Leveraging Large Language Models to Promote AI-Infused STEM Problem-Solving for Middle School Students

Ananya Rao[1], Krish Piryani[1], Shiyan Jiang[2], Tiffany Barnes[1], Jennifer Albert[3], Marnie Hill[1] and Bita Akram[1]

[1]North Carolina State University, NC, USA
[2]University of Pennsylvania, PA, USA
[3]The Citadel - The Military College of South Carolina, SC, USA

Abstract

AI-infused STEM problem-solving is becoming an increasingly important skill for the future STEM workforce, requiring innovative systems to support student learning. Integrating AI into STEM problem-solving requires building a strong foundation in students' computational thinking skills, which can be supported through carefully designed, technically advanced systems. In this paper, we propose augmenting a block-based programming environment specialized for AI-infused STEM problem-solving with large language model (LLM) capabilities to reinforce key computational thinking skills. Through our prior experimentation with students, we have identified three major computational thinking skills essential for mastering learning and transferring knowledge between contexts: abstraction, algorithmic thinking, and generalization. To this end, we enhance an LLM with knowledge about the high-level concept of breadth-first search (abstraction) and the ability to situate students' steps within the required steps of the BFS algorithm (algorithmic thinking). We then evaluate the LLM's ability to provide adaptive feedback as students implement BFS in various STEM contexts (generalization). We present a proof-of-concept evaluation demonstrating how an LLM trained with general BFS knowledge can provide adaptive, contextualized feedback in three different scientific scenarios: pathfinding (mathematics), contact-tracing (biology), and the time-to-live (TTL) package algorithm (networks). This functionality allows our environment to support students in developing abstraction, algorithmic thinking, and generalization skills when applying BFS to scientific problem-solving.

Keywords

K-12 AI Education, Computational Thinking for AI, Personalized Feedback, LLMs for AI Education, AI-Infused STEM Problem Solving

1. Introduction

Artificial intelligence (AI) in science, technology, engineering, and mathematics (STEM) fields is applied across a wide range of areas, from predictive modeling in climate science to automated data analysis in genomics and intelligent robotics in engineering [1]. These advancements underscore the importance of equipping learners with AI literacy — a foundational understanding of AI concepts and their integration into real-world problem solving [2]. For many years, research communities such as the AAAI task force, EAAI, and those in education and computer science education have recognized the critical need to broaden AI literacy [3]. However, the advent of large language models (LLMs) has amplified this need, as these technologies have seamlessly integrated into everyday life and work, making it vital for all disciplines to engage with AI concepts effectively.

A cornerstone of AI literacy is computational thinking (CT), a set of problem-solving skills rooted in computer science that enables learners to approach complex challenges systematically [4, 5]. Despite its significance, many educational curricula fail to emphasize CT skills as a critical foundation for AI-integrated problem solving. This gap is particularly notable in STEM education, where AI's transformative potential demands a deep understanding of its underlying principles and computational approaches. Addressing this gap requires not only teaching AI concepts but also providing ample, carefully crafted scaffolding to support learners in building robust CT skills.

CSEDM'25: 9th Educational Data Mining in Computer Science Education Workshop, June 20, 2025, Palermo, Sicily, Italy
arrao3@ncsu.edu (A. Rao); kopiryan@ncsu.edu (K. Piryani); jiangshiyan2013@gmail.com (S. Jiang); tmbarnes@ncsu.edu (T. Barnes); jalbert@citadel.edu (J. Albert); mehill6@ncsu.edu (M. Hill); bakram@ncsu.edu (B. Akram)

The emergence of LLMs offers unprecedented opportunities to reimagine educational practices in AI and CT. These advanced models can enable scalable, engaging, and effective scaffolding for learners, making it possible to design innovative curricula that enhance CT skills within the context of AI-infused problem solving. By leveraging LLMs, educators and researchers can create a new generation of tools and strategies that empower students to navigate the evolving landscape of STEM and AI-driven innovation.

In this paper, we draw on AI education literature and our experience integrating AI into biological sciences for middle school students to highlight the role of three essential computational thinking skills—abstraction, algorithmic thinking, and generalization—in empowering students to independently apply AI in STEM problem-solving [6]. We first introduce our learning environment, I-SAIL, a visual programming platform designed specifically for AI-infused STEM problem-solving [2]. I-SAIL is accompanied by an integrated curriculum that teaches the BFS algorithm and its application to a variety of STEM problems. Our curriculum follows a use-modify-create approach: during the use phase, students learn the fundamentals of AI; during the modify and create phases, they apply and adapt their knowledge to solve STEM-focused problems [7].

Finally, we describe our plans to design a tailored and fine-tuned LLM that will augment I-SAIL with narratives to support abstraction, algorithmic thinking, and generalization. First, the LLM will provide dynamic, context-sensitive narrations to reinforce students' abstract understanding of the algorithm. Then, the LLM will deliver feedback to guide students as they apply algorithmic thinking to implement BFS using a visual coding scheme. Finally, the LLM will support students with targeted feedback and narration to help them generalize their understanding of BFS to a novel STEM context. We discuss the steps we take to train an LLM for dynamic generation of the narration and how we augment it with tools to process students' problem-solving steps to provide adaptive scaffolding.

2. Background

2.1. AI-Infused STEM Problem-Solving for K-12

AI-infused STEM curricula aim to integrate AI concepts and tools into traditional K-12 STEM learning environments, empowering students to engage with real-world problems through computational and data-driven approaches [8]. These curricula emphasize interdisciplinary thinking by embedding AI principles such as machine learning, data analysis, and algorithm design into STEM lessons, preparing students to understand AI's transformative role in scientific and technical fields [9]. However, existing efforts often overemphasize machine learning while neglecting the broader landscape of AI methodologies and their impact on scientific inquiry and problem-solving [2]. Additionally, little attention has been given to how AI reshapes STEM practices, influencing hypothesis generation, experimentation, and the ethical dimensions of computational models. Addressing these gaps is essential for fostering a more comprehensive understanding of AI and its implications within and beyond STEM disciplines.

Learning technologies provide students with tangible tools to explore AI concepts, yet many fail to bridge the gap between AI algorithms and STEM problem-solving. Block-based platforms like MIT's Scratch, now enhanced with AI extensions [10], offer an intuitive introduction, while tools like Google's Teachable Machine [11] allow students to train models without delving into algorithmic mechanisms. Data science platforms such as Orange support deeper engagement by allowing students to visualize and analyze datasets, yet they often lack guidance on selecting appropriate algorithms for specific STEM challenges[12]. For example, while neural networks may be ideal for image classification in biology, decision trees or linear regression may be more effective for interpretable predictions in chemistry or environmental science. Without explicit support for understanding these trade-offs, students may struggle to critically evaluate, adapt, and apply AI models in meaningful STEM contexts. To fully realize the potential of AI-infused STEM education, learning technologies must go beyond black-box implementations, fostering students' ability to connect algorithmic reasoning with domain-specific problem-solving.

2.2. Scaffolding

Adaptive scaffolding in visual programming environments has been widely studied as a means of supporting K-12 learners in developing computational thinking and programming skills. These systems leverage various techniques, such as real-time feedback, personalized hints, and task-specific guidance, to address the diverse needs of learners [13]. For example, adaptive systems like ProgSnap2 [14] provide context-aware feedback and debugging support, allowing students to identify and correct errors while building problem-solving strategies. This body of research highlights the importance of balancing guidance with promoting independence, as overly directive scaffolding can hinder learners' ability to develop autonomy and critical thinking skills. Additionally, studies [15] emphasize the role of adaptive scaffolding in fostering engagement and persistence, particularly for students with limited prior programming experience. While these systems have demonstrated significant promise in enhancing learning outcomes, they often rely on pre-programmed rules or static models of learner behavior, limiting their ability to offer the dynamic, nuanced support that LLMs can provide [16].

Recent advancements in LLMs have opened opportunities to scaffold learning in visual programming environments by providing AI-powered assistance tailored to the needs of learners in K-12 education. For instance, systems like ChatScratch[10] integrate LLMs to empower children aged 6-12 to engage in autonomous programming by generating suggestions, explanations, and solutions based on their programming goals. Similarly, Scratch Copilot [17] explores the potential of AI to support creative coding for families, emphasizing collaborative learning scenarios. This approach demonstrates how AI can scaffold the programming process by suggesting next steps, debugging errors, and providing explanations to enhance understanding, particularly in multi-user learning environments. Both systems illustrate how LLMs can serve as co-creators and tutors, fostering creative exploration and computational skill-building. Building on these studies, the potential of LLMs for scaffolding visual programming can be further explored to support AI-infused STEM problem-solving.

3. I-SAIL Integrated Learning Environment for AI-Infused STEM Problem Solving

3.1. I-SAIL Learning Environment

The I-SAIL learning environment builds on the strengths of block-based programming (BBP) to lower entry barriers for novice learners while enabling AI-powered problem-solving in scientific contexts. Extending Snap! [18], I-SAIL incorporates pre-programmed AI infrastructures, customizable scientific models, and adaptive learning pathways, to engage students with real-world scientific challenges. Similar to existing block-based platforms such as BlockPy [19] for data analysis and NetsBlox [20] for networking, I-SAIL specialized BBP to integrate AI-specific functionalities and make AI concepts more accessible and applicable to interdisciplinary STEM problems. A key innovation of I-SAIL lies in its dual approach: it introduces custom AI blocks that simplify complex programming without sacrificing conceptual depth and integrates dynamic simulations and AI-driven modeling scenarios to support AI learning within STEM contexts. By bridging AI and STEM education, I-SAIL aligns with AI4K12 progression standards [21], fostering an inclusive and interdisciplinary learning environment that prepares students for AI-driven scientific problem-solving.

3.2. I-SAIL Pedagogy

Our curriculum integrates the Next Generation Science Standards (NGSS) and the AI4K12 framework [21] to ensure alignment with disciplinary best practices while fostering engagement at the intersection of science, engineering, and artificial intelligence. Research shows that integrating AI and STEM activities enhances student learning by reinforcing interdisciplinary connections and deepening conceptual understanding [22]. To make AI-infused STEM education more accessible, we emphasize contextualizing learning within culturally and personally relevant problem spaces, allowing students to

see AI as a practical tool for solving real-world challenges. This approach not only enhances motivation and engagement, particularly among underrepresented groups in computing, but also reinforces AI's applicability in domains such as climate science, health, and engineering [23]. By framing AI as both a scientific tool and a subject of study, our curriculum highlights the reciprocal relationship between scientific inquiry and computational modeling, enabling students to develop authentic problem-solving skills while critically examining AI's role in advancing scientific discovery. Embedding AI learning within familiar STEM contexts fosters a sense of belonging in computationally rich fields, bridging gaps between traditional STEM education and emerging AI-driven technologies.

3.2.1. Activity Design

3.3. Path Finding

The activities in the Use section are designed to help students develop a deep and practical understanding of the targeted AI algorithm. For instance, in the case of Breadth-First Search (BFS), students engage with a series of interactive demonstrations that illustrate the algorithm's mechanics. These activities guide students in exploring how BFS operates by analyzing how it identifies the shortest path between two cities on a map. Pathfinding is chosen as a learning context because it provides a relatable, concrete, and visually intuitive example, allowing students to understand the step-by-step execution of BFS and its real-world applications.

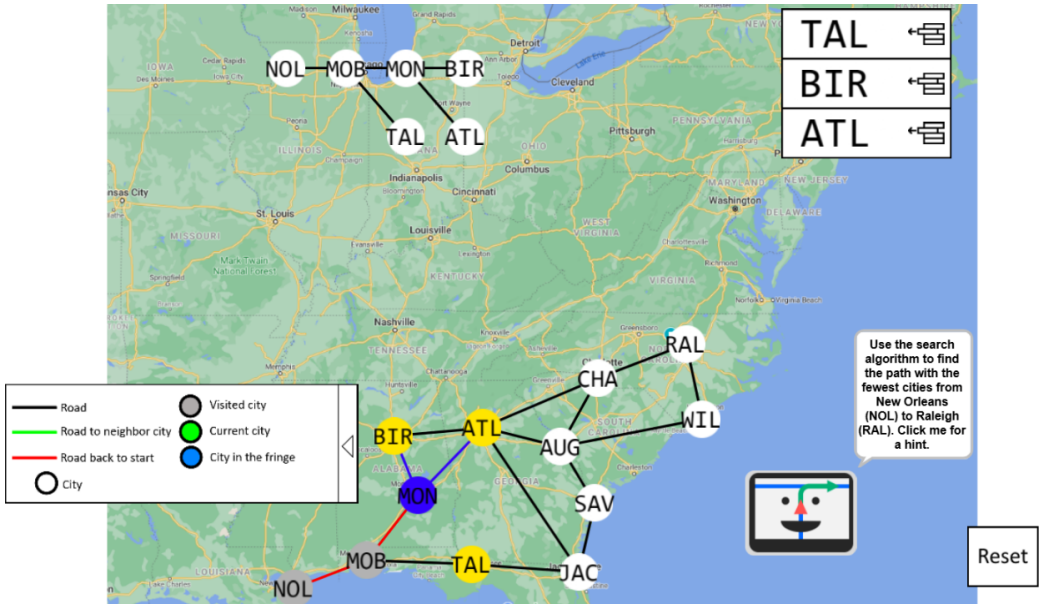

Figure 1: Pathfinding Activity

3.4. Contact Tracing APP

Beyond pathfinding, we integrate scientific and computing-rich contexts to further enhance students' understanding of BFS. One such example is contact tracing in public health, where students design an app to notify individuals exposed to an infectious disease. The app employs a graph-based representation of a social network and utilizes BFS to determine the shortest path from an infected individual to others in the network, thereby identifying those at highest risk and needing self-quarantine.

3.5. Package Time-to-Live

Additionally, we introduce Time to Live (TTL) as a computing-centric context for learning BFS. TTL is widely used in network routing to prevent data packets from looping indefinitely by setting a limit on

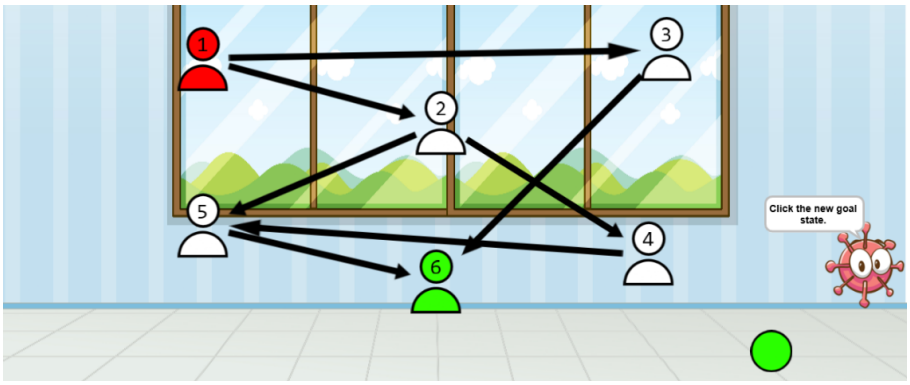

Figure 2: Contact Tracing App Activity

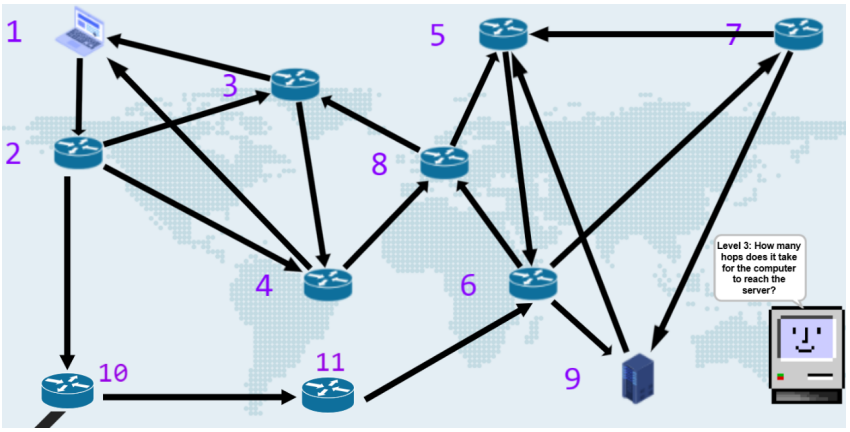

Figure 3: Package Time-to-Live Activity

the number of hops a packet can take. By examining how BFS is applied in TTL, students gain insights into its relevance in network protocols, data transmission, and system efficiency.

3.6. LLM Integration

In recent years, LLMs have demonstrated remarkable adaptability across various domains, making them promising tools for delivering contextualized and adaptive feedback to students. However, integrating LLMs into K-12 classrooms presents unique challenges, requiring careful training and guidance to ensure they provide meaningful support. In this work, we focus on training an LLM to interpret student problem-solving steps within the broader context of the Breadth-First Search (BFS) algorithm. Our approach involves equipping the model with an understanding of the problem-solving activities in the I-SAIL environment and guiding it to generate individualized feedback that is effective, relevant, and aligned with computational thinking principles.

3.7. Contextualization

We trained a large language model (LLM) to provide context-aware, adaptive feedback by providing detailed information about search algorithms and structured problem-solving in the I-SAIL learning environment. The first stage of training involved introducing the LLM to search as a fundamental concept in artificial intelligence, emphasizing its role in systematically exploring problem spaces to find optimal solutions. This foundation allowed the model to recognize structured search strategies and differentiate between correct and incorrect problem-solving approaches.

Next, we refined the LLM's ability to interpret student interactions in the I-SAIL system. Since students interact with a graph-based simulation, making decisions on which elements to explore and

in what order, the LLM needed to evaluate these decisions in real-time. Training prompts defined the mechanics of student engagement, such as adding elements to a search queue and processing them step by step, ensuring the model could assess correctness, detect errors, and provide relevant feedback.

To further improve accuracy, we simulated a range of student interactions, incorporating both correct and incorrect search sequences. This enabled the LLM to recognize common mistakes (e.g., skipping steps, exploring elements out of order, or revisiting processed elements) and adjust its responses accordingly. The model was also refined to ensure feedback was pedagogically effective—guiding students without directly providing solutions, thereby reinforcing structured reasoning and independent problem-solving.

3.8. Synthetic Data Generation

We generated synthetic data comprising graphs with varying BFS problem complexities and simulated student problem-solving sequences in the I-SAIL environment. These datasets included both correct and incorrect solution paths, enabling the LLM to learn structured problem-solving patterns, identify errors, and refine its feedback mechanisms for real student interactions.

3.8.1. Graph Generation

To provide students with multiple opportunities to practice BFS, we prompted the LLM to generate new graphs of varying difficulty. Specifically, we provided the LLM with multiple adjacency matrices from previously used BFS exercises to establish a baseline understanding of graph structures relevant to the learning environment. Based on these examples, the LLM generated two new graphs: one with a simpler structure, which was later adapted into the contact tracing activity (see Figure 2), and another more complex graph, which formed the basis for the Time-to-Live (TTL) package activity (see Figure 3). In the future, this mechanism can be used to automatically generate diverse problem-solving scenarios, enabling students to achieve mastery learning.

3.8.2. Problem-Solving Scenario Generation

To generate realistic student problem-solving scenarios, we employed prompt engineering to guide the LLM in producing both correct and incorrect BFS-based solutions. The prompt design focused on simulating step-by-step student interactions, incorporating common mistakes, and ensuring adherence to BFS principles while introducing controlled errors for feedback generation.

The initial prompts instructed the LLM to generate ten problem-solving sequences per scenario, ensuring a balance between correct and incorrect solutions. Each sequence simulated student decision-making while applying BFS, capturing variations in how students might approach the problem. Incorrect solutions were structured to reflect common errors, such as selecting an unreachable node, skipping neighbors, and mismanaging the BFS fringe. A key aspect of the prompt design was to maintain BFS constraints while allowing for realistic errors that students commonly make.

Refinements were necessary to improve the accuracy and diversity of the generated sequences. Early outputs sometimes violated BFS ordering, requiring adjustments to emphasize strict level-order traversal. Additionally, incorrect sequences initially overrepresented certain mistake types, leading to prompt modifications that ensured a broader distribution of errors across the dataset. Another refinement involved preventing logically impossible outputs, as some mistakes generated by the LLM were inconsistent with the given adjacency matrix. These refinements improved the model's ability to produce both valid and pedagogically meaningful scenarios. After scenario generation, manual evaluations were conducted to verify correctness and instructional relevance.

3.8.3. Individualized Contextualized Feedback Generation

Large language models can be leveraged to dynamically adapt narratives based on students' interactions with the learning environment, serving as co-authors of high-level explanations that guide them through

problem-solving. While LLMs demonstrate a strong capacity for explaining scientific concepts, their effectiveness depends on ensuring that (1) explanations match the student's level, (2) they encourage abstract thinking, and (3) they provide sufficient information to help students translate abstractions into algorithmic implementations. To achieve this, the feedback system was designed to provide contextualized and individualized guidance. By evaluating each student action in real time, the system identifies deviations from BFS principles and generates feedback that aligns with the specific problem scenario, reinforcing both conceptual understanding and problem-solving skills.

A key aspect of the design was ensuring that feedback is context-aware, meaning that it adapts to the domain of the problem. For example, in a pathfinding activity, feedback references cities and roads, while in a contact tracing scenario, it refers to people and social connections. This contextualization helps students relate algorithmic concepts to real-world applications, reinforcing computational thinking and problem-solving skills.

Additionally, the feedback system is designed to be non-directive, providing strategic hints rather than explicitly stating errors. The intent is to encourage students to reflect on their mistakes and refine their approach, promoting active learning. To maintain logical consistency, feedback is structured to assume that all previous steps were correct, preventing confusion and allowing students to focus on resolving the immediate issue.

3.8.4. An Exemplary Scenario and Feedback

In this section, we discuss one of the LLM-generated problem-solving scenarios along with its corresponding feedback. We further show examples of feedback generated in other domains for a similar mistake.

In a pathfinding scenario 1, a student applies BFS to find the shortest route from New Orleans (NOL) to Raleigh but incorrectly fails to add all neighboring cities before exploring(dequeuing) another, violating BFS principles. The student's actions follow this sequence:

NOL-G (Add New Orleans to the queue), NOL-F (Explore New Orleans), MOB-G (Add Mobile to the queue), MOB-F (Explore Mobile), MON-G (Add Montgomery to the queue), TAL-G (Add Tallahassee late), MON-F (Explore Montgomery), BIR-G (Add Birmingham to the queue), BIR-F (Explore Birmingham).

Here, -G indicates adding a city to the queue (fringe), while -F represents fully exploring a city by dequeuing it and expanding its neighbors. The mistake occurs when Mobile is explored before all of New Orleans' direct neighbors (like Tallahassee) are added to the queue, disrupting BFS's level-order traversal. The generated feedback states:

"A city was not added before dequeuing another. BFS requires that all children of a city be added to the fringe before moving on. Every path with the same distance from the start should have a chance to be explored before moving further."

A similar mistake occurs in contact tracing, where a student fails to add all direct contacts before investigating the next exposed individual, prompting the feedback:

"An individual was skipped before confirming their exposure risk. BFS guarantees that every contact at the same level is processed before moving to the next level. Ensure that all individuals at the same exposure level are traced before continuing."

In network TTL, this mistake translates to forwarding a packet before processing all at the current TTL level, leading to the feedback:

"Before processing node 5 (F-action), ensure that all its required neighbors have been added to the TTL queue. BFS requires all directly connected nodes to be added before continuing to process another node. Skipping an addition step before expanding node 5 could result in inefficient routing."

These examples illustrate how BFS principles apply across domains, with contextualized feedback ensuring students correctly implement BFS in different problem-solving contexts.

4. Evaluation

Evaluating LLM-generated feedback for students developing computational thinking (CT) skills—particularly abstraction, algorithmic thinking, and generalization—requires a structured rubric to assess its conceptual accuracy, depth, actionability, generalization, and clarity. Effective feedback should not only be conceptually sound but also specific, actionable, and transferable to various computational problems (Grover & Pea, 2013). The rubric (see table) evaluates feedback along five key dimensions: (1) conceptual accuracy, explanations of BFS are correct; (2) depth and specificity, determining whether feedback provides detailed, context-specific insights; (3) actionability, assessing whether students receive clear next steps; (4) contextualization & transferability, measuring how well the feedback connects BFS concepts to broader STEM applications; and (5) coherence & clarity, the feedback is logically structured and easy to understand.

To conduct the evaluation, two experts independently graded 30% of scenarios, followed by a discussion to resolve discrepancies. After achieving consensus, one expert proceeded to evaluate all 30 problem-solving scenarios. The overall evaluation results indicate that the LLM-generated feedback performed most effectively in Depth and Specificity (2.73) and Actionability (2.83), suggesting that feedback provided clear, structured explanations and actionable next steps. The results show a Coherence score of (2.70), implying that while the feedback was generally well-structured, there were occasional inconsistencies in logical flow.

The overall evaluation results indicate that the feedback system performed most effectively in Actionability (2.83/3.0), Depth and Specificity (2.73/3.0), and Contextualization (2.70/3.0). The system successfully provided clear, structured feedback, offering actionable next steps that aligned with students' problem-solving processes. However, Conceptual Accuracy (2.47/3.0) was slightly lower, suggesting occasional misinterpretations of BFS principles or inconsistencies in algorithmic explanations. Similarly, Coherence (2.70/3.0) was generally strong but showed some instances where feedback could have been more logically structured or explicitly linked to prior student actions.

5. Discussion

This study explores the integration of large language models (LLMs) into AI-infused STEM education to enhance students' learning with regard to computational thinking skills, including abstraction, algorithmic thinking, and generalization. Our approach involved synthetic data generation to model student problem-solving behaviors, leveraging an LLM to generate contextualized and individualized feedback tailored to student interactions. By embedding these capabilities into our learning environment, we aimed to support students in understanding and applying BFS-based problem-solving strategies across diverse STEM contexts.

In evaluating LLM-generated feedback, we found that the system effectively provided actionable, structured guidance, particularly in reinforcing algorithmic thinking and generalization. The feedback was well-aligned with student actions, offering strategic hints rather than direct answers, which encouraged deeper engagement with the problem-solving process. However, conceptual accuracy varied, with some responses misinterpreting BFS principles or oversimplifying explanations. Additionally, balancing specificity with generalization proved challenging, as overly detailed feedback sometimes constrained students' ability to transfer knowledge across contexts. Refining prompt design to improve the clarity, depth, and adaptability of explanations is a key area for future work, ensuring that feedback remains pedagogically sound and tailored to different learning needs.

Ultimately, this work demonstrates the potential for LLM-enhanced learning environments to provide scalable, personalized support in computational thinking education. By continuously refining synthetic data generation, prompt engineering, and feedback mechanisms, we aim to create a system that empowers students to engage in AI-driven problem-solving while fostering transferable computational skills applicable across STEM disciplines.

Table 1
LLM Feedback Rubric.

Criteria	3 (Strong)	2 (Moderate)	1 (Weak)
Conceptual Accuracy	Feedback provides accurate and precise explanations of BFS.	Feedback is partially correct but includes minor inaccuracies or lacks precision.	Feedback contains major conceptual errors or misrepresents BFS concepts.
Depth and Specificity	Feedback is detailed and context-specific, clearly addressing the student's computational thinking skill gaps.	Feedback is somewhat specific, but lacks depth or relevant details.	Feedback is too vague or generic, failing to guide learning.
Actionability	Feedback includes clear, actionable steps for students to improve their abstraction, algorithmic thinking, or generalization skills.	Feedback suggests improvements, but may be too general or require interpretation.	Feedback lacks actionable suggestions, making it unclear how to improve.
Contextualization	Feedback clearly connects BFS to diverse STEM contexts and problem-solving scenarios.	Feedback partially supports contextualization but may lack tangible connections.	Feedback fails to put the problem-solving scenario in the scientific context.
Coherence	Feedback is logically structured and easy to understand.	Feedback is somewhat clear, but may contain minor ambiguities.	Feedback is confusing, disorganized, or difficult to interpret.

6. Conclusion and Future Work

This study explored the use of large language models (LLMs) to enhance AI-infused STEM problem-solving, using breadth-first search (BFS) as a case study. We developed a structured approach that leverages synthetic data generation to create problem-solving scenarios and LLM-generated feedback to provide targeted, context-aware guidance. Our evaluation demonstrated that the system effectively produced actionable feedback, aligning with structured problem-solving strategies. However, variability in conceptual accuracy and the ability to capture nuanced student reasoning remain areas for improvement.

Future work will focus on refining synthetic data generation by incorporating real student interactions to enhance authenticity and diversity in problem-solving scenarios. Additionally, we aim to optimize LLM-generated feedback by fine-tuning prompt engineering techniques to improve explanation clarity and conceptual accuracy. Through these advancements, we aim to create a scalable and intelligent support system that enhances AI-driven problem-solving in STEM education for a diverse range of students.

Acknowledgments

This research was supported by the National Science Foundation (NSF) under Grant DUE-2405862.

Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

7. Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT, Grammarly in order to: Grammar and spelling check, Paraphrase and reword. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] S. Jiang, K. DesPortes, Y. Bergner, H. Zhang, I. Lee, K. Moore, Y. Cheng, B. Perret, B. Walsh, A. Guggenheim, et al., Agents, models, and ethics: Importance of interdisciplinary explorations in ai education, in: Proceedings of the 16th International Conference of the Learning Sciences-ICLS 2022, pp. 1763-1770, International Society of the Learning Sciences, 2022.

[2] B. Akram, S. Yoder, C. Tatar, S. Boorugu, I. Aderemi, S. Jiang, Towards an ai-infused interdisciplinary curriculum for middle-grade classrooms, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 36, 2022, pp. 12681–12688.

[3] D. Touretzky, F. Martin, D. Seehorn, C. Breazeal, T. Posner, Special session: Ai for k-12 guidelines initiative, in: Proceedings of the 50th ACM technical symposium on computer science education, 2019, pp. 492–493.

[4] V. Cateté, N. Lytle, Y. Dong, D. Boulden, B. Akram, J. Houchins, T. Barnes, E. Wiebe, J. Lester, B. Mott, et al., Infusing computational thinking into middle grade science classrooms: lessons learned, in: Proceedings of the 13th workshop in primary and secondary computing education, 2018, pp. 1–6.

[5] N. Lytle, V. Cateté, Y. Dong, D. Boulden, B. Akram, J. Houchins, T. Barnes, E. Wiebe, Ceo: A triangulated evaluation of a modeling-based ct-infused cs activity for non-cs middle grade students, in: Proceedings of the ACM Conference on Global Computing Education, 2019, pp. 58–64.

[6] S. Yoder, C. Tatar, I. Aderemi, S. Boorugu, S. Jiang, B. Akram, Gaining insight into effective teaching of ai problem-solving through csedm: A case study, in: 5th Workshop on computer science educational data mining, 2020.

[7] J. K. Houchins, D. C. Boulden, J. Lester, B. Mott, K. E. Boyer, E. N. Wiebe, How use-modify-create brings middle grades students to computational thinking, International Journal of Designs for Learning 12 (2021) 1–20.

[8] I. Lee, S. Ali, H. Zhang, D. DiPaola, C. Breazeal, Developing middle school students' ai literacy, in: Proceedings of the 52nd ACM technical symposium on computer science education, 2021, pp. 191–197.

[9] K. Park, B. Mott, S. Lee, A. Gupta, K. Jantaraweragul, K. Glazewski, J. A. Scribner, A. Ottenbreit-Leftwich, C. E. Hmelo-Silver, J. Lester, Investigating a visual interface for elementary students to formulate ai planning tasks, Journal of Computer Languages 73 (2022) 101157.

[10] L. Chen, S. Xiao, Y. Chen, Y. Song, R. Wu, L. Sun, Chatscratch: An ai-augmented system toward autonomous visual programming learning for children aged 6-12, in: Proceedings of the CHI Conference on Human Factors in Computing Systems, 2024, pp. 1–19.

[11] M. Carney, B. Webster, I. Alvarado, K. Phillips, N. Howell, J. Griffith, J. Jongejan, A. Pitaru, A. Chen, Teachable machine: Approachable web-based tool for exploring machine learning classification, in: Extended abstracts of the 2020 CHI conference on human factors in computing systems, 2020, pp. 1–8.

[12] D. Vaishnav, B. R. Rao, Comparison of machine learning algorithms and fruit classification using orange data mining tool, in: 2018 3rd international conference on inventive computation technologies (ICICT), IEEE, 2018, pp. 603–607.

[13] S. Marwan, B. Akram, T. Barnes, T. W. Price, Adaptive immediate feedback for block-based

programming: Design and evaluation, IEEE Transactions on Learning Technologies 15 (2022) 406–420.

[14] T. W. Price, D. Hovemeyer, K. Rivers, G. Gao, A. C. Bart, A. M. Kazerouni, B. A. Becker, A. Petersen, L. Gusukuma, S. H. Edwards, et al., Progsnap2: A flexible format for programming process data, in: Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education, 2020, pp. 356–362.

[15] W. Min, B. Mott, J. Lester, Adaptive scaffolding in an intelligent game-based learning environment for computer science, in: Proceedings of the workshop on AI-supported education for computer science (AIEDCS) at the 12th international conference on intelligent tutoring systems, 2014, pp. 41–50.

[16] M. A. Razafinirina, W. G. Dimbisoa, T. Mahatody, Pedagogical alignment of large language models (llm) for personalized learning: a survey, trends and challenges, Journal of Intelligent Learning Systems and Applications 16 (2024) 448–480.

[17] S. Druga, N. Otero, Scratch copilot evaluation: Assessing ai-assisted creative coding for families, arXiv preprint arXiv:2305.10417 (2023).

[18] B. Harvey, D. D. Garcia, T. Barnes, N. Titterton, D. Armendariz, L. Segars, E. Lemon, S. Morris, J. Paley, Snap!(build your own blocks), in: Proceeding of the 44th ACM technical symposium on Computer science education, 2013, pp. 759–759.

[19] A. C. Bart, J. Tibau, E. Tilevich, C. A. Shaffer, D. Kafura, Blockpy: An open access data-science environment for introductory programmers, Computer 50 (2017) 18–26.

[20] B. Broll, A. Ledeczi, Distributed programming with netsblox is a snap!, in: Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, 2017, pp. 640–640.

[21] D. Touretzky, C. Gardner-McCune, D. Seehorn, Machine learning and the five big ideas in ai, International Journal of Artificial Intelligence in Education 33 (2023) 233–266.

[22] E. Hestness, R. C. McDonald, W. Breslyn, J. R. McGinnis, C. Mouza, Science teacher professional development in climate change education informed by the next generation science standards, Journal of Geoscience Education 62 (2014) 319–329.

[23] A. Eguchi, H. Okada, Y. Muto, Contextualizing ai education for k-12 students to enhance their learning of ai literacy through culturally responsive approaches, KI-Künstliche Intelligenz 35 (2021) 153–161.