



# Adaptive vs. planned metacognitive scaffolding for computational thinking: Evidence from generative AI-supported programming in elementary education

Jinfang Liu<sup>a</sup> , Yi Zhang<sup>a,\*</sup> , Wei Li<sup>b</sup>, Qiyun Wang<sup>c</sup>, Pingxiu Niu<sup>d</sup>, Xue Zhang<sup>a</sup>

<sup>a</sup> Faculty of Artificial Intelligence in Education, Central China Normal University, Wuhan 430079, Hubei, China

<sup>b</sup> School of Educational Science and Technology, Northwest Minzu University, Lanzhou 730030, Gansu, China

<sup>c</sup> National Institute of Education, Nanyang Technological University, Singapore 637616, Singapore

<sup>d</sup> Wuhan Sino-French Eco-City Foreign Languages School, Wuhan 430079, Hubei, China

## ARTICLE INFO

### Keywords:

Computational thinking  
Generative AI agents  
Adaptive learning  
Adaptive metacognitive scaffolding  
Planned metacognitive scaffolding  
Elementary education

## ABSTRACT

Metacognitive scaffolding plays a pivotal role in supporting the development of computational thinking (CT) among elementary students. Compared to planned scaffolding, adaptive scaffolding supported by generative AI (GAI) agents offers a promising option by providing real-time, adaptive and personalized response. This study investigated how adaptive metacognitive scaffolding (AMS) and planned metacognitive scaffolding (PMS) influenced elementary students' CT in a GAI-supported programming environment. 74 students were assigned to AMS, PMS, or control conditions. Using a mixed-methods approach, we examined CT task performance, CT skill use, metacognitive regulation, and cognitive load through standardized assessments, programming artifacts, discourse logs, video-coded behaviors, classroom observation records, cognitive load ratings. Results showed that AMS significantly enhanced contextualized CT performance and promoted more complex, diverse skill use compared to PMS and control groups. AMS also fostered more recursive metacognitive behaviors and reduced mental efforts, while PMS supported higher meaningful engagement. These findings highlight the potential of adaptive scaffolding powered by GAI to support CT development in elementary programming education.

## 1. Introduction

The rise of generative AI (GAI), such as ChatGPT and other large language models (LLMs), plays a pivotal role in developing elementary students' computational thinking (CT) (Yilmaz & Yilmaz, 2023). CT encompasses abilities such as decomposition, pattern recognition, abstraction, algorithmic design, and evaluation, which has been recognized as a foundational skill for addressing complex problems across academic and real-world domains (Israel-Fishelson & Hershkovitz, 2022). In response, CT has been increasingly integrated into K-12 education globally, often via visual programming platforms like Scratch, which aim to lower entry barriers for novice learners (Grover, 2021; Xu et al., 2023). Yet, many elementary students struggle to grasp abstract programming

This article is part of a special issue entitled: GenAI enhanced learning published in Computers & Education.

\* Corresponding author. Faculty of Artificial Intelligence in Education, Central China Normal University, NO.152 Luoyu Street, HongShan District, Wuhan 430079, Hubei, China.

E-mail addresses: [742410106@qq.com](mailto:742410106@qq.com) (J. Liu), [zhangyi@mail.ccnu.edu.cn](mailto:zhangyi@mail.ccnu.edu.cn) (Y. Zhang), [liweigzh@163.com](mailto:liweigzh@163.com) (W. Li), [qiyun.wang@nie.edu.sg](mailto:qiyun.wang@nie.edu.sg) (Q. Wang), [1149798221@qq.com](mailto:1149798221@qq.com) (P. Niu), [2818428254@qq.com](mailto:2818428254@qq.com) (X. Zhang).

constructs—especially loops and conditionals—and frequently encounter difficulties in designing structured algorithmic solutions (Hsu & Gainsburg, 2021; Weintrop & Wilensky, 2019). Through natural language interaction, GAI agents can provide timely, adaptive, and personalized feedback, helping students understand abstract concepts and navigate ill-structured problems (Rahman & Watanobe, 2023). Recent studies report encouraging results from using GAI tools to assist novice programmers, including improved comprehension and enhanced CT skills (Hijón-Neira et al., 2024; Weng et al., 2024).

However, concerns have emerged regarding potential overreliance on GAI-generated solutions (Rahman & Watanobe, 2023). Some argue that depending too heavily on automated feedback may undermine students' active cognitive engagement and weaken their problem-solving abilities (Ifelebuegu et al., 2023; Memarian & Doleck, 2023). Elementary students, in particular, may lack the metacognitive strategies necessary to critically evaluate GAI responses or reflect on their own learning processes (Mosaiyebzadeh et al., 2023). These findings suggest that while GAI holds educational promise, its effectiveness depends on instructional supports that actively mitigate over-reliance risks (Li & Samah, 2025). Specifically, well-designed scaffolding should intervene by prompting students to reflect GAI-generated solutions and consciously regulate their problem-solving strategies (Avci, 2024; Mittal et al., 2024). By embedding appropriate cognitive prompts and encouraging justification of answers, scaffolding helps students shift from passive reception of GAI outputs to active cognitive processing, thereby reducing the likelihood of disengagement from automation (Ba et al., 2025; Karaoglan Yilmaz & Yilmaz, 2025; Lee & Wu, 2025)—especially for younger learners with no prior experience (Chang et al., 2023).

To address this challenge, researchers have increasingly emphasized the value of metacognitive scaffolding, which helps students regulate their thinking through planning, monitoring, and adjusting actions rather than passively accepting GAI outputs (Wu et al., 2025). Scaffolding mechanisms—such as prompting students to justify GAI-provided solutions or reflect on the reasoning process—are designed to activate students' metacognition and foster strategic engagement during problem-solving tasks (Joksimovic et al., 2023; Urban et al., 2024), especially in programming contexts (Kim et al., 2022; Zhou et al., 2023). A key distinction in this domain lies between planned metacognitive scaffolding (PMS), which provides pre-set prompts in a uniform manner, and adaptive metacognitive scaffolding (AMS), which delivers personalized, real-time feedback tailored to learners' ongoing needs (Saye & Brush, 2002). While PMS is easier to implement, its static nature may not sufficiently address students' varying levels of reliance on GAI feedback. In contrast, AMS—especially when powered by GAI—can dynamically detect students' problem-solving behaviors and deliver timely prompts that encourage deeper cognitive engagement, actively correcting tendencies toward passive acceptance (Al-Fattal, 2025; Sardi et al., 2025; Yan et al., 2025). The latest research suggests that integrating AMS with GAI agents may enhance college students' deep learning by aligning support with learning trajectories (Wu et al., 2025).

Despite this promise, most existing research on GAI and CT has focused on higher education or generalized outcomes (Crompton & Burke, 2023), with limited attention to how different scaffolding types function in elementary programming education (Liao et al., 2024; Xu et al., 2023). Aligned with the special issue, we frame this study within personalized learning—adapting instruction to individual learners' needs, strengths, and interests—and conceptualize AMS as the in-situ mechanism that delivers such personalization in GAI-supported tasks.

To address this gap, the present quasi-experimental study investigates the effects of adaptive and planned metacognitive scaffolding on elementary students' CT learning in a visual programming environment (VPE) supported by GAI agents. The findings aim to inform the design of personalized instructional support systems and contribute new insights into how different types of scaffolding influence CT development in younger learners.

## 2. Literature review

Research on CT in elementary VPEs has consistently documented difficulties with abstract control structures (e.g., loops, conditionals) and with coordinating multiple CT skills, underscoring the need for targeted metacognitive scaffolding that supports planning, monitoring, and adjusting. PMS offers structured, uniformly delivered routines that cultivate reflection and germane investment, yet its static timing and granularity limit responsiveness to learners' moment-to-moment needs. In contrast, AMS, enabled by GAI, delivers brief, behavior-triggered and context-aware prompts at the point of need to restore in-task regulation, with implications for improved contextualized task performance, richer integration of CT skills, and a more favorable cognitive-load profile. This background frames our comparison of AMS and PMS within a GAI-supported VPE.

### 2.1. CT in visual programming among elementary learners

CT refers to the cognitive processes involved in formulating problems and expressing their solutions in ways that a computer or other agents can effectively execute (Aho, 2012; Wing, 2014). While CT has been conceptualized in multiple frameworks, recent scholarship generally distinguishes between two dimensions: domain-specific CT, which focuses on students' mastery of programming constructs (e.g., sequence, loop, condition, data, operator, event, and sorting), and domain-general CT, which emphasizes transferable problem-solving skills such as decomposition, pattern recognition, abstraction, algorithm design, and evaluation (Gao et al., 2025; Shute et al., 2017; Weintrop et al., 2016). These two perspectives are commonly used to assess CT development and together reflect the dual cognitive and technical nature of CT in educational programming contexts (Gao et al., 2025; Tsai et al., 2021).

In elementary education, VPEs, such as Scratch, have been widely implemented in elementary classrooms to support the development of CT in an accessible and age-appropriate way. By offering drag-and-drop interfaces, these VPEs minimize syntactic complexity and allow younger learners to focus on algorithmic logic rather than code syntax (Troiano et al., 2025; Wong, 2024). A growing body of research suggests that VPEs can enhance students' engagement, and willingness to tackle challenging tasks, making it

a promising entry point for CT education (Montuori et al., 2024; Wong et al., 2024).

However, the benefits of VPEs alone are insufficient for ensuring meaningful CT development. Younger learners are still developing executive functions, including working memory, attention regulation, and planning ability (Lye & Koh, 2014), thus they often face developmental challenges in grasping abstract programming concepts and applying structured algorithmic logic (Fanchamps et al., 2021; Grover, 2021). Although VPEs lower the technical threshold for participation, they do not eliminate the cognitive demands associated with planning, debugging, and implementing structured solutions (Batni & Junaini, 2025; Ma et al., 2023). Furthermore, many elementary students lack effective metacognitive strategies for monitoring and adjusting their problem-solving processes, often resulting in misconceptions, fragmented conceptual understanding, and reduced persistence (Huang et al., 2024; Kim et al., 2022).

These challenges highlight the need to go beyond merely introducing CT content; educators must also support the processes through which CT is constructed and enacted. Specifically, there is a growing need to embed cognitive and metacognitive scaffolding that helps learners manage the recursive and strategic demands of CT tasks. Yet much of the existing research has focused on outcome measures (e.g., task scores or CT concept mastery), with comparatively little emphasis on instructional scaffolds that support students' cognitive and metacognitive engagement throughout the learning process (Shin et al., 2023; Zhou et al., 2023).

Personalization is essential for elementary novices engaging with GAI. Younger learners benefit from instructional support that (a) externalizes planning and monitoring moves, (b) counteracts automation bias by requiring explanation and justification of GAI suggestions, and (c) calibrates task difficulty to sustain productive cognitive load (Eberhart et al., 2025; Lee & Wu, 2025; Montuori et al., 2024; Wu & Yu, 2024; Yan et al., 2025). These needs motivate a direct contrast between PMS—stable, teachable routines that build reflective habits—and AMS—contingent, just-in-time prompts responsive to learners' moment-to-moment states in GAI-mediated tasks.

## 2.2. Metacognitive scaffolding in CT

Metacognitive scaffolding refers to instructional support designed to help learners plan, monitor, and adjust their cognitive processes during complex tasks (Azevedo et al., 2004). In CT education, such scaffolding plays a vital role in guiding students to plan solutions, monitor their strategies, evaluate progress, and adjust when encountering errors (Dorgnier et al., 2025; Eberhart et al., 2025; Wu et al., 2025). This is especially crucial for elementary students, who often lack spontaneous metacognitive strategies and need structured support to build self-regulation alongside CT proficiency (Huang et al., 2024; Wang et al., 2024).

A widely used form of metacognitive scaffolding is planned scaffolding, which delivers pre-designed prompts, checklists, or reflection worksheets to support students' engagement across the learning process. For instance, teachers may provide scaffolding cards that ask students to predict program output or explain the rationale behind specific blocks in Scratch (Huang et al., 2024; Shin et al., 2023). These tools aim to guide learners through essential phases of problem-solving, including planning, monitoring, and adjusting (Kim et al., 2022; Shin et al., 2023). Empirical studies have shown that planned scaffolds can enhance task accuracy, foster strategy awareness, and prevent early disengagement (e.g., Gao et al., 2025; Li et al., 2024; Wang et al., 2024).

However, planned scaffolding has inherent limitations: delivered as a fixed sequence of prompts, it is insensitive to students' moment-to-moment needs or misconceptions. For elementary novices, this static delivery can invite shallow engagement and missed opportunities to apply supports at the right time and grain size (Lim et al., 2024; Zhou et al., 2023). These constraints motivate the turn to adaptive scaffolding that personalizes guidance in situ—tuning the timing, content, and intensity of prompts to sustain monitoring and adjusting during CT learning (Hijón-Neira et al., 2024; Weng et al., 2024). Because PMS is static and non-contingent, younger learners may miss when to apply supports—foreshadowing our hypotheses about adaptive prompts in Section 2.4.

## 2.3. Adaptive scaffolding supported by GAI

Adaptive scaffolding refers to dynamic, responsive instructional support that adjusts in real time based on students' ongoing performance and needs (Roll & Wyllie, 2016). In contrast to planned scaffolds, adaptive scaffolding provides immediate, tailored guidance that addresses emerging challenges in the learning process (Lim et al., 2024; Zheng et al., 2023). In CT education, this can help students reframe problems, refine logic, or test new approaches as they encounter difficulties (Rahman & Watanobe, 2023). Such supports are especially critical for elementary learners, who often require individualized feedback to stay engaged and overcome cognitive overload (van der Meij & de Jong, 2011).

Recent advances in GAI have made adaptive scaffolding more scalable and context-sensitive. LLMs can interpret natural language, identify learner misconceptions, and offer targeted prompts or skill suggestions in real time (Wu et al., 2025). In CT contexts, GAI can serve as an interactive agent that prompts self-explanation, diagnoses logic errors, and encourages strategy exploration (Hijón-Neira et al., 2024; Song et al., 2025). Moreover, when applied in VPEs, GAI agents can support students with context-aware prompts—e.g., recommending blocks, asking “why” questions, or triggering metacognitive strategies (Gani et al., 2025; Huang et al., 2025).

Studies have also suggested that GAI tools can facilitate CT skills and scaffold metacognitive questioning, supporting a more intentional and recursive approach to learning (Fang, Guo, et al., 2025; Wu et al., 2025). These features make GAI-powered scaffolds particularly promising for younger learners navigating the complex and often non-linear nature of CT tasks. In addition, Chen et al. (2025) and Huang et al. (2025) revealed that GAI had the potential to reduce mental effort and optimize irrelevant cognitive load.

Realizing these benefits in elementary classrooms, however, hinges on careful design calibration to learners' age, task complexity, and the guidance-autonomy balance (Lee & Wu, 2025; Liao et al., 2024). These properties suggest advantages for contextualized performance and in-task regulation, motivating our empirical comparison of AMS and PMS in a GAI-supported VPE.

## 2.4. The present study

Despite growing interest in GAI-supported CT, prior work has largely emphasized planned, uniformly delivered supports and higher-education contexts; direct comparisons of planned versus adaptive scaffolds in elementary programming remain scarce (Fang, Chen, et al., 2025; Liao et al., 2024; Tu & Hwang, 2024; Wu & Yu, 2024). Building on this gap, we compare AMS with PMS in a GAI-supported VPE for elementary learners.

We examine three complementary outcome families: (a) CT task performance (product), (b) CT skill use (process enactment), and (c) metacognitive regulation and cognitive load (psychological mechanism). CT task performance is captured by an abstract, decontextualized measure (CT test, CTT), a contextualized, problem-based measure (Bebras CT challenge), and indicators from an open-ended programming project (goal attainment; program size reflecting structural complexity). CT skill use is assessed from Conbot interaction logs and problem-solving videos, revealing how learners enact and coordinate strategies (e.g., control structures, abstraction, pattern recognition).

For the mechanism, we treat metacognitive regulation as tri-phase cycles of planning, monitoring, and adjusting during problem solving, operationalized via the frequency and completeness of P→M→A episodes and their sequencing (from classroom observations, Scratch process logs, and post-task reflections). In parallel, we use cognitive load as a proxy for metacognitive demands: intrinsic cognitive load (ICL) indexes in-task complexity, extraneous cognitive load (ECL) reflects instructional clarity/efficiency, and germane cognitive load (GCL) reflects constructive investment in schema building (Leppink et al., 2013). Under this framing, AMS—via behavior-triggered, just-in-time prompts—should both increase complete P→M→A cycles and relieve ICL; PMS—via structured reflection—should elevate GCL, with minimal effects expected on ECL when instructions are clear.

## 2.5. Hypotheses

H1 (CT task performance). Compared with PMS, AMS will yield stronger gains on contextualized CT tasks (Bebras; open-ended indicators), whereas differences on abstract CT concepts (CTt) will be smaller within a short intervention.

H2 (CT skill use). Compared with PMS, AMS will lead to more complex integration of CT skills as evidenced in programming artifacts, Conbot logs, and video-coded behaviors.

H3 (metacognitive regulation and cognitive load). AMS will increase the frequency and completeness of P→M→A regulation episodes and reduce ICL during problem solving; PMS will increase GCL through structured self-explanation and retrospection routines, with ECL differences expected to be minimal.

## 3. GAI agents development and design

To support students' learning and regulation in the Scratch-based VPE, two GAI agents were developed and deployed: Conbot, a knowledge-based assistant for all learners, and Xiaotongbao, the AMS exclusively for the AMS group. Both agents were implemented on the Coze platform using the Dou Bao LLM and structured through a three-layer architecture (see Fig. 1).

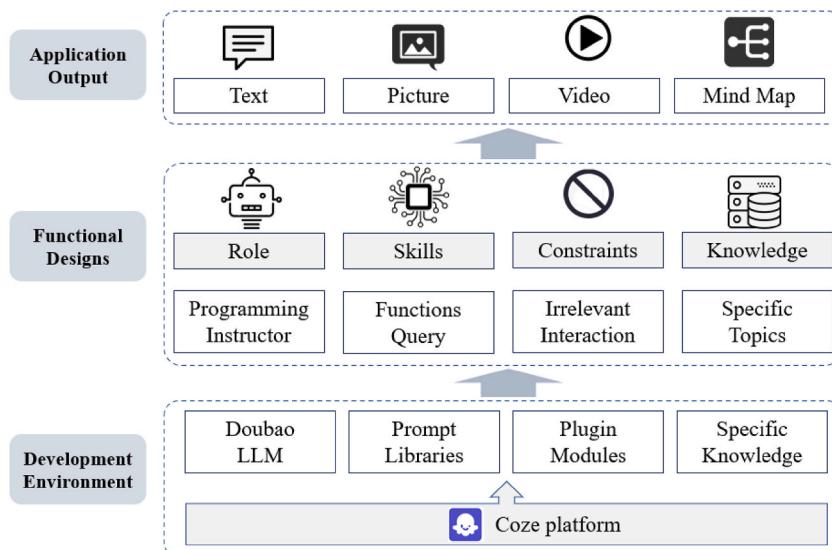


Fig. 1. Architecture and design layers of Conbot.

### 3.1. Conbot: Knowledge-level GAI agent for consistent instructional support

Conbot served as a foundational GAI question and answering (Q & A) chatbot providing real-time responses to students' queries about key programming concepts. It was embedded across all conditions to ensure baseline instructional supports. The development framework of Conbot comprised three layers:

**Development Environment Layer:** Conbot was developed via the Coze platform, supported by the Dou Bao LLM. This environment enabled integration with prompt libraries, plugin modules, and domain-specific knowledge databases aligned with the Scratch curriculum.

**Functional Design Layer:** The chatbot was assigned a role ("fifth-grade programming teacher") and core skills (e.g., "explain sorting algorithm", "describe Scratch block functions"). Added relevant knowledge in database and design constraints ensured age-appropriateness and topical focus.

**Application Output Layer:** Conbot supported multimodal outputs including text explanations, picture and video illustrations, and concept maps. These were adaptively presented based on student inputs (e.g., the keyword "bubble sort" triggered a textual explanation, optionally followed by a video or map mind).

### 3.2. Xiaotongbao: an adaptive GAI agent supporting metacognitive strategies

Building upon Conbot, Xiaotongbao introduced an adaptive mechanism for metacognitive scaffolding. It retained Conbot's knowledge functions while adding behavior-triggered interventions designed around planning, monitoring, adjusting strategies.

Xiaotongbao's system detected student behavior patterns—such as frequent structural errors—and responded with context-aware prompts. These prompts were categorized according to metacognitive phases and aligned with CT development stages. For instance, when a student initiated a bubble sort task, Xiaotongbao activated planning prompts ("What will you do first?"), followed by monitoring support ("What was the result of the second round of sorting?"), and adjusting suggestions if errors were detected. The agent continued this recursive cycle until the student regained autonomous problem-solving flow.

As shown in Fig. 2, Xiaotongbao extends the base knowledge delivery functions by integrating a behavioral recognizing module and a dynamic prompt engine. These enhancements enable the agent to provide real-time metacognitive coaching aligned with students' task progress.

It should be noted that, for ease of understanding, we used Conbot to refer to the Q & A system (including the Q & A functions of knowledge in Xiaotongbao) in the following text, and used Xiaotongbao specifically for AMS.

## 4. Methodology

### 4.1. Participants

A total of 74 fifth-grade students (aged 9–11) from an elementary school in an urban area of mainland China participated in this study. The students were randomly assigned to one of three groups: AMS (15 boys, 13 girls), PMS (14 boys, 12 girls), and control groups (11 boys, 9 girls). Most participants had minimal or no prior experiences with Scratch and GAI interaction. To address this, all students completed two preparatory sessions before the formal intervention, during which they were introduced to the Scratch interface and human-agent dialogues using scaffolded example tasks.

The intervention was conducted during regularly scheduled computing classes. Informed consent was obtained from all participants' guardians. The study received approval from the Ethics Committee of the first author's affiliated institution and complied with all relevant ethical guidelines for research involving human participants.

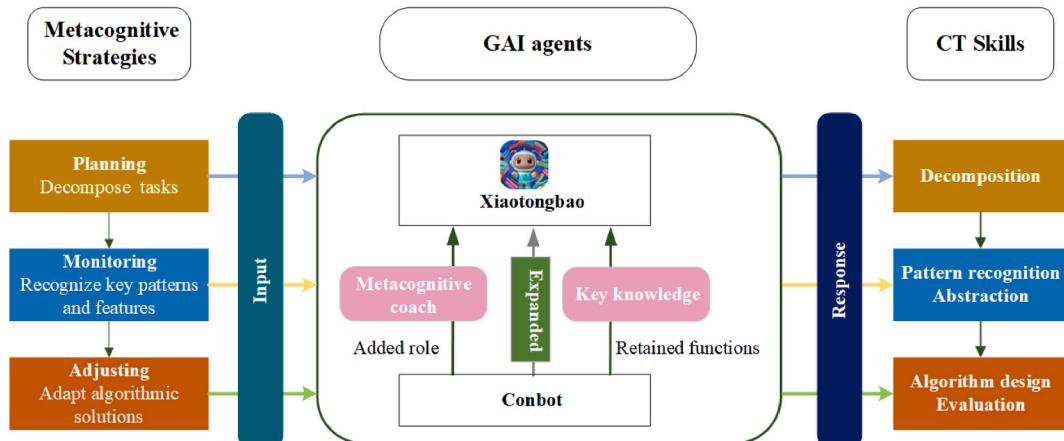


Fig. 2. Functional architecture of Xiaotongbao used in the AMS group.

#### 4.2. Experiment design

This study adopted a pretest–posttest quasi-experimental design involving three independent groups: an AMS group, a PMS group, and a control group. The independent variable was the types of metacognitive scaffolding provided, with three levels:

- AMS: real-time, personalized prompts generated by Xiaotongbao embedded in the programming tasks for AMS group.
- PMS: pre-designed worksheets and guiding prompts distributed at fixed stages by the teacher for PMS group.
- No scaffolding intervention: the same instructional content but without any form of metacognitive scaffolding for control group.

The dependent variables included three core dimensions of CT development:

- CT task performance: evaluated via standardized pretest and posttest instruments and program projects to assess students' ability of completing programming tasks with accuracy and completeness.
- CT skill use: assessed using two complementary methods: (1) analysis of students' input logs with the Conbot and (2) behavioral coding of video-recorded programming sessions, focusing on five key CT skills (decomposition, pattern recognition, abstraction, algorithm design, and evaluation).
- Metacognitive regulation behaviors: coded from observation checklists and student reflection logs, and Scratch records focusing on planning, monitoring, and adjusting behaviors during problem-solving tasks.

To ensure comparability across conditions, all three groups followed the same visual programming curriculum within the Scratch environment. The total intervention spanned 13 weeks: two weeks of preparatory training, six weeks of experimental instruction with embedded scaffolding conditions, and five weeks for pretests, posttests, and final programming project. Instructor, class size, instructional materials, and time-on-task were held constant across groups; the only manipulated factor was the type of metacognitive scaffolding provided.

#### 4.3. Materials

This section outlines the instruments used for data collection and the instructional materials used across experimental conditions.

##### 4.3.1. Measurement instruments

To examine the effects of different types of metacognitive scaffolding on students' CT development, we assessed three primary outcome dimensions: CT task performance, CT skill use, and metacognitive regulation behaviors. Additionally, a post-task cognitive load measure was used to capture students' subjective task experience.

###### (1) CT Task Performance

CT performance was evaluated using three complementary assessments:

- a. Standardized assessments. We used a 28-item CT test (CTt, adapted from Román-González et al., 2017) and a 15-item Bebras challenge (easy/medium/hard; Bebras Australia challenge 2020–2022). Both target fundamental CT constructs (e.g., sequence, loop, condition). Items were localized for upper-elementary students through a translate–review–revise process and expert review by four reviewers (two AI professors with  $\geq 10$  years K–12 experience; two PhD candidates in psychology/education). CTt was scored 0/1 (max 28); Bebras was weighted by difficulty (0/6/9/12; max 150). This combination is commonly used in Chinese elementary CT research (e.g., Gao et al., 2025).
- b. Open-ended programming project. Students created a Scratch program to find and display the “lucky” (largest) numbers from three arrays. Performance was scored on two dimensions: goal attainment (task completion) and structural complexity (whether to use nested loops/conditionals).

###### (2) CT Skills Use

Students' use of CT skills was assessed using a dual-mode strategy:

- a. Conbot interaction logs. Students' text inputs to the Conbot were coded into seven CT dimensions—sequence, loop, condition, data, operator, event, sorting (added for observed math/data behaviors)—adopted from Lei et al. (2024) and Zhou et al. (2024). After cleaning, 211 valid records remained (multi-label coding scheme in *Supplementary Materials Table S1*).
- b. Problem-solving videos. Structured video coding captured five CT skills—decomposition, pattern recognition, abstraction, algorithm design, evaluation—during task completion (see *Supplementary Materials Table S2*). Frequency counts per student were used for group comparisons.

###### (3) Metacognitive Regulation Behaviors

Triangulated sources included classroom observations, post-task reflection worksheets, and Scratch process logs (block deletions/additions, order revisions). Behaviors were coded for planning, monitoring, and adjusting with a standard checklist aligned to three

regulation phases.

#### 4.4. Supplementary measure: cognitive load

To capture students' perceived mental workload, a post-task cognitive load questionnaire was administered after the transfer task. A post-task cognitive-load questionnaire (Leppink et al., 2013) with three subscales—*intrinsic* (3 items), *extraneous* (3 items), *germane* (4 items)—was adapted via translation and expert review to ensure age-appropriate wording (e.g., "The number-sorting task is difficult for me", see *Supplementary Materials Table S6*). Items used a 7-point Likert scale; reliability was high ( $\alpha = 0.939/0.899/0.949$  for ICL/ECL/GCL).

##### 4.4.1. Coding schemes and expert validation

Open-ended data (Scratch projects, Conbot inputs, videos, observation notes) were coded with structured rubrics grounded in prior work and validated by domain experts.

###### (1) CT task performance: Programming Project

Rubric derived from Chao (2016) and Gao et al. (2025) captured goal attainment and program structure (see Table 1, detailed principles in *Supplementary Materials Table S3*). We distinguished Program Size 1 solutions using nested control structures and Program Size 2 (without nesting) to reflect structural complexity. Two raters (a computing educator, and a graduate student) scored 34 % of submissions ( $\kappa = 0.803$ ), resolved disagreements, then completed scoring.

###### (2) CT Skill Use: Conbot Input Logs

After removing off-task queries, 211 inputs were multi-coded into seven dimensions (sequence, loop, condition, data, operator, event, sorting; see Table 2). Two computational or educational experts reviewed the manual; two trained coders independently coded 30 % of records ( $\kappa = 0.821$ ), then reconciled and completed coding. We subsequently used Epistemic Network Analysis (ENA) to visualize co-occurrence patterns across conditions.

###### (3) CT Skill Use: Problem-Solving Videos

To complement log analysis, we also conducted structured video coding to assess students' CT skill use during task completion. Coders labeled each observed behavior (e.g., modifying command sequences, identifying repeated patterns). A sample of 16 students' videos (30 %) yielded  $\kappa = 0.812$ . Frequency counts per student were calculated and compared across groups (AMS, PMS, control) using nonparametric tests and visualized via boxplots.

###### (4) Metacognitive Regulation: Observation and Reflection

Students' metacognitive regulation behaviors were coded from three sources: classroom observation protocols, post-task reflection worksheets, Scratch process logs. Coding focused on planning, monitoring, and adjusting, with each behavior coded for frequencies and order. Observers were trained on a pre-validated instrument, achieving agreement through dual coding of two full lessons and calibration meetings. Detailed coding schemes used for behavioral analysis are provided in *Supplementary Materials Tables S4–S5*.

**Table 1**  
Coding table for students' final programming projects.

Dimensions	Description
Goal Attainment (Maximum score: 6)	Students used a command or set of commands to achieve a sub-goal: Found or displayed each lucky number, scoring one point.
Program Size 1 (Maximum score: 32)	If a student used a nested structure to find a lucky number, the scoring principles were prioritized. Loop structure: a) Students received five points for using both inner and outer loops, plus three points for each additional repeat until b) Students received two points each for correctly using the inner or outer loop count/condition or command Conditional structure: a) Students scored four points for using if-then b) Students scored four points each, by using the correct condition or command
Program Size 2 (Maximum score: 28)	If a student used simple structures to find a lucky number, the scoring principles were followed. Loop structure: a) Students received one point for using a repeating structure and two points for using a repeating until structure b) Students scored one point for using the correct loop count/condition or command Conditional structure: a) Students scored one point for using if-then b) Students scored one point by using the correct condition or command

**Table 2**

Coding scheme for Conbot input records based on CT Skill dimensions.

Code	Description	Example Input
sequence	A series of step-by-step actions	"The first step is to find the largest number."
loop	Instructions involving repetition	"What is the loop in finding the largest number?"
condition	Making decisions based on conditions	"What do you add to 'if, then' in a program to find max?"
data	Use of variables or data lists	"What are the roles of variables?"
operator	Use of mathematical or relational operators	"Should I use '>' or '<' when comparing numbers?"
event	Occurring certain behaviors under a specific command	"Should I use a loop or conditional to control the number of comparisons?"
sorting	Arranging data in a specific order	"How do I sort these numbers from largest to smallest?"

#### 4.4.2. Instructional and scaffolding materials

This study implemented a unified instructional system composed of programming tasks, scaffolding resources, and embedded GAI agents to support students' CT development. All materials were designed to align with the students' cognitive level, programming fluency, and curricular background in fifth grade programming education. The instructional design ensured that each component contributed to both CT and metacognitive regulation.

##### (1) Programming Tasks and Learning Environment

The intervention was conducted in a Scratch-based VPE tailored for fifth grade students. Three progressively challenging programming tasks were designed to support the development of key CT competencies:

- Find the Maximum: Students iterated through a list to identify the maximum value.
- Selection Sort for Maximum: Students applied simple conditional and loop structures to find the maximum.
- Bubble Sort for Maximum: Students implemented nested conditional and loop structures to find the maximum.

These tasks were embedded in everyday contexts such as comparing heights or shopping prices to enhance engagement. Each task spanned approximately two weeks. See *Supplementary Materials Table S7* for full task descriptions. Before the formal intervention, all students completed two warm-up exercises to familiarize themselves with Scratch's interface and blocks.

##### (2) Scaffolding Design: AMS vs. PMS

To investigate the differential effects of metacognitive support, two distinct scaffolds were implemented.

AMS: Students in the adaptive scaffolding condition engaged with Xiaotongbao. Xiaotongbao monitored students' programming behaviors in real time and delivered context-sensitive prompts triggered by behavioral indicators. These prompts were dynamically aligned with metacognitive phases—e.g., suggesting decomposition when planning was lacking, or offering feedback when strategies failed. This adaptive support enabled responsive regulation aligned with individual learning trajectories. The example of Xiaotongbao interface and its real-time guidance prompts was shown in Fig. 3, representing the adaptive scaffolding experience for the AMS group.

PMS: Students in the planned scaffolding condition received paper-based metacognitive scaffolding sheets designed by the instructor and research team. These PMSs corresponded to each programming task and included guiding prompts covering full metacognitive phases—planning, monitoring, and adjusting. For example, the PMS for the bubble sort task prompted students to identify sorting strategies, predict possible bugs, and evaluate solution efficiency. Although not graded, the worksheets contributed to formative assessment and encouraged reflective habits. An example of the PMS used for the Bubble Sort task was presented in Fig. 4, illustrating how students structured their solutions.

In order to compare the different types of metacognitive support more intuitively, we compared the different forms of metacognitive strategies obtained by students in the AMS and PMS groups under the same monitoring and adjusting guidance. See *Supplementary Materials Figs. S1–S2* of the detailed comparison.

##### (3) Shared Support: Conbot.

To ensure instructional consistency across all groups, all students, including those in the PMS and control conditions, had access to Conbot. Conbot served as a Q & A assistant, supporting students with programming knowledge, and Scratch block functions, and task clarifications upon request. Unlike Xiaotongbao, Conbot did not provide metacognitive prompts and did not intervene automatically based on behavior. See *Supplementary Materials Fig. S3* for screenshots of Conbots representative interaction records.

#### 4.5. Procedure

The study was conducted over a 13-week period with three groups. All students completed the same programming tasks and followed the same instructional pacing but received different forms of scaffolding support. The procedure was divided into four phases.

Phase 1: Baseline Assessment (Weeks 1–2)

Students completed baseline assessments, including CTt and Bebras tests to measure initial CT. After the pretests, students were

**Planning**

小瞳宝-冒泡排序  
现在请你按照从小到大方式对初始数组 [34,12,45,27] 进行冒泡排序。

我先问你，这个排序任务可以分解成几个子任务呀？也就是需要比较多少轮？

RootUser\_2105208090  
4轮  
Xiaotongbao asked students how many rounds it would take to bubble-sort an unordered array to guide them in breaking down the task into subtasks.

小瞳宝-冒泡排序  
不太对哦～冒泡排序呢，简单来说就是比较相邻的元素，如果顺序错误就吧它们交换过来，一趟一趟地比较，直到整个数组都排好序。每一趟都会把当前未排序部分的最大（或最小）元素“浮”到正确的位置。像这个数组，只需要比较 3 轮就行啦。你可以看看这个视频帮助理解哦：  
<https://www.bilibili.com/video/BV1f34y1s7ao> 那现在再想想，需要比较多少轮呢？

RootUser\_2105208090  
3轮  
Xiaotongbao gave feedback on the students' incorrect answers, provided the video address of the relevant concepts, and asked the students the above questions again to make sure that they had fully grasped the section.

小瞳宝-冒泡排序  
答对啦，真棒！接下来进入新环节，第一轮排序结果是什么呀？

RootUser\_2105208090  
12,34,45,27

**Monitoring**

小瞳宝-冒泡排序  
这一轮的结果不太对哦。第一轮排序是要把最大的数放到数组的最后一一位。先比较 34 和 12，12 小，所以 34 和 12 交换位置，得到 12,34；接着比较 34 和 45，34 小，位置不变；再比较 45 和 27，27 小，45 和 27 交换位置，第一轮结束，结果应该是 12,34,27,45。你是否需要冒泡排序过程图来帮助理解呢？

RootUser\_2105208090  
冒泡排序过程图  
After giving students feedback on their incorrect answers of the results of the first round of bubble sort, Xiaotongbao explained the process of bubbling sort in text and a mind map.

小瞳宝-冒泡排序  
冒泡排序的示例  
原始数据：初始数据为[34,12,45,27]  
第一步排序：第一步比较第一个和第二个数字，34大于12，交换位置，数组变为[12,34,45,27]  
第二步排序：第二步比较第二个和第三个数字，45大于34，位置不变，数组仍为[12,34,45,27]  
第三步排序：第三步比较第三个和第四个数字，45大于27，交换位置，数组变为[12,34,27,45]  
第四步排序：第四步比较第一个和第二个数字，12小于34，位置不变，数组仍为[12,34,27,45]  
第五步排序：第五步比较第二个和第三个数字，34大于27，交换位置，数组变为[12,27,34,45]  
第六步排序：第六步比较第三个和第四个数字，34大于45，交换位置，数组变为[12,27,45,34]  
最终排序结果：最终数组为[12,27,45,34]

a. An example for planning and monitoring strategies

**Adjusting**

RootUser\_2105208090  
理解了

小瞳宝-冒泡排序  
好呀，那进入新环节。你的 scratch 程序实现了目标吗？

RootUser\_2105208090  
还没有  
Xiaotongbao directed students' attention to errors in the algorithmic codes. After providing students with several common errors, modifications and hints are given for students' individualized errors.

小瞳宝-冒泡排序  
请你想思考从哪一步调整？记得检查内循环和外循环的条件、语句哦，建议你根据运行的错误进行修改。

RootUser\_2105208090  
我外循环的条件不对，应该是什么

小瞳宝-冒泡排序  
外循环的条件通常和数组元素的个数有关哦。对于 4 个元素的数组，外循环要比较轮数是元素个数减 1，也就是 3 次。你想想怎么调整外循环条件能保证循环 3 次呀。

RootUser\_2105208090  
明白了，数组长度减1

小瞳宝-冒泡排序  
很棒！那接下来到评估环节啦，你今天的学习感受如何？选一个表情来代表吧：A 😊（非常开心）、B 😃（比较开心）、C 😐（一般）、D 😞（不开心）

RootUser\_2105208090  
非常开心  
Xiaotongbao prompted students to evaluate learning experience and encouraged the transfer of knowledge to real-world problems.

小瞳宝-冒泡排序  
听到你学得开心真好！现在进入反思环节啦，你觉得今天学习的冒泡排序内容如何与生活联系起来？

RootUser\_2105208090  
可以帮助我们在体育课排列队伍

小瞳宝-冒泡排序 06-06 16:32  
这个想法很不错💡 比如按身高从低到高排队就可以用到冒泡排序的思路。那你再思考一下，在生活里还有没有其他地方也能用到类似冒泡排序这种比较和交换的方法呢😊

b. An example for adjusting strategies

Fig. 3. An example of the AMS for bubble sorting.

randomly assigned to one of the three groups. Stratified sampling was not necessary, as students had no prior experience with Scratch programming or metacognitive scaffolding.

#### Phase 2: Training (Weeks 3–4)

All students received structured training on how to use the Scratch and Conbot. Those in AMS and PMS groups were also trained to interact with Xiaotongbao or structured scaffolding worksheets. This phase ensured that all students became familiar with the tools and formats used in their respective conditions.

**冒泡排序**

冒泡排序就是在一组无序的数据中，从前面开始，**比较相邻两个数的大小**，再根据比较结果**交换**两个数的位置，如此**循环**，直到把整组数从小到大排列**完成**。

**Planning**

78 按照从小到大的方式进行冒泡排序。

①一、规划目标  
你觉得这个冒泡排序任务需要比较**3**轮？（填空题）

②二、排序过程  
1. 排序过程  
按轮次记录（示例填写第一轮）

轮数	比较对象	是否交换	当前序列	发现规律
1	65 <b>□</b> 90	<input checked="" type="checkbox"/>	65, 90, 35, 78	65<90, 不交换
1	90 <b>□</b> 35	<input checked="" type="checkbox"/>	65, 35, 90, 78	90>35, 交换
1	90 <b>□</b> 78	<input checked="" type="checkbox"/>	65, 35, 78, 90	90>78, 交换

2. 请你写出每一轮的排序结果：  
第一轮：65, 35, 78, 90  
第二轮：35, 65, 78, 90  
第三轮：35, 65, 78, 90  
你的每一轮排序结果正确吗？正确。如果不正确，请你思考下冒泡排序的过程。

③常见问题工具箱

问题类型	解决策略
漏掉比较步骤	用荧光笔标记当前比较位置
交换次数过多	检查是否重复交换相同数字

**Monitoring**

The planned scaffold prompted the correct scratch code.

The planned scaffold asked students how many rounds it would take to bubble-sort an unordered array to guide them in breaking down the task into subtasks.

The planned scaffold demonstrated the first round of sorting. Then, it asked students about the results of each round of bubble sort and guided students to focus on the correct answers.

The planned scaffold provided several solutions to common errors.

The planned scaffold asked students whether the scratch program achieved the goal. If not, it guided students to adjust their solutions.

**Adjusting**

The planned scaffold prompted the correct scratch code.

The planned scaffold asked students how many rounds it would take to bubble-sort an unordered array to guide them in breaking down the task into subtasks.

The planned scaffold demonstrated the first round of sorting. Then, it asked students about the results of each round of bubble sort and guided students to focus on the correct answers.

The planned scaffold prompted students to evaluate learning experience and encouraged the transfer of knowledge to real-world problems.

**四、评估与反思**

1. 你今天的学习感受如何？C

A B C D

2. 你觉得今天学习的内容如何与生活联系起来？  
我觉得冒泡排序还是非常有用的，可以帮我们在购物整理放按照大小摆放书籍，联系生活，也可以在生活中通过排序以解决排序问题，但是有点困难。

Fig. 4. Example of the PMS worksheet for the bubble sort task. (Note: This worksheet guided students through task planning, monitoring, and adjusting during the algorithm.)

#### Phase 3: Intervention (Weeks 5–10)

Over six weeks, students completed three core programming tasks—Find the Maximum, Selection Sort, and Bubble Sort—designed to elicit CT skill use and metacognitive regulation behaviors. Based on the Gao et al. (2025), Fig. 5 illustrates the overall learning process design applied across three groups, highlighting key instructional phases and the integration of CT and metacognitive scaffolding.

- Knowledge Review. Each session began with the teacher reviewing key concepts from the previous lesson. For the AMS and PMS groups, the teacher also revisited exemplary applications of scaffolds used in prior sessions, highlighting both strengths and areas for improvement in students' responses.
- Learning New Concepts. The teacher introduced key concepts using vivid, real-world examples to facilitate understanding. Students were then encouraged to explore the core knowledge further through continuous, generative interaction with Conbot, engaging in iterative conversation until they demonstrated adequate comprehension of the targeted knowledge.
- Solving Computational Problems. During this phase, students applied their CT knowledge to complete programming tasks. Based on the same instruction context and different interventions among the three groups, all students implemented the algorithms developed during problem-solving in the Scratch environment.
  - The AMS group received adaptive, real-time metacognitive prompts from Xiaotongbao.
  - The PMS group received Conbot and paper-based scaffolding worksheets that guided students through the metacognitive phases.
  - The control group received Conbot and basic task instructions without additional metacognitive scaffolding.
- Sharing Solutions. In the final stage of each session, students shared their algorithmic solutions with peers and received the common instruction from the instructor.

Data collected during this phase included:

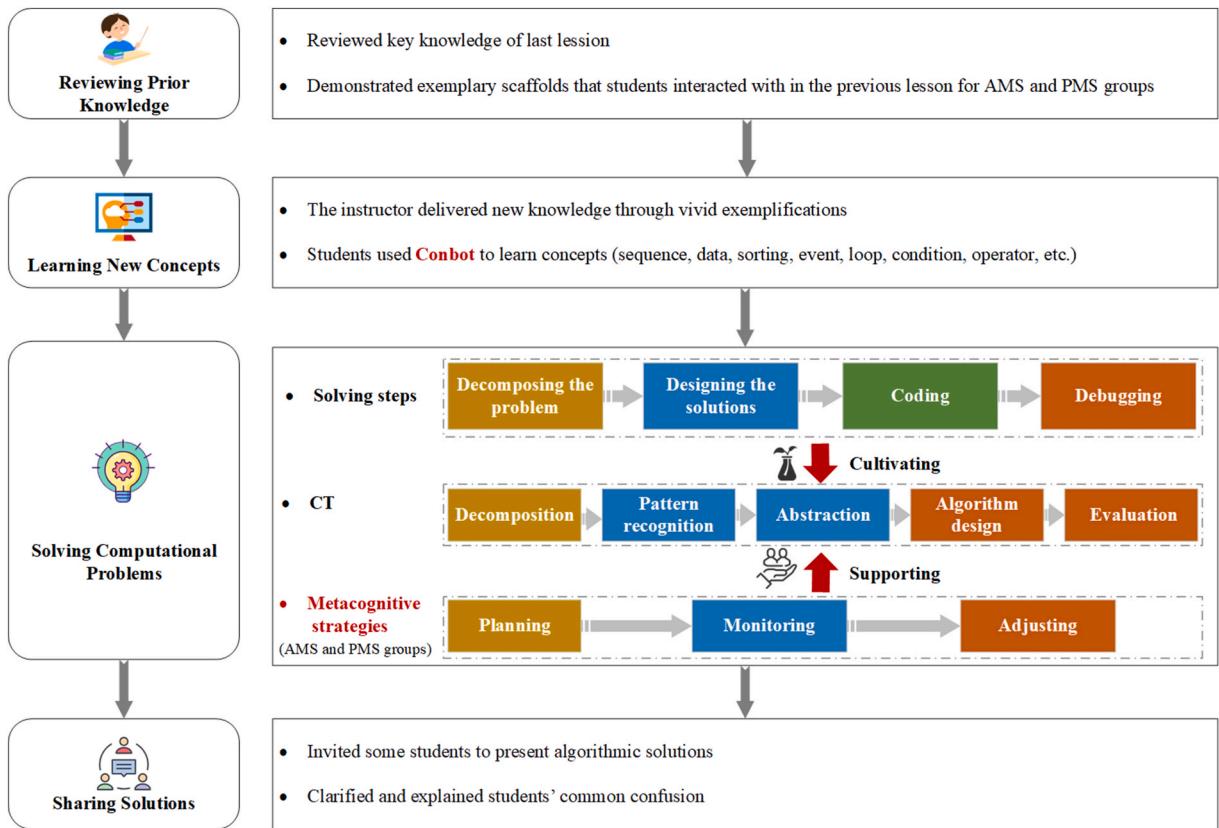


Fig. 5. Learning process of each task during experimental intervention.

- Conbot interaction logs and Scratch code files.
- Classroom observations and worksheet entries.

#### Phase 4: Posttest and Final Assessment (Weeks 11–13)

In Week 11, students completed a final programming project. In Weeks 12–13, students took the posttest versions of the CTt and Bebras tests. They also completed a cognitive load questionnaire. The final programming project artifacts, screen recordings, classroom observations, reflection logs, and subjective report data were collected, which served as summative outcome measures for assessing improvements in CT assessments across the three groups.

#### 4.6. Data analysis

All data were analyzed following a sequential procedure based on data characteristics and research aims. The analyses included parametric tests, nonparametric tests, behavioral coding, and network-based modeling. Statistical analyses were conducted using SPSS 27.0, and ENA was carried out using the web-based platform.

First, to determine the appropriateness of parametric tests, the normality of continuous variables was assessed using the Shapiro-Wilk test, and homogeneity of variance was tested using Levene's test. When assumptions of normality and homogeneity were met, one-way analysis of covariance (ANCOVA) was used to examine group differences in students' performance on computational tasks, with pre-test scores entered as covariates. Effect sizes were reported using partial eta-squared ( $\eta_p^2$ ), with thresholds of 0.01 (small), 0.06 (medium), and 0.14 (large) as suggested by Field (2013). Bonferroni correction was applied in post hoc tests to control for multiple comparisons. An analysis of variance (ANOVA) and post tests were conducted on the students' self-reported cognitive load. For variables that violated the assumption of normality, such as scores on the final open-ended programming project, the Kruskal-Wallis test was employed. Pairwise group differences were examined using Dunn's tests with Bonferroni-adjusted *p*-values (Conover, 1999).

Next, ENA was further conducted on students' interactions with the Conbot. Students' utterances were segmented into overlapping discourse windows, and co-occurrence matrices of CT skills were generated. We visualized the structural density and clustering of co-occurring behaviors (Shaffer et al., 2016), allowing examination of how different scaffolding types influenced students integrated patterns.

Behavioral data on students' use of CT skills and metacognitive regulation behaviors were obtained through observation and video coding. Two trained raters applied a structured coding framework covering five CT skill dimensions, as well as three metacognitive

behavior categories. Inter-rater reliability was established, with Cohen's  $\kappa$  exceeding 0.80 across all categories (McHugh, 2012). The frequencies of coded behaviors were compared between groups using Kruskal-Wallis tests or Mann-Whitney  $U$  tests, as appropriate. Temporal distributions were also analyzed to capture behavioral shifts across task stages.

## 5. Results

This section presents the results of data analyses conducted to examine the effects of adaptive and planned scaffolding on elementary students' CT. The findings are reported according to the three target constructs identified in this study: CT task performance, CT skill use, and metacognitive regulation. These constructs were operationalized through a combination of standardized tests, behavioral coding, and network-based discourse analysis.

### 5.1. Performance on CT tasks

This section reports the effects of scaffolding conditions on students' CT task performance, assessed through both standardized instruments and open-ended programming tasks. Group-level comparisons are presented for pre/posttest gains and final project scores.

#### 5.1.1. CT task scores: standardized tests of CT

To assess baseline comparability, ANOVAs were conducted on the pretest scores of the standardized CTt and the Bebras test. Results revealed no significant group differences: CTt,  $F_{(2, 71)} = 0.218, p = 0.805$ ; Bebras,  $F_{(2, 71)} = 0.064, p = 0.938$ . Shapiro-Wilk and Levene's tests confirmed normality and homogeneity of variances across groups ( $p > 0.05$ ), indicating statistical equivalence at baseline.

Paired-samples  $t$ -tests were used to examine within-group gains (see Table 3). On the CTt, only the control group showed a statistically significant improvement,  $t_{(19)} = 2.401, p < 0.05$ . While the AMS and PMS groups demonstrated positive trends, their gains were not statistically significant. On the Bebras test, however, both AMS and PMS groups showed significant improvements: AMS,  $t_{(27)} = 7.146, p < 0.001$ , PMS,  $t_{(25)} = 3.126, p < 0.01$ . The control group's gain was marginal,  $t_{(19)} = 2.053, p = 0.054$ .

To control for baseline variation, ANCOVAs were conducted using pretest scores as covariates. No significant group differences were found on the posttest of the CTt,  $F_{(2, 70)} = 0.980, p = 0.381, \eta_p^2 = 0.028$ . However, significant differences were found on the Bebras posttest,  $F_{(2, 70)} = 10.298, p < 0.001, \eta_p^2 = 0.232$ . Post hoc comparisons revealed that the AMS group significantly outperformed both the PMS group (adjusted  $p < 0.05$ ) and the control group (adjusted  $p < 0.001$ ).

The between-group difference on the CTt did not reach statistical significance ( $p = 0.070$ ). This divergence from the Bebras pattern is consistent with the nature of the two assessments: CTt targets abstract, decontextualized constructs, whereas Bebras items are contextualized and problem-based, closely mirroring the learning activities and the moments at which AMS intervenes. As a result, AMS is more likely to yield near-transfer gains on contextualized tasks within a brief intervention, while improvements on abstract conceptual knowledge may require longer consolidation or additional, concept-focused activities. Given the sample size and duration, the study was better powered to detect medium effects on contextualized performance than small effects on abstract CT. We therefore interpret CTt and Bebras as complementary indicators rather than conflicting results.

#### 5.1.2. CT task performance in open-ended programming tasks

Students' CT task performance was further assessed using their final programming projects. A Kruskal-Wallis test showed significant group differences for both goal attainment ( $\chi^2_{(2)} = 11.936, p < 0.01$ ) and program size ( $\chi^2_{(2)} = 14.471, p < 0.01$ ).

Dunn's post hoc tests (Bonferroni-adjusted, see Table 4) revealed that the AMS group significantly outperformed both PMS and control groups on goal attainment (vs. PMS: adjusted  $p < 0.05$ ; vs. control: adjusted  $p < 0.01$ ). Similarly, AMS outperformed both groups on program size (vs. PMS: adjusted  $p < 0.05$ ; vs. control: adjusted  $p < 0.01$ ). No significant differences were found between the PMS and control groups (see Table 4).

Qualitative analysis of student programs further supported these findings. 78.6 % of AMS students (22 out of 28) incorporated at least one nested loop or conditional structure, compared to 53.8 % in PMS and 35.0 % in the control group. AMS students also showed more frequent integration of multiple CT structures, such as conditionals and loops, suggesting more advanced and modularized problem-solving.

Overall, these results indicated that AMS facilitated higher-quality programming outputs. The AMS group not only completed more tasks but also constructed more structurally complex and flexible solutions. Their frequent use of nested control structures, loops, and

**Table 3**  
Paired-samples  $t$ -test results for Standardized CTt and Bebras.

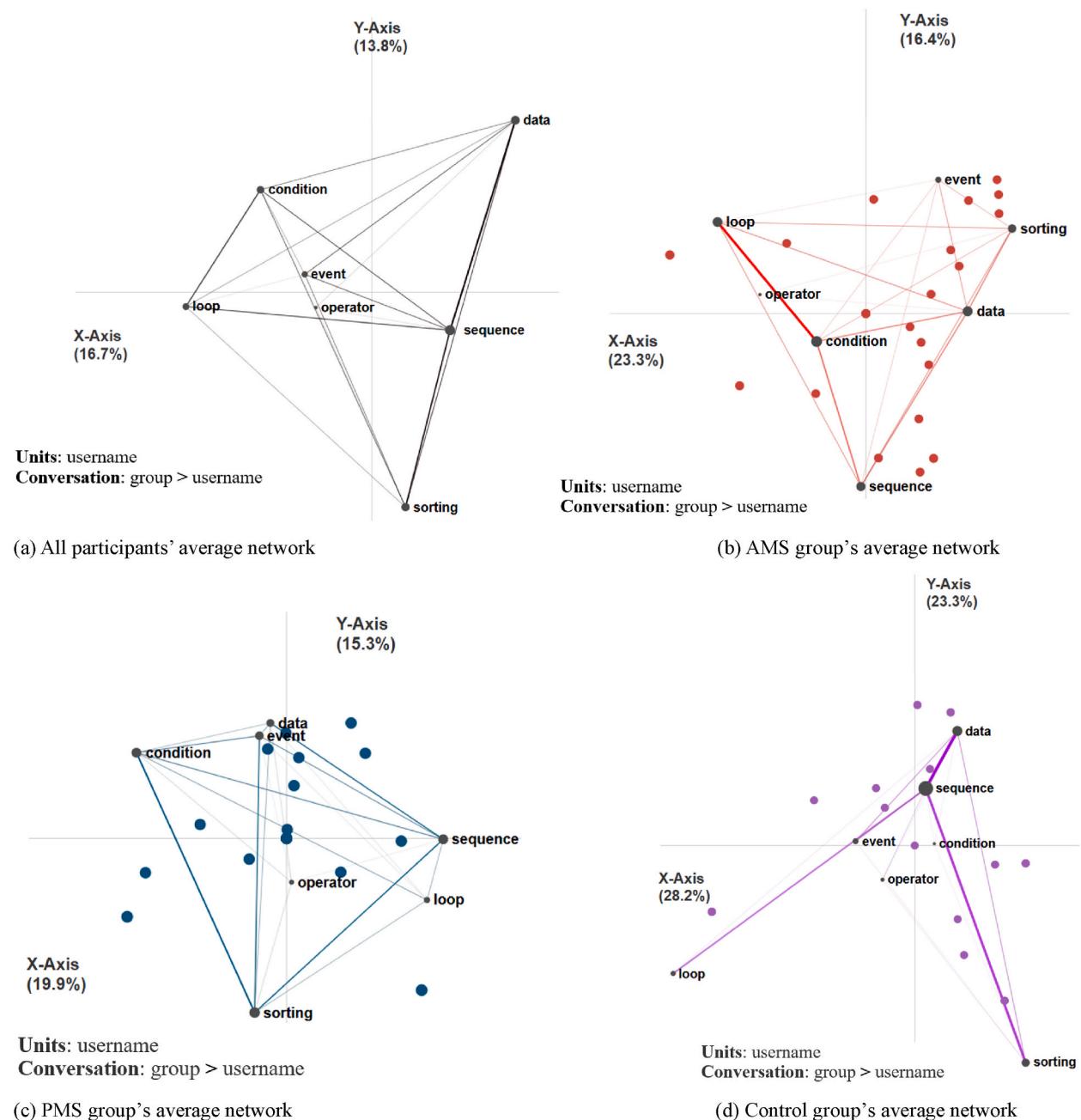
Measure	Group	Mean	SD	t	p
CTt	AMS	2.107	5.909	1.887	0.070
	PMS	1.500	3.881	1.971	0.060
	Control	2.250	4.191	2.401	0.027
Bebras Test	AMS	34.000	25.176	7.146	0.000
	PMS	15.115	24.658	3.126	0.004
	Control	7.350	16.007	2.053	0.054

**Table 4**

Post Hoc comparisons for programming task performance (Dunn's test).

Dimension	Group	Std. Z	p	Adjusted p
Goal attainment	AMS vs. PMS	2.688	0.007	0.022
	AMS vs. Control	3.144	0.002	0.005
	PMS vs. Control	0.634	0.526	1.000
Program size	AMS vs. PMS	2.426	0.015	0.046
	AMS vs. Control	3.706	0.000	0.001
	PMS vs. Control	1.427	0.154	0.461

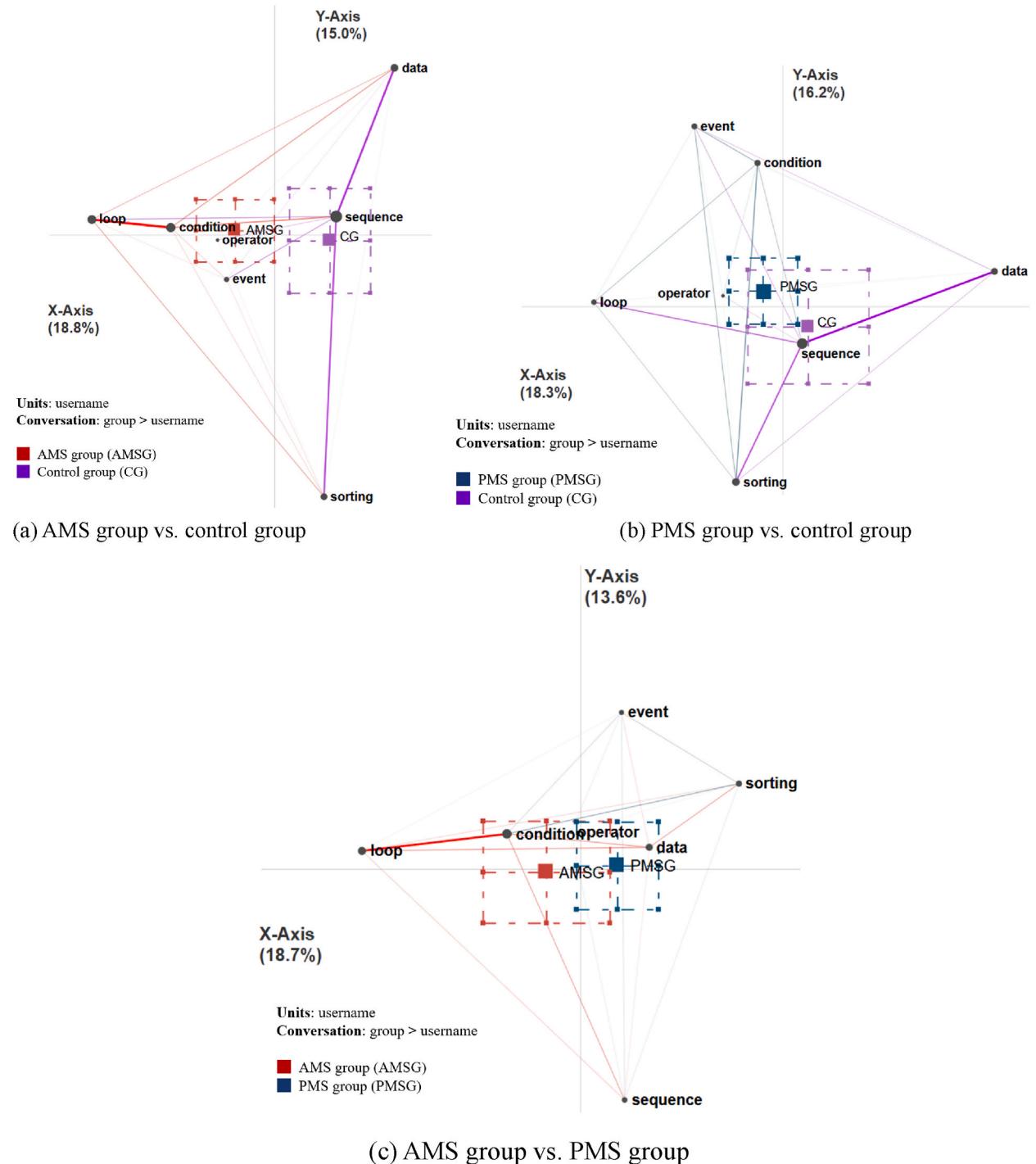
Note: Bonferroni correction applied.

**Fig. 6.** Average Discourse Networks of CT Skill Use. (Note: Nodes represent seven CT skills; edges indicate co-occurrence frequency within discourse windows. Thicker edges reflect stronger co-occurrence. X-axis = emphasis on complex control structures; Y-axis = variability in skill integration.)

conditionals reflected deeper engagement with CT skills in authentic, open-ended tasks.

### 5.2. Use of CT skills during problem solving

To explore students' enactment of CT skills during the intervention, we triangulated two data sources: GAI-mediated interaction logs and video-recorded programming sessions. The former was analyzed using ENA, while the latter were coded using a structured



**Fig. 7.** Group comparisons of CT skill networks in SVD-reduced space. (Note: X-axis = emphasis on complex control structures; Y-axis = variability in skill integration.)

behavioral rubric.

### 5.2.1. Discourse network patterns in Conbot interaction

To investigate how students used CT skills during interaction with Conbot, ENA was conducted on discourse data from all three groups. The ENA model visualizes the co-occurrence structure of seven coded CT skills—sequence, loop, condition, data, operator, event, and sorting—across overlapping discourse segments.

Group-averaged discourse networks (see Fig. 6) revealed that sequence consistently functioned as a central node across all conditions. The AMS group displayed denser connections between loop and condition, suggesting more integrated control structures. The PMS group showed clustering among sequence, sorting, and condition, while the control group exhibited simpler and sparser connections, mainly between sequence, data, and sorting.

Group centroid comparisons in the reduced semantic space further supported these differences (see Fig. 7). A significant separation was found between the AMS and control groups ( $U = 121.5, p < 0.01$ ) and between the PMS and control groups ( $U = 140.0, p < 0.05$ ) along the X-axis, indicating greater integration of complex CT skills in the AMS and PMS groups. No statistically significant differences were detected between AMS and PMS groups ( $U = 299.5, p = 0.26$ ).

These results indicate that only the AMS and PMS conditions produced a statistically distinct discourse structure, whereas the AMS group was characterized by stronger connections among higher-order CT skills (condition and loop).

### 5.2.2. Group differences in skill structures

To further examine students' use of CT skills during programming tasks, we conducted video-based coding of students' final projects. The rubric included five CT skill categories: decomposition, pattern recognition, abstraction, algorithm design, evaluation.

Given the non-normal distribution of skill frequency data, Kruskal-Wallis tests were used to compare group differences (see Table 5). Significant differences were found for pattern recognition ( $\chi^2(2) = 13.654, p < 0.01$ ) and abstraction ( $\chi^2(2) = 6.367, p < 0.05$ ). No significant group effects were observed for decomposition, algorithm design, evaluation, or the overall CT skill index.

Post hoc Dunn's tests with Bonferroni correction showed that the AMS group significantly outperformed the control group in both pattern recognition (adjusted  $p < 0.01$ ) and abstraction (adjusted  $p < 0.05$ ). No significant differences were found between AMS and PMS groups or between PMS and control groups on any CT skills (see Fig. 8).

These findings suggest that AMS more effectively supported students' engagement with selected CT skills, especially pattern recognition and abstraction.

## 5.3. Metacognitive regulation behaviors

This section examines how students engaged in metacognitive regulation during problem solving. We analyzed metacognitive behaviors to capture three core regulatory processes: planning, monitoring, and adjusting. These behaviors were coded from classroom observation data, post-task reflection worksheets and scratch process logs.

### 5.3.1. Behavioral indicators of metacognitive regulation

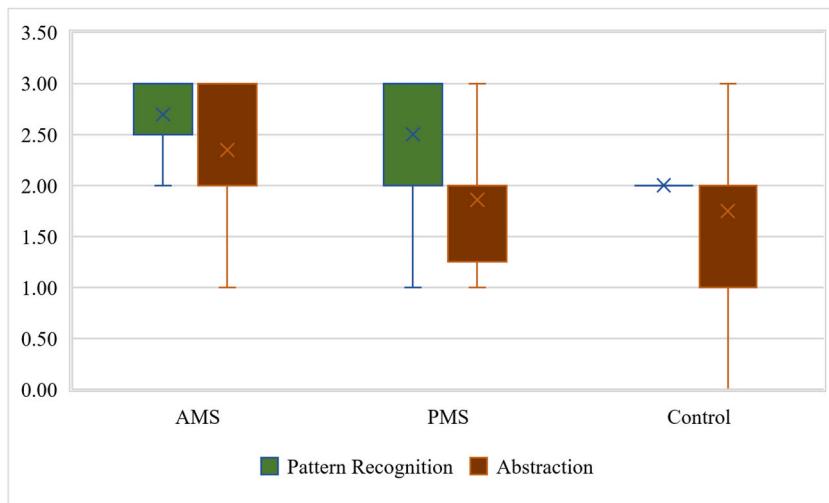
Students' metacognitive regulation behaviors varied notably across groups. Learners in the AMS group demonstrated the most frequent and diverse engagement in planning, monitoring, and adjusting, with some students exhibiting up to seven distinct episodes of each. In contrast, the control group showed limited metacognitive activity—particularly adjusting—while the PMS group displayed intermediate levels of frequencies and complexity.

In addition to individual behavior counts, we examined the presence of sequential patterns, including planning followed by monitoring ( $P \rightarrow M$ ), monitoring followed by adjusting ( $M \rightarrow A$ ), and full regulation cycles ( $P \rightarrow M \rightarrow A$ ). These sequential behaviors were more prevalent in the AMS group than in the PMS and control groups, suggesting that AMS was more effective in supporting the integration and chaining of metacognitive acts during complex tasks.

Representative cases from the AMS group illustrated the depth and coherence of metacognitive regulation observed. Student E1 completed four full  $P \rightarrow M \rightarrow A$  cycles during one programming task, demonstrating the ability to strategically plan, detect execution issues, and respond with targeted adjustments. Student E3 engaged in multiple  $M \rightarrow A$  sequences, identifying errors in real time and iteratively refining algorithmic structures based on those insights. Student E4 exhibited stable regulation routines across multiple subtasks, consistently applying planning, monitoring, and adjusting in succession, with observable improvements in task structure.

**Table 5**  
Group differences in CT skill use (Kruskal-Wallis test results).

Skill	AMS	PMS	Control	$\chi^2$	$p$
	Mean $\pm$ SD				
Decomposition	5.478 $\pm$ 1.377	5.500 $\pm$ 1.401	4.450 $\pm$ 2.012	5.015	0.081
Pattern Recognition	2.696 $\pm$ 0.559	2.500 $\pm$ 0.650	2.000 $\pm$ 0.649	13.654	0.001
Abstraction	2.348 $\pm$ 0.775	1.857 $\pm$ 0.864	1.750 $\pm$ 0.786	6.367	0.041
Algorithm Design	3.913 $\pm$ 2.234	3.357 $\pm$ 2.274	3.250 $\pm$ 2.468	1.114	0.573
Evaluation	2.739 $\pm$ 0.689	2.786 $\pm$ 0.579	2.250 $\pm$ 0.967	5.194	0.074
Overall CT	3.435 $\pm$ 0.906	3.200 $\pm$ 0.941	2.740 $\pm$ 1.230	3.762	0.152



**Fig. 8.** Distribution of pattern recognition and abstraction skill frequencies. (Note: The AMS group demonstrated higher medians and broader distributions. Outliers are marked individually.)

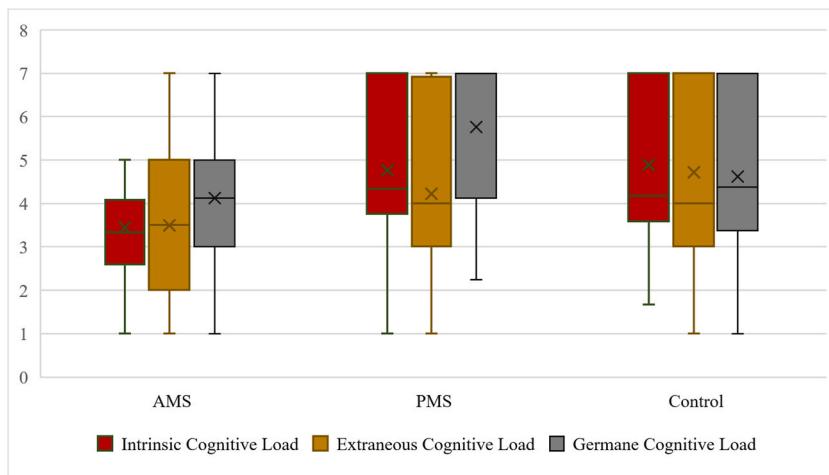
The PMS group focused on the P→M strategies, and although they also attempted the M→A strategies, most attempts were ineffective. In contrast, most control group students demonstrated at most one instance of P→M and rarely engaged in adjusting behaviors.

These findings suggest that AMS not only increased the occurrence of metacognitive behaviors but also facilitated their meaningful combination into recursive, task-embedded regulatory structures. *Supplementary materials Table S5* provided representative behavioral paths observed in each group, highlighting contrasting regulatory profiles among AMS, PMS, and control students. These behavior sequences illustrated how regulation was structurally embedded and dynamically deployed in AMS-supported learning.

### 5.3.2. Cognitive load as a proxy for metacognitive demands

To further examine the cognitive demands associated with different scaffolding conditions, we analyzed students' self-reported cognitive load across three dimensions: ICL, ECL, GCL. These dimensions served as indicators of mental effort, instructional clarity, and constructive processing, respectively.

A one-way ANOVA revealed significant group differences in both ICL and GCL (see Fig. 9). Students in the AMS group reported the lowest levels of ICL (*Mean* = 3.440, *SD* = 1.605,  $F(2, 71) = 4.840, p < 0.05$ ), significantly lower than those in the PMS (*Mean* = 4.756, *SD* = 2.130) and control (*Mean* = 4.884, *SD* = 1.755) groups. This suggests that AMS effectively reduced the complexity and cognitive strain of task engagement. Although no statistically significant differences were observed in ECL ( $F(2, 71) = 2.399, p = 0.098$ ), the AMS group reported descriptively lower levels compared to the PMS and control groups. For GCL, students in the PMS group reported significantly higher scores (*Mean* = 5.760, *SD* = 1.624,  $F(2, 71) = 6.060, p < 0.01$ ) than those in the AMS (*Mean* = 4.125, *SD* = 1.745) group. This suggests that planned scaffolding may have been more effective in promoting productive mental effort and reflective



**Fig. 9.** The results for the different groups of cognitive loads.

processing during and after task completion.

Taken together, these results highlight complementary effects of the two scaffolding approaches. AMS was particularly effective in reducing students' ICL during problem solving, allowing for more fluid and responsive regulatory engagement. PMS, on the other hand, with the highest GCL, encouraged deeper reflection and constructive schema development through structured self-questioning and retrospective analysis.

## 6. Discussion

Anchored in H1–H3, this discussion contrasts AMS and PMS in a GAI-supported VPE across three outcome families—CT task performance, CT skill use, and metacognitive regulation (Table 6). In brief, the patterns align with our hypotheses: consistent with H1, AMS outperformed PMS on contextualized CT tasks (Bebas; open-ended goal attainment and program size) while showing no reliable advantage on the abstract CTt; consistent with H3, AMS reduced ICL during problem solving whereas PMS increased GCL via structured reflection; evidence for H2 on skill integration was directionally favorable to AMS yet non-significant. These results indicate complementary strengths—AMS for in-task regulation and novice support, PMS for post-task consolidation and learners with stronger self-regulation—which we elaborate below.

### 6.1. CT task performance: enhancing task completion and structural complexity through adaptive scaffolding

The findings of Bebras tests indicate that AMS, supported by a GAI agent, significantly enhanced students' CT task performance compared to PMS and control conditions. This advantage was observed across both standardized assessments and open-ended programming tasks: AMS students achieved higher completion rates and produced structurally more complex code. These results align with prior research showing that real-time, individualized support is particularly effective in guiding novice learners through ill-structured challenges (Liao et al., 2024; Wu et al., 2025).

One likely explanation lies in the contingent nature of adaptive scaffolding. By delivering immediate, context-sensitive prompts in response to students' ongoing actions, the GAI agent provided just-in-time regulation aligned with learners' cognitive states. This echoes the literature on contingent scaffolding, which emphasizes responsiveness as a key mechanism for maintaining engagement and reducing task-related cognitive overload (Chen et al., 2025; Rahman et al., 2025; Song et al., 2025). In contrast, PMS offered static and pre-scripted prompts, while potentially helpful for initial planning, may have lacked the flexibility needed for in-process guidance during execution (Lim et al., 2024; Zheng et al., 2023).

Programming artifacts from the AMS group revealed more frequent use of nested loops, conditionals, and iterative refinement. These patterns suggest not only enhanced task performance but also a deeper application of structural CT skills. Such findings support work by Wang et al. (2023, 2024), who underscore the importance of metacognitive support in fostering recursive reasoning and structural planning. Although AMS consistently outperformed the control group, the performance gap between AMS and PMS was narrower, suggesting that well-designed planned scaffolding retains pedagogical value, particularly for supporting task orientation and reflection.

Notably, AMS's advantages were strongest on contextualized tasks (Bebas; open-ended goal attainment and solution complexity), whereas CTt showed no reliable advantage. This pattern aligns with the mechanism of AMS: behavior-triggered, just-in-time prompts restore planning, monitoring, and adjusting during task execution, which supports applied reasoning and structural completeness. By contrast, abstract CT concepts assessed by CTt may change more slowly and benefit from post-task consolidation—a role that our data associate with PMS through increased GCL. We therefore frame AMS and PMS as complementary: AMS for in-task regulation and near-transfer performance, PMS for strengthening conceptual consolidation over time. Future work might explore hybrid designs that combine PMS's structure with AMS's real-time adaptability to serve learners with varying support needs.

### 6.2. CT Skills Use: promoting complex and diverse strategies through adaptive scaffolding

Beyond overall performance, the results demonstrate substantial differences in students' use of CT skills across the three conditions. Analyses of GAI-mediated discourse and video-based programming behaviors consistently revealed that AMS students not only

**Table 6**

Summary of AMS vs. PMS across outcome measures.

Outcome family	Measure	Direction (AMS vs. PMS)	Sig.
CT task performance	CTt (standardized, abstract)	AMS ≥ PMS	ns
	Bebas (contextualized)	AMS > PMS	*
	Open-ended: goal attainment	AMS > PMS	*
	Open-ended: program size	AMS > PMS	*
CT skill use	Interaction traces (Combot)	AMS ≥ PMS	ns
	Programming artifacts (structural complexity)	AMS ≥ PMS	ns
Metacognitive regulation and cognitive load	ICL (Intrinsic cognitive load)	AMS < PMS	*
	ECL (Extraneous cognitive load)	AMS < PMS	ns
	GCL (Germane cognitive load)	PMS > AMS	**

Notes. ns =  $p \geq 0.05$ ; \* $p < 0.05$ ; \*\* $p < 0.01$ . For ICL/ECL, lower is desirable; for GCL, higher is desirable.

employed CT skills more frequently but also displayed more complex and integrated patterns of skill use than those in PMS and control groups.

ENA visualizations showed that sequence was the most common skill across all conditions. However, only the AMS and PMS groups demonstrated rich co-occurrence patterns involving higher-order skills such as loop, condition. Consistent with the research findings of Huang et al. (2024), metacognitive scaffolds were beneficial to students' complex CT skills. The AMS group, in particular, exhibited significantly denser network structures, with centroid separation indicating greater skill integration compared to the PMS and control groups. These findings suggested that it was not the GAI interface itself, but the presence of metacognitive scaffolding—especially adaptive support—that enabled learners to organize and deploy CT skills in more strategic combinations. This conclusion once again validates the finding of Wu et al. (2025) that adaptive metacognitive strategies strengthened students' thinking development.

This interpretation was reinforced by video analyses, which showed that AMS students outperformed their peers in pattern recognition and abstraction—two CT skills associated with flexible problem representation and solution generation (Gao et al., 2025; Tang et al., 2020). Unlike PMS learners, who tended to follow more linear execution paths, AMS learners frequently engaged in recursive behavior cycles involving planning, revision, and reimplementation, pointing to greater metacognitive awareness and skill fluency. In particular, the adaptive prompts delivered during execution appeared to facilitate transitions between different CT skills, helping learners to shift between planning and action more fluidly (Hijón-Neira et al., 2024).

Taken together, these findings suggest that adaptive scaffolding fosters not only higher frequencies but also greater quality in CT skill deployment. By responding to students' real-time needs, AMS helped activate complex reasoning patterns and iterative refinement. PMS, in contrast, was more effective in supporting task preparation but less capable of sustaining dynamic orchestration during programming. This supports prior work on real-time scaffolding (e.g., Chien et al., 2024; Rahman & Watanobe, 2023) and points to the importance of adaptivity in supporting both metacognitive flexibility and skill integration. Future studies should examine how combining structured prompts with embedded adaptive support might optimize CT skill development across learners.

### *6.3. Metacognitive regulation: enhancing self-regulatory processes and managing cognitive load through adaptive scaffolding*

The analysis of students' metacognitive regulation behaviors and cognitive load profiles provided further evidence of the unique benefits of adaptive scaffolding in supporting self-regulated learning during programming-based CT tasks. Classroom observation-based behavior coding showed that students in the AMS group demonstrated significantly more frequent and complete metacognitive regulation episodes—including planning, monitoring, and adjusting—than those in the PMS and control groups. In particular, AMS learners frequently engaged in full P→M→A cycles, indicating not only goal-setting and progress tracking but also strategic revision of problem-solving approaches based on ongoing feedback (Liu et al., 2024).

These recursive behavior chains suggest that adaptive scaffolding fostered dynamic and context-sensitive regulation, allowing learners to internalize self-directed routines (Sivasakthi & Meenakshi, 2025; Yilmaz & Yilmaz, 2023; Zheng et al., 2023). In contrast, PMS students often followed linear sequences (e.g., P→M) without further adjustment, reflecting lack of more reactive and prompted regulation pattern (Wang et al., 2023). Similarly, students in the control group, who received no structured metacognitive support, rarely engaged in monitoring or adjustment, relying instead on surface-level approaches or trial-and-error attempts.

These group differences were reinforced by students' self-reported cognitive load. The AMS group reported significantly lower levels of ICL than both PMS and control groups, suggesting that real-time scaffolding helped reduce the mental effort required to manage complex tasks (Chen et al., 2025; Huang et al., 2025). ECL was also lowest in the AMS group, though not statistically significant, while GCL was highest in the PMS group. This pattern suggests that while planned scaffolding encouraged productive engagement with reflective prompts, its lack of adaptability limited its responsiveness to learners' evolving challenges during execution (Wang et al., 2023).

Together, these findings demonstrate that AMS offers dual regulatory benefits: it activates core metacognitive processes while simultaneously optimizing students' cognitive resources. Whereas PMS promoted reflection at the planning level, its static nature may have constrained deeper in-task adjustments. These results support the idea that effective scaffolding must combine structure with flexibility—offering both advance organization and moment-by-moment responsiveness. In this study, AMS achieved this through its generative feedback mechanism, which dynamically supported recursive regulation and cognitive efficiency throughout the programming process. Such adaptive scaffolding hold promise for support not only CT learning, but also broader self-regulatory capacities in technology-enhanced learning environments.

### *6.4. Educational implications and innovations of the GAI-supported scaffolding environment*

Our GAI-supported environment makes the personalization theme concrete by operationalizing AMS as in-situ support. The innovation lies not in using an agent per se but in how the system couples learners' code and interaction traces with a small repertoire of metacognitive prompts. When learners stall or show low-effort acceptance of GAI output, the agent issues just-in-time questions that target planning, monitoring, and adjusting (e.g., probing a loop's effect or asking for justification of a conditional). Prompts are short, timed to the moment of need, and governed by a minimal-help rule, so they restore self-regulation without over-guidance. This design explains the observed reduction in ICL under AMS.

At the classroom level, the same mechanism enables personalization at scale. Each student receives individualized prompts while the teacher maintains whole-class orchestration. The prompt log doubles as a formative assessment stream: it shows who needed which kind of help, when, and in relation to what code state. In practice, teachers can use these logs to identify students exhibiting automation bias, to form targeted mini lessons on specific control structures, and to document evidence of growth over time.

A complementary role is played by PMS. Rather than intervening during execution, PMS structures brief post-task reflections—such as explaining why a condition works or generalizing the solution to different array sizes—which channel effort into GCL. Our results therefore suggest a practical policy: deploy AMS during task execution to sustain regulation and reduce ICL, and use PMS after the task to consolidate understanding and raise GCL. This AMS-in-task/PMS-post-task sequencing provides a simple recipe that teachers can apply without redesigning the entire lesson.

Finally, the environment incorporates guardrails that matter for equity and authenticity. The agent suppresses direct-answer patterns that short-circuit thinking, and defers to teacher rules to preserve autonomy. Because the prompts are content-agnostic and triggered by generic behaviors, the policy is transferable beyond block-based programming to text-based coding, data handling, and broader STEM inquiries where learners iteratively plan-monitor-adjust. In short, the novelty of our environment is a classroom-ready mechanism for real-time personalization with actionable analytics for teachers, while the instructional value lies in a clear, reusable sequencing of AMS and PMS that aligns support with learners' cognitive demands.

### 6.5. Limitations and future directions

Several limitations of the current study should be acknowledged. First, the relatively small and unequal sample sizes may have limited the statistical power of some analyses. Although nonparametric tests were applied to account for distributional concerns, subtle group-level effects may have gone undetected. Future studies should employ larger and more balanced samples to validate and expand upon these findings.

Second, the intervention spanned a relatively short instructional period. While immediate improvements in CT performance and metacognitive behavior were observed, it remains unclear whether these gains would persist over time or transfer to other learning contexts. Longitudinal studies and delayed posttests could offer insight into the durability and generalizability of adaptive scaffolding effects.

Third, the behavioral data—particularly video-coded regulation sequences—relied on manual coding and interpretation. Despite high inter-rater agreement, the potential for subjective bias remains. Incorporating real-time behavioral analytics such as log-based trace analysis, interaction mining, or physiological sensing may yield more granular and scalable measures in future work.

Fourth, both the adaptive and planned scaffolding designs were researcher-driven and teachers. While grounded in theory, these supports were not co-developed with learners. Future iterations could benefit from participatory design processes that incorporate learner voice or dynamically adjust support structures based on learner profiles or performance patterns.

Finally, the study focused specifically on CT learning in a block-based programming environment for elementary students. While appropriate for the target population, the findings may not fully extend to other CT contexts such as text-based coding, data science, or interdisciplinary STEM integration. Future work should examine how adaptive scaffolding principles translate across domains and developmental levels.

In sum, this study contributes to a growing body of research on adaptive learning environments by illustrating how GAI-supported scaffolding can enhance CT learning and metacognitive regulation among elementary students. Despite its limitations, the findings provide a strong foundation for further exploration of adaptive, responsive, and developmentally appropriate scaffolding systems.

## 7. Conclusion

This study investigated the differential effects of adaptive and planned metacognitive scaffolding on elementary students' development of CT in a GAI-supported VPE. AMS yielded significant advantages on contextualized tasks (Bebras; open-ended goal attainment and program complexity) and reduced ICL during problem solving, whereas PMS increased GCL through structured reflection. Trends on CT skill integration favored AMS but were not statistically significant.

Beyond outcomes, our GAI-supported environment contributes a practical mechanism for classroom-scale personalization: behavior-triggered, just-in-time metacognitive prompts for in-task regulation (AMS) combined with post-task structured reflection (PMS). For teachers, prompt logs provide formative analytics to target support, identify low-effort acceptance of GAI outputs, and document growth. As schools adopt GAI, the promise lies not in replacing instruction but in amplifying it—meeting learners at the moment of need while cultivating durable self-regulation.

Future work should examine longer interventions, broader CT contexts (e.g., text-based coding, data science, STEM), and learner-co-designed variants that tailor trigger policies to classroom goals. Scaling studies with larger, balanced samples and automated trace analytics can further validate the AMS-in-task/PMS-post-task policy as a general template for personalized CT instruction.

### CRediT authorship contribution statement

**Jinfang Liu:** Writing – original draft, Software, Methodology, Formal analysis, Data curation, Conceptualization. **Yi Zhang:** Writing – review & editing, Supervision, Project administration, Funding acquisition, Conceptualization. **Wei Li:** Writing – review & editing, Validation, Methodology. **Qiyun Wang:** Writing – review & editing. **Pingxiu Niu:** Software, Data curation. **Xue Zhang:** Software, Data curation.

### Consent to participate

The participants all agreed to take part in this study.

## Ethical statement

This study was approved by the Ethics Committee of Central China Normal University (CCNU-IRB-202405013b).

## Consent for publication

The publication of this study has been approved by all authors.

## Funding

This study was funded by the National Natural Science Foundation of China (Grant No. 72274076).

## Acknowledgments

We sincerely thank all the research assistants and participants who have contributed to this study.

## Appendix A. Supplementary data

Supplementary data to this article can be found online at <https://doi.org/10.1016/j.compedu.2025.105473>.

## Data availability

Data will be made available on request.

## References

- Aho, A. V. (2012). Computation and computational thinking. *The Computer Journal*, 55(7), 832–835. <https://doi.org/10.1093/comjnl/bxs074>
- Al-Fattal, A. (2025). You Do It, AI does It, you compare and reflect: Exploring reflective learning with generative AI in principles of marketing. *Journal of Marketing Education*. <https://doi.org/10.1177/027347532513566>
- Avcı, Ü. (2024). Students' GAI acceptance: Role of demographics, creative mindsets, anxiety, attitudes. *Journal of Computer Information Systems*, 1–15. <https://doi.org/10.1080/08874417.2024.2386545>
- Azevedo, R., Cromley, J. G., & Seibert, D. (2004). Does adaptive scaffolding facilitate students' ability to regulate their learning with hypermedia? *Contemporary Educational Psychology*, 29(3), 344–370. <https://doi.org/10.1016/j.cedpsych.2003.09.002>
- Ba, S., Zhan, Y., Huang, L., & Lu, G. (2025). Investigating the impact of ChatGPT-assisted feedback on the dynamics and outcomes of online inquiry-based discussion. *British Journal of Educational Technology*. <https://doi.org/10.1111/bjet.13605>
- Batni, B., & Junaini, S. N. (2025). Redefining computational thinking: Synergizing unplugged activities with block-based programming. *Education and Information Technologies*, 30(2), 2361–2388. <https://doi.org/10.1007/s10639-024-12869-8>
- Chang, D. H., Lin, M. P. C., Hajian, S., & Wang, Q. Q. (2023). Educational design principles of using AI chatbot that supports self-regulated learning in education: Goal setting, feedback, and personalization. *Sustainability*, 15(17), Article 12921. <https://doi.org/10.3390/su151712921>
- Chao, P. Y. (2016). Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. *Computers & Education*, 95, 202–215. <https://doi.org/10.1016/j.compedu.2016.01.010>
- Chen, J., Mokmin, N. A. M., & Qi, S. (2025). Generative AI-powered arts-based learning in middle school history: Impact on achievement, motivation, and cognitive load. *The Journal of Educational Research*, 1–13. <https://doi.org/10.1080/00220671.2025.2510395>
- Chien, C. C., Chan, H. Y., & Hou, H. T. (2024). Learning by playing with generative AI: Design and evaluation of a role-playing educational game with generative AI as scaffolding for instant feedback interaction. *Journal of Research on Technology in Education*, 1–20. <https://doi.org/10.1080/15391523.2024.2338085>
- Conover, W. J. (1999). *Practical nonparametric statistics* (3rd ed.). Wiley.
- Crompton, H., & Burke, D. (2023). Artificial intelligence in higher education: The state of the field. *International journal of educational technology in higher education*, 20(1), 22. <https://doi.org/10.1186/s41239-023-00392-8>
- Dorgnier, R., Mazerolle, M., Maquestiaux, F., & Picard, L. (2025). Enhancing primary school learning through growth mindset and memory strategy interventions. *The Journal of Educational Research*, 118(1), 37–50. <https://doi.org/10.1080/00220671.2024.2431695>
- Eberhart, J., Ingendahl, F., & Bryce, D. (2025). Are metacognition interventions in young children effective? Evidence from a series of meta-analyses. *Metacognition and Learning*, 20(1), 7. <https://doi.org/10.1007/s11409-024-09405-x>
- Fanchamps, N. L., Slangen, L., Specht, M., & Hennissen, P. (2021). The impact of SRA-programming on computational thinking in a visual oriented programming environment. *Education and Information Technologies*, 26, 6479–6498. <https://doi.org/10.1007/s10639-021-10578-0>
- Fang, J. W., Chen, J., Weng, Q. L., Tu, Y. F., Hwang, G. J., & Xia, Y. C. (2025). Effects of a GenAI-based debugging approach integrating the reflective strategy on senior high school students' learning performance and computational thinking. *Educational Technology & Society*, 28(3), 66–81. <https://www.jstor.org/stable/48827951>
- Fang, J. W., Guo, X. G., Meng, X. P., Hwang, G. J., & Tu, Y. F. (2025). Realization of parasocial interaction theory in STEAM education with an AI pedagogical agent: Insights from learning performance, collaboration tendency, and problem-solving tendency. *Interactive Learning Environments*, 1–24. <https://doi.org/10.1080/10494820.2025.2508326>
- Field, A. (2013). *Discovering statistics using IBM SPSS statistics* (4th ed.). Sage.
- Gani, M. O., Ayyasamy, R. K., Alhashmi, S. M., Alam, K. S., Sangodiah, A., Khaleduzzman, K., & Ponnusamy, C. (2025). Towards enhanced assessment question classification: A study using machine learning, deep learning, and generative AI. *Connection Science*, 37(1). <https://doi.org/10.1080/09540091.2024.2445249>. Article 2445249.
- Gao, X., Yang, Y., Du, Y., & Sun, D. (2025). Effect of mind mapping-based scaffolding on elementary students' computational thinking in block-based programming. *Journal of Educational Computing Research*, 63(1), 236–271. <https://doi.org/10.1177/07356331241303067>
- Grover, S. (2021). Teaching and assessing for transfer from block-to-text programming in middle school computer science. In C. Hohensee, & J. Lobato (Eds.), *Transfer of learning: Research in mathematics education* (pp. 251–276). Cham: Springer. [https://doi.org/10.1007/978-3-030-65632-4\\_11](https://doi.org/10.1007/978-3-030-65632-4_11)



- Tu, Y. F., & Hwang, G. J. (2024). University students' conceptions of ChatGPT-supported learning: A drawing and epistemic network analysis. *Interactive Learning Environments*, 32(10), 6790–6814. <https://doi.org/10.1080/10494820.2023.2286370>
- Urban, M., Déchteraenko, F., Lukavský, J., Hrabalová, V., Svacha, F., Brom, C., & Urban, K. (2024). ChatGPT improves creative problem-solving performance in university students: An experimental study. *Computers & Education*, 215, Article 105031. <https://doi.org/10.1016/j.compedu.2024.105031>
- van der Meij, J., & de Jong, T. (2011). The effects of directive self-explanation prompts to support active processing of multiple representations in a simulation-based learning environment. *Journal of Computer Assisted Learning*, 27(5), 411–423. <https://doi.org/10.1111/j.1365-2729.2011.00411.x>
- Wang, C. Y., Gao, B. L., & Chen, S. J. (2024). The effects of metacognitive scaffolding of project-based learning environments on students' metacognitive ability and computational thinking. *Education and Information Technologies*, 29(5), 5485–5508. <https://doi.org/10.1007/s10639-023-12022-x>
- Wang, T., Zheng, J., Tan, C., & Lajoie, S. P. (2023). Computer-based scaffoldings influence students' metacognitive monitoring and problem-solving efficiency in an intelligent tutoring system. *Journal of Computer Assisted Learning*, 39(5), 1652–1665. <https://doi.org/10.1111/jcal.12824>
- Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., et al. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology*, 25(1), 127–147. <https://doi.org/10.1007/s10956-015-9581-5>
- Weintrop, D., & Wilensky, U. (2019). Transitioning from introductory block-based and text-based environments to professional programming languages in high school computer science classrooms. *Computers & Education*, 142. <https://doi.org/10.1016/j.compedu.2019.103646>, Article 103646.
- Weng, X., Ye, H., Dai, Y., & Ng, O. (2024). Integrating artificial intelligence and computational thinking in educational contexts: A systematic review of instructional design and student learning outcomes. *Journal of Educational Computing Research*, 62(6), 1420–1450. <https://doi.org/10.1177/07356331241248686>
- Wing, J. (2014). Computational thinking benefits society. In *40th anniversary blog of social issues in computing, 2014*.
- Wong, G. K. (2024). Amplifying children's computational problem-solving skills: A hybrid-based design for programming education. *Education and Information Technologies*, 29(2), 1761–1793. <https://doi.org/10.1007/s10639-023-11880-9>
- Wong, G. K., Jian, S., & Cheung, H. Y. (2024). Engaging children in developing algorithmic thinking and debugging skills in primary schools: A mixed-methods multiple case study. *Education and Information Technologies*, 29(13), 16205–16254. <https://doi.org/10.1007/s10639-024-12448-x>
- Wu, J., Wang, J., Lei, S., Wu, F., & Gao, X. (2025). The impact of metacognitive scaffolding on deep learning in a GenAI-supported learning environment. *Interactive Learning Environments*, 1–18. <https://doi.org/10.1080/10494820.2025.2479162>
- Wu, R., & Yu, Z. (2024). Do AI chatbots improve students learning outcomes? Evidence from a meta-analysis. *British Journal of Educational Technology*, 55(1), 10–33. <https://doi.org/10.1111/bjet.13334>
- Xu, E., Wang, W., & Wang, Q. (2023). A meta-analysis of the effectiveness of programming teaching in promoting K-12 students' computational thinking. *Education and Information Technologies*, 28(6), 6619–6644. <https://doi.org/10.1007/s10639-022-11445-2>
- Yan, L., Martinez-Maldonado, R., Yan, L., Martinez-Maldonado, R., Jin, Y., Echeverria, V., Milesi, M., Fan, J., Zhao, L., Alfredo, R., Li, X., Gašević, D, Yan, L., Martinez-Maldonado, R., Jin, Y., Echeverria, V., Milesi, M., Fan, J., Zhao, L., Alfredo, R., Li, X., & Gašević, D., (2025). The effects of generative AI agents and scaffolding on enhancing students' comprehension of visual learning analytics. *Computers & Education*. doi:10.1016/j.compedu.2025.105322. Article 105322.
- Yilmaz, R., & Yilmaz, F. G. K. (2023). The effect of generative artificial intelligence (AI)-based tool use on students' computational thinking skills, programming self-efficacy and motivation. *Computers and Education: Artificial Intelligence*, 4. <https://doi.org/10.1016/j.caai.2023.100147>, Article 100147.
- Zheng, J., Lajoie, S. P., Wang, T., & Li, S. (2023). Supporting self-regulated learning in clinical problem-solving with a computer-based learning environment: the effectiveness of scaffolds. *Metacognition and Learning*, 18(3), 693–709. <https://doi.org/10.1007/s11409-023-09352-z>
- Zhou, Y., Chai, C. S., Li, X., Ma, C., Li, B., Yu, D., & Liang, J. C. (2023). Application of metacognitive planning scaffolding for the cultivation of computational thinking. *Journal of Educational Computing Research*, 61(6), 1123–1142. <https://doi.org/10.1177/07356331231160294>
- Zhou, R., Xie, C., He, X., Li, Y., Fan, Q., Yu, Y., & Yan, Z. (2024). Effect of different flow design approaches on undergraduates' computational thinking during pair programming. *Journal of Educational Computing Research*, 62(7), 1865–1895. <https://doi.org/10.1177/07356331241268474>