative AI tools, and a majority ($n=59\%$) found it helpful for learning. The study highlighted some expressed concerns about over-reliance and a weaker grasp of fundamentals, and the study recommended delaying introducing generative AI tools to students first to develop fundamental programming skills [47].

B. Usage of Generative AI Tools

Generative AI tools such as ChatGPT and PyTutor offer valuable support in education by providing continuous guidance, problem-solving hints, and explanations that assist learners in understanding complex subjects. Educators must equip students with generative AI tools to improve their critical thinking skills and deepen the foundational skills necessary for their academic journey to succeed [17]. However, research suggests that there may be a risk of students being overly dependent on these tools, especially those with limited prior knowledge, which might hinder their ability to solve problems on their own [48]. Educators should guide students carefully to avoid overusing generative AI tools, which has challenges, such as students struggling with advanced concepts covered in programming courses due to over-reliance [17]. This over-reliance might cause students to grasp fundamental ideas at a surface level rather than interact with the content more deeply [43]. Generative AI tools must be used as a teaching tool rather than a replacement for skill development to reduce these hazards [43]. As students rely more on AI-generated code snippets, programming courses need to be reevaluated with an emphasis on new abilities [49]. To improve evaluations and guarantee that students retain and advance core abilities, educators are urged to proactively employ AI technologies in problem-solving activities [50].

C. Benefits of Generative AI Tools

Generative AI tools have proved to be beneficial in education. The findings show a growing use and transformative power in education and other media,

marketing, and healthcare sectors [9]. Because of the transformations that come with these tools, the education sector must think of new pedagogies and assessment methods to accommodate these transformations [9]. Generative AI tools have enhanced human capabilities that assist students in undertaking long projects as they help programming students understand, create, review, and debug programs [43], [46], [51], [43]. Generative AI tools, especially ChatGPT, have interactive engagements that attract students to interact with the tool while working on their various homework assignments, such as essays, argumentation, and coding [43], [51]. Generative AI tools like PyTutor provide ongoing guidance, hints, pseudocode, code solutions, and code explanations to students learning Python [48]. There is reported efficiency that comes with generative tools in programming education, significantly boosting task speed, with students completing tasks nearly three times faster on average [52]. Besides speed, students use generative tools to practice and refine their code, improving their critical thinking abilities [51]. Novice programming students can generate fairly accurate code by interacting with these tools, potentially bypassing self-regulation strategies [45]. Generative AI tools could help programming students detect bugs and explain the code and concepts [50], [51]. The effective integration of generative AI in programming has promoted equity in the education landscape by offering personalised learning experiences to students and the ability to access assistance 24 hours [17], [44]. With generative AI tools, students can interact with these tools as if interacting with another human being [51]. Generative AI tools are effective for beginners, helping to bridge knowledge gaps and enhance programming skills, mainly when used in moderation [48]. The findings show that generative AI tools are helpful if they are used in a moderated and responsible way.

D. Challenges of Generative AI Tools

Generative AI tools such as ChatGPT were seen to lack a mechanism for verification of the output and a deep understanding of the context, which can be a challenge to students in programming education [51]. Generative AI tools can distract students from deep understanding and critical thinking as the tools provide them with programming solutions [51], [52]. This can disturb the development of programming students' critical thinking and problem-solving skills [52]. Generative AI tools can mislead students since the outcomes of these tools are not always correct [43]. Most of these generative AI tools lack sources of references, have limited mathematical understanding, have potential biases, and have occasional inaccuracies that limit their reliability, making them less effective primary learning tools [51]. There are ethical concerns around the use of generative AI in education, such as data privacy, trust, and lack of regulations to ensure responsible and effective use [8], [9]. Students who over-rely on generative tools during their foundational courses tend to struggle with advanced topics, showing a gap that can be created by irresponsible use of these generative AI tools [17]. There is, therefore, a need to balance the use of generative AI tools in programming education to foster a deep understanding of fundamental concepts rather than a superficial understanding. Generative AI tools may not be suitable for logic errors faced by programming students [50].

V. DISCUSSIONS

This study aimed to assess whether Generative AI tools assist students in solving programming problems in higher

education contexts with a particular focus on computational thinking and problem-solving. Our findings from a scoped review revealed that GenAI tools significantly support students in enhancing computational thinking and problem-solving, which are cornerstones of programming. As highlighted in our literature, our findings highlighted how students can use GenAI tools to tackle programming concepts. Informed by our scaffolding theory, GenAI can assist students in getting started and support complex tasks. These findings are consistent with study in [42] which identified the power of GenAI in debugging tasks. This contrasts with old or built-in debugging tools that didn't provide scaffolding and led students into a "Zone of Confusion", where they potentially get further confused with the suggestions. The debugger is not dialogic, and the scaffolding is not adaptive.

Contrary to expectations on how GenAI tools enhance learners' decomposition, abstraction, and algorithm design skills, most of the findings concentrated on students' completion of tasks. Perhaps some of this discrepancy is due to differences in wording and how certain aspects like decomposition are addressed during this dialogic discussion. Our scaffolding theory, however, corroborates the finding on how dialogic GenAI tools are because of students' use of them to solve problems [51] and the immediacy of feedback.

Our study highlights significant ethical challenges associated with using GenAI in academia, particularly regarding issues of originality in student work. Literature and previous findings agree that institutions struggle to assess students using these tools, suggesting that academia may lag in adapting assessment methods to this new reality. This delay may stem from a reluctance to change established practices. Neither our literature review nor our findings suggest that transforming the educator's role is a viable solution to address the changing programming landscape dominated by GenAI tools. Moreover, we anticipated seeing educators advocating the use of GenAI tools and giving direction on usage to accommodate students with differing computational thinking and problem-solving abilities in our findings, which we did not. Furthermore, educators may use GenAI tools behind the scenes while presenting a facade of not doing so, creating a disconnect between academic attitudes and actual usage. Given these issues, a study investigating how educators, especially those teaching programming, use GenAI could offer valuable insights into the broader adoption of these tools in teaching.

Further, as expected, just as highlighted in the literature [7], [8], [9], the findings highlighted overreliance on tools as one of the flip sides of GenAI without learners furthering their problem-solving skills. Undeniably, these findings are valid from our study. However, we cannot underestimate the boost and motivation that the tools have for students. While problem-solving does not always call for a structured approach as GenAI offers, it can provide a clear framework to tackle complex issues where a student doesn't have a starting point. We strongly believe that applying to real-life problems or programming challenges can easily be accomplished with a base framework for solving problems.

From a practical perspective, it is evident that GenAI has changed the learning landscape; therefore, there is a need from a policy perspective to start thinking about the transformational roles of educators. As discussed in our literature section and findings, software houses have already incorporated these tools into the programming integrated

development interfaces, popular ones being Pycharm and Visual Studio code. In such cases, there is a need to revisit the policy and not only to balance the use of GenAI. Another possible consideration is approaching the software powerhouses to develop GenAI tools, which do not start by providing the whole answer on a silver plate but by providing developmental scaffolding. We speculate that GenAI developers are less focused on refining their tools to suit differential cognitive levels, possibly due to a primary emphasis on creating broadly applicable solutions that appeal to a wide range of users rather than addressing the nuanced needs of learners with varying levels of prior knowledge and problem-solving abilities. Regardless, when these students get hired, the work environment allows them to use any tool to solve a problem. A key drive we should also focus on is having learning environments as simulated workplace environments. This would allow students to practise their skills in a real-world setting before entering the workforce. It would also allow them to develop a deeper understanding of the technologies they will be working with. Another aspect that we did not find was the adaptive level of support. GenAI does not have metrics to gauge the existing level of competence. It remains unclear whether

While our study provides valuable insights into the potential enhancement of computational thinking and problem-solving using GenAI, we must acknowledge some associated limitations. Our first limitation is methodological limiting. Perhaps a longitudinal empirical study for programming students would add more credibility to our findings. Using a scoped review makes it difficult to ascertain the level of scaffolding that helped students move to a Zone of Proximal Development.

While there are noted limitations, our study took some steps to minimise their impact and enhance their validity. Using scoped literature allowed us to get co-occurrence of themes from 20 very recent articles and within the discipline understudy. Our findings provided a foundation for computational thinking and GenAI discussions in programming students. Our findings indicate the practical usefulness of GenAI tools in enhancing computational thinking.

VI. CONCLUSIONS

In summary, integrating generative AI into programming education offers substantial promise for enhancing computational thinking and problem-solving abilities. Like any other tool, our study has sufficiently highlighted the potential of GenAI while acknowledging some of the flip side of these tools. For instance, issues of over-reliance on tools that limit creativity and a structured approach to problem-solving are a one-size-fits-all, yet in real life, problems are unique. Our findings have highlighted a gap in policy, which is acting as an inhibitor to the usage of these tools.

Future studies should be carried out over a long period and should start by assessing students' levels of problem-solving at a particular age before the immersive use of GenAI and then check the results after the use of these tools. Some metrics on computational thinking would form the basis of such a proposed study.

REFERENCES

- [1] S. Yeni, J. Nijenhuis-Voogt, M. Saeli, E. Barendsen, and F. Hermans, "Computational thinking integrated

- in school subjects – A cross-case analysis of students' experiences," *Int J Child Comput Interact*, vol. 42, Dec. 2024, doi: 10.1016/j.ijcci.2024.100696.
- [2] M. Moresi, M. J. Gómez, and L. Benotti, "Predicting Students' Difficulties from a Piece of Code," *IEEE Transactions on Learning Technologies*, vol. 14, no. 3, pp. 386–399, Jun. 2021, doi: 10.1109/TLT.2021.3092998.
 - [3] X. Huang and C. Qiao, "Enhancing Computational Thinking Skills Through Artificial Intelligence Education at a STEAM High School," *Sci Educ (Dordr)*, vol. 33, no. 2, pp. 383–403, Apr. 2024, doi: 10.1007/s11191-022-00392-6.
 - [4] P. Tarau, "On Teaching Logic Programming in the Era of Generative AI," in *ICLP 2024*, 2024. [Online]. Available: <http://ceur-ws.org>
 - [5] V. Klassen and U. Häsel-Weide, "PerspeCTivO–Design-based research in perspective taking and computational thinking by using the robot Ozobot," in *Organising Committee*, Hans-Georg Weigand Publisher, 2024. [Online]. Available: <https://www.dm.uniba.it/meda4>
 - [6] I. Pesovski, R. Santos, R. Henriques, and V. Trajkovik, "Generative AI for Customizable Learning Experiences," *Sustainability (Switzerland)*, vol. 16, no. 7, Apr. 2024, doi: 10.3390/su16073034.
 - [7] R. Yilmaz and F. G. K. Yilmaz, "Augmented intelligence in programming learning: Examining student views on the use of ChatGPT for programming learning," *Computers in Human Behavior: Artificial Humans*, vol. 1, no. 2, p. 100005, Aug. 2023, doi: 10.1016/j.chbah.2023.100005.
 - [8] P. Bassner, E. Frankford, and S. Krusche, "Iris: An AI-Driven Virtual Tutor for Computer Science Education," in *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, Association for Computing Machinery, Jul. 2024, pp. 394–400. doi: 10.1145/3649217.3653543.
 - [9] N. El-Khalili, M. Al-Nashashibi, and S. Al-E'mari, "The Impact of Generative Artificial Intelligence on Learning: A Case Study at the University of Petra," in *2nd International Conference on Cyber Resilience, ICCR 2024*, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.1109/ICCR61006.2024.10532824.
 - [10] M. Al-kfairy, D. Mustafa, N. Kshetri, M. Insiew, and O. Alfandi, "Ethical Challenges and Solutions of Generative AI: An Interdisciplinary Perspective," *Informatics*, vol. 11, no. 3, p. 58, Aug. 2024, doi: 10.3390/informatics11030058.
 - [11] G. Falloon, "Advancing young students' computational thinking: An investigation of structured curriculum in early years primary schooling," *Comput Educ*, vol. 216, Jul. 2024, doi: 10.1016/j.compedu.2024.105045.
 - [12] T. Hurt *et al.*, "The computational thinking for science (CT-S) framework: operationalizing CT-S for K–12 science education researchers and educators," Dec. 01, 2023, *Springer Science and Business Media Deutschland GmbH*. doi: 10.1186/s40594-022-00391-7.
 - [13] S. A. Triantafyllou, T. Sapounidis, and Y. Farhaoui, "Gamification and Computational Thinking in Education: A systematic literature review," 2024, *Editorial Salud, Ciencia y Tecnologia*. doi: 10.56294/sctconf2024659.
 - [14] S. Zhang, G. K. W. Wong, and P. C. F. Chan, "Playing coding games to learn computational thinking: What motivates students to use this tool at home?," *Educ Inf Technol (Dordr)*, vol. 28, no. 1, pp. 193–216, Jan. 2023, doi: 10.1007/s10639-022-11181-7.
 - [15] H. Nold, "Using Critical Thinking Teaching Methods to Increase Student Success: An Action Research Project," *International Journal of Teaching*, vol. 29, no. 1, pp. 17–32, 2017, [Online]. Available: <http://www.isetl.org/ijtlhe/>
 - [16] L. Roest, H. Keuning, and J. Jeuring, "Next-Step Hint Generation for Introductory Programming Using Large Language Models," Dec. 2023, [Online]. Available: <http://arxiv.org/abs/2312.10055>
 - [17] E. Dickey, A. Bejarano, and C. Garg, "AI-Lab: A Framework for Introducing Generative Artificial Intelligence Tools in Computer Programming Courses," *SN Comput Sci*, vol. 5, no. 6, Aug. 2024, doi: 10.1007/s42979-024-03074-y.
 - [18] W. Yan, T. Nakajima, and R. Sawada, "Benefits and Challenges of Collaboration between Students and Conversational Generative Artificial Intelligence in Programming Learning: An Empirical Case Study," *Educ Sci (Basel)*, vol. 14, no. 4, Apr. 2024, doi: 10.3390/educsci14040433.
 - [19] Zimmermann M, Janetzko H, and Haymond B, "Integrating Generative AI Methods in Computer Science Education: Perspectives, Strategies, and Outcomes," in *EDULEARN24 Proceedings*, 2024, pp. 10358–10365.
 - [20] E. Kasneci *et al.*, "ChatGPT for good? On opportunities and challenges of large language models for education," Apr. 01, 2023, *Elsevier Ltd*. doi: 10.1016/j.lindif.2023.102274.
 - [21] F. Fui-Hoon Nah, R. Zheng, J. Cai, K. Siau, and L. Chen, "Generative AI and ChatGPT: Applications, challenges, and AI-human collaboration," 2023, *Routledge*. doi: 10.1080/15228053.2023.2233814.
 - [22] E. Dickey, A. Bejarano, and C. Garg, "AI-Lab: A Framework for Introducing Generative Artificial Intelligence Tools in Computer Programming Courses," *SN Comput Sci*, vol. 5, no. 6, Aug. 2024, doi: 10.1007/s42979-024-03074-y.
 - [23] F. Rashid, N. Duong-Trung, and N. Pinkwart, "Generative AI in Education: Technical Foundations, Applications, and Challenges," 200AD. [Online]. Available: www.intechopen.com
 - [24] S. Wang, F. Wang, Z. Zhu, J. Wang, T. Tran, and Z. Du, "Artificial intelligence in education: A systematic literature review," *Expert Syst Appl*, vol. 252, Oct. 2024, doi: 10.1016/j.eswa.2024.124167.
 - [25] I. Gligorea, M. Cioca, R. Oancea, A. T. Gorski, H. Gorski, and P. Tudorache, "Adaptive Learning Using Artificial Intelligence in e-Learning: A Literature Review," *Educ Sci (Basel)*, vol. 13, no. 12, Dec. 2023, doi: 10.3390/educsci13121216.

- [26] E. W. Riddle, D. Kewalramani, M. Narayan, and D. B. Jones, "Surgical Simulation: Virtual Reality to Artificial Intelligence," *Curr Probl Surg*, vol. 61, no. 11, p. 101625, Nov. 2024, doi: 10.1016/j.cpsurg.2024.101625.
- [27] F. Eshraghian, N. Hafezieh, F. Farivar, and S. de Cesare, "AI in Software Programming: Understanding Emotional Responses to GitHub Copilot," *Information Technology & People*, 2024, doi: 10.1108/ITP-01.
- [28] R. Huang *et al.*, "The impact of gamification in educational settings on student learning outcomes: a meta-analysis," *Educational Technology Research and Development*, vol. 68, no. 4, pp. 1875–1901, Aug. 2020, doi: 10.1007/s11423-020-09807-z.
- [29] N. Y. Motlagh, M. Khajavi, A. Sharifi, and M. Ahmadi, "The Impact of Artificial Intelligence on the Evolution of Digital Education: A Comparative Study of OpenAI Text Generation Tools including ChatGPT, Bing Chat, Bard, and Ernie," *arXiv preprint arXiv:2309.02029*, 2023.
- [30] A. Zitar, S. I. Ali, and N. Islam, "Worker and workplace Artificial Intelligence (AI) coexistence: Emerging themes and research agenda," *Technovation*, vol. 124, pp. 1–17, Jun. 2023, doi: 10.1016/j.technovation.2023.102747.
- [31] L. I. Ruiz-Rojas, L. Salvador-Ullauri, and P. Acosta-Vargas, "Collaborative Working and Critical Thinking: Adoption of Generative Artificial Intelligence Tools in Higher Education," *Sustainability (Switzerland)*, vol. 16, no. 13, Jul. 2024, doi: 10.3390/su16135367.
- [32] F. Xiao (肖斐文), S. Zhu (朱思宇), and X. Wen (辛闻), "Exploring the Landscape of Generative AI (ChatGPT)-Powered Writing Instruction in English as a Foreign Language Education: A Scoping Review," *ECNU Review of Education*, pp. 1–19, Jan. 2025, doi: 10.1177/20965311241310881.
- [33] G. Wells, *Dialogic Inquiry: Towards a Sociocultural Practice and Theory of Education*. New York: Cambridge University Press, 1999.
- [34] I. Verenikina, "Understanding Scaffolding and the ZPD in Educational Research," University of Wollongong. [Online]. Available: <https://ro.uow.edu.au/edupapers/381>
- [35] R. R. Van Der Stuyf, "Scaffolding as a Teaching Strategy," *Adolescent learning and development*, vol. 52, no. 3, pp. 5–18, 2002.
- [36] D. H. Schunk, *Learning theories: an educational perspective*. Pearson Education, 2012.
- [37] J. Tudge, "Vygotsky, the zone of proximal development, and peer collaboration: Implications for classroom practice," in *Vygotsky and education: Instructional implications and applications of sociohistorical psychology*, L. C. Moll, Ed., Cambridge University Press, 1992, pp. 155–172.
- [38] D. Moher, A. Liberati, J. Tetzlaff, D. G. Altman, and PRISMA Group, "Preferred reporting items for systematic reviews and meta-analyses: the PRISMA statement," *Ann Intern Med*, vol. 151, no. 4, pp. 264–269, 2009, doi: 10.7326/L23-0065.
- [39] M. Lewis, "Focus group interviews in qualitative research: a review of the literature," *Action research E-reports*.
- [40] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qual Res Psychol*, vol. 3, no. 2, pp. 77–101, 2006, doi: 10.1191/1478088706qp063oa.
- [41] J. Liao, L. Zhong, L. Zhe, H. Xu, M. Liu, and T. Xie, "Scaffolding Computational Thinking With ChatGPT," *IEEE Transactions on Learning Technologies*, vol. 17, pp. 1668–1682, 2024, doi: 10.1109/TLT.2024.3392896.
- [42] H. Wan, H. Luo, M. Li, and X. Luo, "Automated Program Repair for Introductory Programming Assignments," *IEEE Transactions on Learning Technologies*, vol. 17, pp. 1745–1760, 2024, doi: 10.1109/TLT.2024.3403710.
- [43] M. D. Popovici, "ChatGPT in the Classroom. Exploring Its Potential and Limitations in a Functional Programming Course," *Int J Hum Comput Interact*, 2023, doi: 10.1080/10447318.2023.2269006.
- [44] F. French, D. Levi, C. Maczo, A. Simonaityte, S. Triantafyllidis, and G. Varda, "Creative Use of OpenAI in Education: Case Studies from Game Development," *Multimodal Technologies and Interaction*, vol. 7, no. 8, Aug. 2023, doi: 10.3390/mti7080081.
- [45] P. Prasad and A. Sane, "A Self-Regulated Learning Framework using Generative AI and its Application in CS Educational Intervention Design," in *SIGCSE 2024 - Proceedings of the 55th ACM Technical Symposium on Computer Science Education*, Association for Computing Machinery, Inc, Mar. 2024, pp. 1070–1076. doi: 10.1145/3626252.3630828.
- [46] A. Ouazaki, K. Bergram, J. C. Farah, D. Gillet, and A. Holzer, "Generative AI-Enabled Conversational Interaction to Support Self-Directed Learning Experiences in Transversal Computational Thinking," in *Proceedings of the 6th Conference on ACM Conversational User Interfaces, CUI 2024*, Association for Computing Machinery, Inc, Jul. 2024. doi: 10.1145/3640794.3665542.
- [47] A. Vadaparty *et al.*, "CS1-LLM: Integrating LLMs into CS1 Instruction," in *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, Association for Computing Machinery, Jul. 2024, pp. 297–303. doi: 10.1145/3649217.3653584.
- [48] A. C. M. Yang, J. Y. Lin, C. Y. Lin, and H. Ogata, "Enhancing python learning with PyTutor: Efficacy of a ChatGPT-Based intelligent tutoring system in programming education," *Computers and Education: Artificial Intelligence*, vol. 7, Dec. 2024, doi: 10.1016/j.caeai.2024.100309.
- [49] T. T. Dang, H. C. Ling, and N. Q. Tran, "Combating ChatGPT-Based Programming Test Cheating - An Evaluation Using Public Problems," in *Proceedings - 2024 6th International Conference on Computer Science and Technologies in Education, CSTE 2024*, Institute of Electrical and Electronics Engineers Inc.,

- 2024, pp. 161–165. doi: 10.1109/CSTE62025.2024.00037.
- [50] S. MacNeil *et al.*, “Decoding Logic Errors: A Comparative Study on Bug Detection by Students and Large Language Models,” in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Jan. 2024, pp. 11–18. doi: 10.1145/3636243.3636245.
- [51] N. Anderson, A. McGowan, L. Galway, P. Hanna, M. Collins, and D. Cutting, “Implementing Generative AI and Large Language Models in Education,” in *ISAS 2023 - 7th International Symposium on Innovative Approaches in Smart Technologies, Proceedings*, Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/ISAS60782.2023.10391517.
- [52] N. Baláž, J. Porubán, M. Horváth, and T. Kormaník, “Using ChatGPT During Implementation of Programs in Education,” in *OpenAccess Series in Informatics*, Schloss Dagstuhl- Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Sep. 2024. doi: 10.4230/OASIcs.ICPEC.2024.18.