Vol. 9, No.1, 2026, pp. 874-884
DOI: https://doi.org/10.32698/02330

Contents lists available at Journal Global Econedu

Journal of Educational and Learning Studies
ISSN: 2655-2760 (Print) ISSN: 2655-2779 (Electronic)

Journal homepage: http://jurnal.globaleconedu.org/index.php/jels

AI-assisted adaptive problem-based learning in a programming course: its effects on students' problem-solving and debugging skills

Septriyan Anugrah*), Nadya Faranda, Meldi Ade Kurnia Yusri, Nofri Hendri
Department of Curriculum and Educational Technology, Faculty of Education, Universitas Negeri Padang, Padang, Indonesia

Article Information

Article history:

Received Aug 11th, 2026
Revised Aug 23rd, 2026
Accepted Sep 02nd, 2026

Keywords:

Generative artificial intelligence,
adaptive grouping,
problem-based learning,
problem solving,
debugging

ABSTRACT

This study examined differences in problem-solving and debugging skills between students who participated in AI-assisted adaptive Problem-Based Learning (PBL) and those who participated in the Case Method in a programming course. A quasi-experimental nonequivalent pretest-posttest control-group design involved 69 students from the Department of Curriculum and Educational Technology, Universitas Negeri Padang. The experimental group (n = 35) followed Arends' PBL with data-informed adaptive grouping and structured use of ChatGPT, Gemini, and Claude during the investigation phase, while the control group (n = 34) followed the Case Method without generative AI. Multivariate analysis of covariance (MANCOVA) tested the two posttest scores while controlling for both pretest scores. The results showed a significant multivariate difference between the instructional conditions, Pillai's Trace = 0.456, $F(2, 64) = 26.780$, $p < 0.001$, partial $\eta^2 = .456$. Follow-up tests also showed significant differences in problem solving (partial $\eta^2 = 0.335$) and debugging (partial $\eta^2 = 0.435$). Adjusted means were higher for the experimental group on both skills. The findings suggest that an instructional package integrating PBL, adaptive grouping, and structured AI assistance has the potential to support problem-solving and debugging skills. However, because the three components were implemented simultaneously, their individual effects cannot be separated.

Corresponding Author:

Septriyan Anugrah,
Universitas Negeri Padang,
Email: septriyan@fip.unp.ac.id

Introduction

Learning programming requires not only mastery of syntax but also the ability to design solutions and trace the causes of programs that do not work as intended. Problem solving and debugging are interrelated but have different focuses. Problem solving encompasses interpreting requirements, decomposing problems, designing algorithms, implementing solutions, and evaluating them comprehensively. Debugging, meanwhile, focuses on reproducing errors, tracing their sources, diagnosing them, applying corrections, and retesting. Previous studies have shown that novice programmers often have underdeveloped mental models of programs, rely on trial-and-error strategies, and struggle to identify the source of errors (Ahmadzadeh et al., 2007; Fitzgerald et al., 2008; Qian & Lehman, 2018; Whalley et al., 2023; Yang et al., 2024). Distinguishing these two skills is important

because students may be able to design an appropriate algorithm without necessarily being able to diagnose implementation errors, or vice versa.

These difficulties become more pronounced in classes whose students have diverse entry-level abilities and programming experience. Students in the Programming Languages course at the Department of Curriculum and Educational Technology, Universitas Negeri Padang, entered the course with varying levels of prior programming experience. The overall pretest means used in the adjusted analysis were 56.14 for problem solving and 54.67 for debugging on a 0-100 scale, indicating substantial room for improvement. Educational background was not used as a fixed marker of student ability; instructional adjustments were based primarily on pretest results and formative evidence collected during the learning process. The adaptive-learning literature emphasizes the importance of learner information, performance evidence, and implementation context in determining the support provided (Hooshyar et al., 2024; Kabudi et al., 2021; Mirata et al., 2020; Mirata & Bergamin, 2023; Normadhi et al., 2019; Plooy et al., 2024). In this study, adaptivity took the form of data-informed pedagogical adjustments to group composition, role assignment, scaffolding, and task challenge, rather than an automated platform or fixed grouping based on student categories.

Problem-Based Learning (PBL) was selected because problem-centered instruction aligns with the processes students undertake when completing programming tasks. The five phases of Arends' PBL provide a clear instructional sequence: orientation helps students understand the problem; organization structures roles and peer support; investigation facilitates algorithm development and error diagnosis; artifact development requires implementation and testing; and evaluation focuses attention on the quality of the problem-solving process (Arends, 2012). Reviews and meta-analyses indicate that PBL can support higher-order thinking and problem-solving skills when problem characteristics, facilitation, and assessment are aligned with learning objectives (Dochy et al., 2003; Liu et al., 2022; Manuaba et al., 2022; Yew & Goh, 2016). In programming contexts, PBL also provides an active-learning framework for studying algorithms and fundamental programming concepts (Aires et al., 2023). However, this study was not designed to test the contribution of each PBL phase separately.

Generative artificial intelligence can provide rapid assistance when students encounter difficulties, but its pedagogical value depends heavily on how it is integrated into instruction. Reviews of AI in higher education emphasize that learning design, the educator's role, and human oversight remain central to responsible AI use (Holmes et al., 2022; Hwang et al., 2020; Zawacki-Richter et al., 2019). In programming education, AI can help explain concepts, interpret error messages, support debugging, and provide code examples. However, its effects on student performance are not always consistent and remain influenced by students' entry-level abilities and how they verify AI outputs (Dickey et al., 2024; Groothuijsen et al., 2024; Kazemitabaar et al., 2023; Kohenvacs et al., 2025; Sun et al., 2024). The risks of inaccurate outputs, overreliance, and weak verification processes also require attention (Kasneci et al., 2023; Mogavi et al., 2024; Tlili et al., 2023). Therefore, ChatGPT, Gemini, and Claude were not treated as three separate interventions; instead, they were used under a common protocol for use, testing, documentation, and oral explanation.

The control class used the Case Method as an active comparator. This method also requires students to analyze cases, apply concepts, discuss alternatives, and make defensible decisions. However, Case Method cases generally provide more structured contexts and information than the open-ended problems used in PBL (Zhang et al., 2022). Learning outcomes, content, instructional duration, and posttest instruments were aligned across both classes. The experimental class differed in that it completed more open-ended PBL tasks, used adaptive grouping and support, and received structured access to generative AI during the investigation phase. Thus, the study compared two instructional packages rather than treating access versus no access to AI as the sole difference. This distinction is important because the research design does not permit the causal effects of PBL, adaptive grouping, and AI assistance to be separated.

The current state of research shows a shift from unrestricted chatbot use toward more structured integration of AI in programming education. Omeh et al. (2025), for example, integrated Gemini into PBL and reported improved programming skills while also showing that learning outcomes could vary with students' academic ability. Na Nongkhai et al. (2026) found that a programming-support system combining adaptive mechanisms and generative AI was more effective than adaptive support alone. Research on dynamic grouping has also shown that changes in group composition can influence collaborative processes, although their effects on learning outcomes are not always consistent(Chen et al., 2020; Zhong et al., 2024). Nevertheless, limited research has simultaneously combined instructor-mediated adaptive grouping, Arends' PBL syntax, multiple generative AI platforms under a single verification protocol, the Case Method as an active comparator, and the measurement of problem solving and debugging as two distinct skills. The novelty of this study lies in how these elements were integrated and evaluated as a single instructional design, not in a claim that each component was entirely new.

Based on this gap, the study examined differences in problem-solving and debugging skills between students who participated in AI-assisted adaptive PBL and those who participated in the Case Method after controlling for entry-level ability. Three hypotheses were proposed. H1: The combined posttest scores for problem solving and debugging differ between the two classes after controlling for both pretest scores. H2: The problem-solving posttest scores differ between the two classes after controlling for both pretest scores. H3: The debugging posttest scores differ between the two classes after controlling for both pretest scores. Figure 1 summarizes the integrated instructional framework. It shows the sequence of the intervention and the expected outcomes, not a statistical mediation or interaction model among PBL, adaptive grouping, and AI.

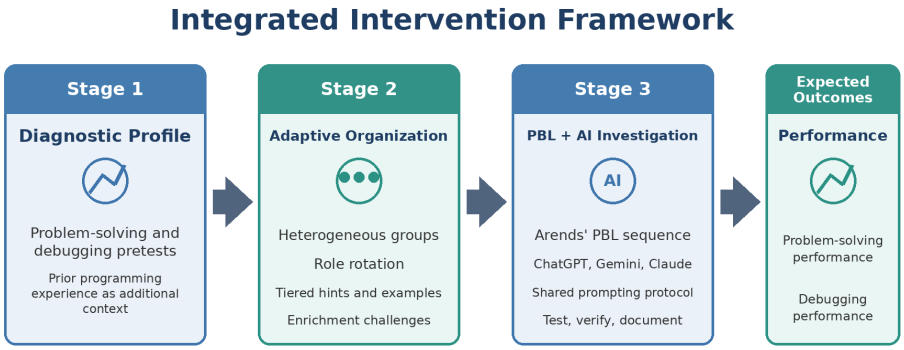

Figure 1. Conceptual framework of the study

Method

This study employed a quasi-experimental nonequivalent pretest-posttest control-group design involving two intact classes. Individual-level randomization was not performed because class membership had been administratively determined before the study. The experimental class implemented Arends' PBL as an integrated instructional condition that included adaptive grouping, tiered pedagogical support, and structured use of generative AI during the investigation phase. The control class implemented the Case Method. Learning outcomes, course content, instructional duration, and assessment instruments were aligned as far as the course context allowed. Because conditions were assigned at the class level, potential selection bias remains even though differences in entry-level ability were statistically controlled. Table 1 summarizes the research design.

Table 1. Quasi-experimental design

Group	Pretest	Treatment	Posttest	n
Experimental	O_1 and O_2	Arends' PBL with adaptive grouping and structured AI-assisted investigation	O_3 and O_4	35
Control	O_1 and O_2	Case Method without generative AI support	O_3 and O_4	34

Note. O1 = problem-solving pretest; O2 = debugging pretest; O3 = problem-solving posttest; O4 = debugging posttest.

The study was conducted in the Programming Languages course at the Department of Curriculum and Educational Technology, Faculty of Education, Universitas Negeri Padang. Sixty-nine students participated, comprising 35 students in the experimental class and 34 in the control class. All participants had complete data for the analyzed variables. Their educational backgrounds included information-technology vocational high schools, general senior high schools, and Islamic senior high schools, and they had varied prior programming experience. This background information was used only as additional context and was not treated as an ability label or a substitute for the problem-solving and debugging pretest scores.

The intervention was conducted during the third semester of the 2025 academic year. Two intact classes were purposively selected based on similarities in course context, learning outcomes, and schedules, and were then assigned as the experimental and control classes. The classes were not randomly assigned. Consequently, the instructional-model factor in the analysis represents the complete instructional condition in each class rather than the effect of any single component. Table 2 presents the study variables and their operational definitions.

Table 2. Variables and operational definitions

Variable	Role	Operational definition	Scale
Instructional model	Fixed factor	1 = Arends' PBL with adaptive grouping and AI support; 2 = Case Method	Nominal
Problem-solving pretest	Covariate	Entry-level score for solving programming problems	0–100
Debugging pretest	Covariate	Entry-level score for identifying and correcting code errors	0–100
Problem-solving and debugging posttests	Dependent variable	Postintervention scores for two separately assessed skills	0–100

Each instructional cycle in the experimental class began with an authentic programming problem that required students to produce a program meeting specified requirements. The problems were more open-ended than the cases used in the control class, requiring students to identify relevant information, decompose the problem, design an algorithm, implement code, and test several possible cases. The instructor acted as a facilitator by providing guiding questions and performance-based feedback rather than final solutions. These task characteristics formed part of the integrated PBL condition and were considered when interpreting differences between the groups.

The intervention lasted eight sessions and covered the history of programming languages, algorithms, pseudocode, sequential structures, selection, and iteration. These topics provided a foundation for subsequent topics such as arrays, functions, modular programming, searching, and sorting. The five phases of Arends' PBL were implemented in each instructional cycle. Adaptation primarily involved adjusting group composition, assigning roles, varying the level of hints and examples, and providing enrichment challenges. Generative AI was used during the investigation phase under a common verification protocol for all students. Table 3 shows how adaptive instruction and AI were integrated into the phases of Arends' PBL.

Table 3. Integration of adaptive instruction and AI into Arends' PBL syntax

Arends' PBL phase	Instructor activities	Student activities	Adaptive and AI integration
1. Orient students to the problem	Present an authentic programming problem, expected outputs, constraints, and success criteria.	Identify the context, facts, required outputs, and initial questions.	All students receive the same problem; initial hints may differ according to ability profiles.
2. Organize students for learning	Form groups of four to five students based on pretest scores, programming experience, and educational background, then assign initial roles.	Agree on roles as problem analyst, algorithm designer, programmer, debugging lead, and documenter.	Adaptive grouping creates heterogeneous groups; roles are rotated to prevent domination by experienced students.
3. Support the investigation	Provide guiding questions, monitor interactions with AI, and ensure that every AI output is verified.	Search for concepts, compare algorithms, request hints, interpret error messages, and test solutions.	ChatGPT, Gemini, and Claude are used to obtain explanations, hints, alternatives, and debugging support; final code may not be used without verification.
4. Develop and present artifacts	Facilitate code review, presentations, and program testing.	Present algorithms, code, test results, debugging logs, and decisions to accept or reject AI suggestions.	Adaptive support is provided through rubrics, testing examples, and enrichment challenges tailored to progress.
5. Analyze and evaluate	Lead reflection on strategies, errors, member contributions, and the quality of AI use.	Evaluate solutions, write individual reflections, and explain changes in understanding.	Progress profiles are updated to determine subsequent support and grouping.

Adaptive grouping was implemented in four steps. First, problem-solving and debugging pretest scores were used to map students' entry-level abilities. Second, prior programming experience and educational background were considered as additional contextual information rather than fixed ability categories. Third, groups of four to five students were formed heterogeneously so that each group included a combination of abilities conducive

to peer support. Fourth, the roles of problem analyst, algorithm designer, programmer, debugging lead, and documenter were rotated on the basis of formative evidence collected during instruction. This procedure was designed to expand every student's opportunities to reason, write code, and debug. Adaptive grouping was not tested as a separate treatment or moderator.

Instructional adjustments were made continuously across the eight sessions by considering formative quiz results, debugging logs, students' ability to explain code, and test-case results. When a group encountered difficulty, the instructor could provide additional hints or simpler examples. Conversely, groups demonstrating greater mastery received additional constraints or optimization challenges. Thus, adaptation decisions were based on performance observed during learning rather than solely on educational background. Group composition, role assignment, and forms of support remained flexible and could change as needed. Because the adaptation process was controlled by the instructor, the intervention was positioned as pedagogical rather than algorithmic adaptation.

ChatGPT, Gemini, and Claude were offered as optional support tools during the investigation phase and were not compared with one another. To reduce variation in usage patterns, every interaction followed the same six steps: state the learning objective and initial attempt; request an explanation, a guiding question, or a stepwise hint; do not use complete code without understanding it; test every suggestion with test cases; compare the AI response with course materials or documentation; and record the prompt, response, decision, and rationale in an AI interaction log. The instructor monitored compliance through interaction logs, code reviews, and student explanations. This procedure standardized the instructional function of AI use, although it could not standardize model outputs or the versions of the platforms used.

To maintain academic integrity, students were required to explain their code orally, identify sections influenced by AI, and refrain from entering personal data into AI platforms. Assessment was based not only on the final code but also on algorithms, test cases, debugging logs, and students' reasons for accepting or rejecting AI suggestions. The AI interaction logs were used to verify intervention implementation and support accountability in the learning process; they were not analyzed as an outcome variable or mediator in this study.

The control class used the Case Method. The instructor presented programming cases containing a context, system requirements, facts, code fragments, input data, or observed outputs. Students identified problems, applied concepts, analyzed alternatives, and defended their recommendations. Generative AI was not used during assessed activities in the control class. Learning outcomes, content coverage, instructional duration, and posttest instruments were aligned with those of the experimental class, while the cases remained more structured than the open-ended PBL problems. Therefore, differences in outcomes cannot be attributed solely to access to AI.

The research instruments measured two interrelated but separately assessed skills: problem solving and debugging. Parallel Form A was used for the pretest and Form B for the posttest; each form consisted of eight problem-solving tasks and eight debugging tasks. The problem-solving tasks assessed students' ability to interpret and represent requirements, decompose problems, construct algorithms, select programming structures, implement solutions, conduct comprehensive testing, and evaluate results. The debugging tasks began with erroneous or nonfunctioning code and assessed the ability to reproduce and classify errors, trace execution, localize the source of an error, formulate a causal hypothesis, apply a correction, retest, and explain the rationale for the correction. Although both skills involve implementation and testing, the task objectives and scoring evidence distinguished the process of constructing a solution from that of diagnosing and correcting errors. Table 4 presents the indicators for the research instruments.

Table 4. Indicators of the research instruments

Instrument	Primary indicators	Performance evidence	Scoring
Problem solving	Problem interpretation and representation, decomposition, algorithm design, implementation, comprehensive solution testing, and evaluation.	Requirements analysis, pseudocode or flowcharts, code, test cases, and a complete explanation of the solution.	Eight items scored with a 0-4 analytic rubric; raw scores of 0-32 are converted to a 0-100 scale.
Debugging	Error reproduction and classification, tracing, localization, causal diagnosis, correction, retesting, and explanation.	Debugging logs, tracing evidence, code changes, test results before and after correction, and justification for the correction.	Eight items scored with a 0-4 analytic rubric; raw scores of 0-32 are converted to a 0-100 scale.

Content validity was evaluated by five experts in educational evaluation, educational technology, programming education, and learning assessment. They used a four-point scale to rate relevance, clarity, and representativeness of the indicators. Aiken's V values for the 16 items averaged 0.933 and ranged from 0.911 to 0.956; all items were judged adequate and retained. The validation process also considered the alignment between the problem-solving and debugging tasks and the performance evidence they were intended to measure. These results provided content-based validity evidence for assessing the two skills separately, although they did not constitute a latent-variable test of discriminant validity.

The instruments were piloted with 32 students outside the main study sample. Corrected item-total correlations were adequate, with a minimum value of 0.640, so all items were retained. Internal reliability was also very high. Cronbach's alpha coefficients for problem solving were 0.948 for Form A and 0.950 for Form B, while the coefficients for debugging were 0.936 and 0.929, respectively. Each indicator was scored with a 0-4 analytic rubric; higher scores indicated increasingly complete, accurate, and defensible performance. Raw scores from 0 to 32 were then converted to a 0-100 scale.

The equivalence of the two instrument forms was examined in the pilot sample. Correlations between Forms A and B were 0.937 for problem solving and 0.921 for debugging. Paired-samples t tests showed that the means of the two forms did not differ significantly for either problem solving, $t(31) = 1.367$, $p = 0.182$, or debugging, $t(31) = 0.000$, $p = 1.000$. Two raters scored the responses independently using the same analytic rubric. Interrater reliability was estimated using a two-way random-effects intraclass correlation coefficient (ICC) with absolute agreement. Average-measures coefficients of 0.994 for problem solving and 0.993 for debugging indicated very high interrater consistency.

The primary analysis used multivariate analysis of covariance (MANCOVA) to test the two posttest dependent variables simultaneously while controlling for both pretest scores. The fixed factor was the instructional model, specifically AI-assisted adaptive PBL versus the Case Method. Assumption testing included residual normality using the Shapiro-Wilk test, homogeneity of variance using Levene's test, homogeneity of covariance matrices using Box's M, linearity using pretest-posttest scatterplots, and homogeneity of regression slopes using interactions between the instructional model and each covariate. Pillai's Trace was selected as the principal multivariate statistic because the two groups were relatively balanced and this statistic is comparatively robust to moderate violations of multivariate assumptions.

When the multivariate effect of the instructional model was significant, the analysis proceeded with between-subjects effects from the same MANCOVA model, estimated marginal means, between-group comparisons with 95% confidence intervals, and partial η^2. Under this approach, the univariate analyses for problem solving and debugging continued to control for both pretest scores. R^2 was reported as the proportion of variance explained by the complete model, which included the instructional-model factor and both covariates, and was therefore not interpreted as the proportion of variance caused solely by the intervention. Prior supplementary one-covariate ANCOVA analyses were not used as the basis for inference because one model used a covariate that was theoretically inappropriate for the debugging outcome. Statistical significance was set at .05.

Homogeneity of regression slopes was examined through interactions between the instructional model and each covariate. The assumption was considered satisfied when an interaction was nonsignificant at the 0.05 level. Two p values for interactions involving the problem-solving posttest were relatively close to this threshold, at $p = 0.081$ and $p = 0.079$. The main results were therefore interpreted cautiously, and these values were not treated as definitive evidence that the regression slopes would be identical in another sample. After assumption testing was completed, interaction terms were removed from the primary MANCOVA model so that the effect of the instructional model could be estimated while controlling for both pretest scores.

Results and Discussion

Complete data were available for all 69 students. The mean problem-solving posttest score in the AI-assisted adaptive PBL class was 70.09 (SD = 9.306), compared with 62.85 (SD = 8.507) in the Case Method class. For debugging, the experimental-class mean was 69.83 (SD = 8.823), while the control-class mean was 60.29 (SD = 7.725). The overall pretest means used to calculate the adjusted means were 56.14 for problem solving and 54.67 for debugging. Because the two classes were not formed through randomization, baseline equivalence was not assumed; both pretest scores were retained as covariates in the primary model. Descriptively, the posttest difference between classes was larger for debugging, as shown in Table 5.

Table 5. Descriptive posttest statistics by instructional model

Dependent variable	Instructional model	n	M	SD
Problem solving	AI-assisted adaptive PBL	35	70.09	9.306

Dependent variable	Instructional model	n	M	SD
Problem solving	Case Method	34	62.85	8.507
Debugging	AI-assisted adaptive PBL	35	69.83	8.823
Debugging	Case Method	34	60.29	7.725

Residual normality was satisfied for problem solving, W = 0.975, p = 0.187, and debugging, W = 0.993, p = 0.971. Levene's tests in the MANCOVA model were nonsignificant for problem solving, F(1, 67) = 3.171, p = 0.079, and debugging, F(1, 67) = 0.011, p = 0.917. Box's M test was also nonsignificant, Box's M = 6.230, F = 2.010, p = 0.110. Scatterplots showed positive, approximately linear pretest-posttest relationships. In addition, none of the interactions between the instructional model and the covariates reached the .05 significance level. Thus, the principal MANCOVA assumptions were considered satisfied. Nevertheless, two regression-slope tests for problem solving produced p values relatively close to 0.05, so the results still require cautious interpretation. Table 6 summarizes the assumption tests.

Table 6. Summary of MANCOVA assumption tests

Assumption	Indicator	Statistic	p	Decision
Residual normality	Problem solving	W = 0.975	0.187	Met
Residual normality	Debugging	W = 0.993	0.971	Met
Homogeneity of variance	Problem solving	F(1, 67) = 3.171	0.079	Met
Homogeneity of variance	Debugging	F(1, 67) = 0.011	0.917	Met
Homogeneity of covariance matrices	Two dependent variables	Box's M = 6.230; F = 2.010	0.110	Met
Linearity	Pretest-posttest scatterplot	Positive and approximately linear	N/A	Visually met
Homogeneity of regression slopes	Instructional model × problem-solving pretest	Problem-solving posttest: F(1, 63) = 3.136; debugging posttest: F(1, 63) = 0.004	0.081; 0.948	Met
Homogeneity of regression slopes	Instructional model × debugging pretest	Problem-solving posttest: F(1, 63) = 3.194; debugging posttest: F(1, 63) = 0.073	0.079; 0.787	Met

The interaction between the instructional model and the problem-solving pretest was nonsignificant for both the problem-solving posttest, F(1, 63) = 3.136, p = 0.081, and the debugging posttest, F(1, 63) = 0.004, p = 0.948. Similarly, the interaction between the instructional model and the debugging pretest was nonsignificant for the problem-solving posttest, F(1, 63) = 3.194, p = 0.079, and the debugging posttest, F(1, 63) = 0.073, p = 0.787. Under the formal criterion, the homogeneity-of-regression-slopes assumption was satisfied. However, because p = 0.081 and p = 0.079 were relatively close to 0.05, the adjusted problem-solving results were interpreted cautiously, particularly with regard to their generalizability to other samples.

After controlling for both pretest scores, the instructional model had a significant multivariate effect on the combined problem-solving and debugging posttest scores, Pillai's Trace = 0.456, F(2, 64) = 26.780, p < 0.001, partial η^2 = 0.456. H1 was therefore supported. Both covariates also had significant multivariate effects: problem-solving pretest, Pillai's Trace = 0.589, F(2, 64) = 45.773, p < 0.001, partial η^2 = 0.589; and debugging pretest, Pillai's Trace = 0.547, F(2, 64) = 38.643, p < 0.001, partial η^2 = 0.547. The magnitude of the two covariate effects indicates that entry-level ability continued to contribute meaningfully to variation in posttest scores, even though the two instructional conditions differed substantially after entry-level ability was controlled (Table 7).

Table 7. Results of the MANCOVA multivariate tests

Effect	Pillai's Trace	F	df	p	Partial η^2
Problem-solving pretest	0.589	45.773	2, 64	< 0.001	0.589
Debugging pretest	0.547	38.643	2, 64	< 0.001	0.547
Instructional model	0.456	26.780	2, 64	< 0.001	0.456

The partial η^2 value of 0.456 indicates a substantial multivariate effect size for the instructional-model factor in the estimated model. It cannot be interpreted as the percentage of improvement separately caused by AI, adaptive grouping, or PBL. It is also distinct from the total R^2 for each dependent variable. In this quasi-experimental design, partial η^2 represents the magnitude of the adjusted difference between the two instructional conditions after both pretest scores were taken into account.

Tests of between-subjects effects showed significant differences in problem solving, $F(1, 65) = 32.742$, $p < 0.001$, partial $\eta^2 = 0.335$, and debugging, $F(1, 65) = 50.013$, $p < 0.001$, partial $\eta^2 = 0.435$. Thus, H2 and H3 were supported. The complete model, including the instructional-model factor and both covariates, yielded $R^2 = 0.675$ for problem solving (adjusted $R^2 = 0.660$) and $R^2 = 0.684$ for debugging (adjusted $R^2 = 0.670$). These R^2 values describe the ability of the complete model to explain variation in scores, not the magnitude of the intervention's effect in isolation. Table 8 presents the adjusted means and confidence intervals for the between-group comparisons.

Table 8. Follow-up univariate results, adjusted means, and group comparisons

Variable	F(1, 65)	p	Partial η^2	Adjusted M: PBL	Adjusted M: Case	Mean difference [95% CI]
Problem solving	32.742	< 0.001	0.335	70.317	62.614	7.703 [5.014, 10.391]
Debugging	50.013	< 0.001	0.435	69.738	60.388	9.350 [6.710, 11.991]

Note. Complete model: problem solving, $R^2 = 0.675$ and adjusted $R^2 = 0.660$; debugging, $R^2 = 0.684$ and adjusted $R^2 = 0.670$. The model includes the instructional-model factor and both pretest scores. Adjusted means were calculated at the mean problem-solving pretest score of 56.14 and the mean debugging pretest score of 54.67.

The adjusted mean for problem solving was 70.317 in the AI-assisted adaptive PBL class and 62.614 in the Case Method class. The difference was 7.703 points, $p < 0.001$, with a 95% CI [5.014, 10.391]. For debugging, the corresponding adjusted means were 69.738 and 60.388, with a difference of 9.350 points, $p < 0.001$, 95% CI [6.710, 11.991]. Descriptively, partial η^2 was larger for debugging (0.435) than for problem solving (0.335). However, because no statistical test directly compared the two effect sizes, this pattern cannot be used to conclude that the intervention definitively had a stronger effect on debugging.

The two covariates showed outcome-specific patterns. The problem-solving pretest predicted the problem-solving posttest, $F(1, 65) = 61.682$, $p < 0.001$, partial $\eta^2 = 0.487$, but did not predict the debugging posttest, $F(1, 65) = 0.033$, $p = 0.857$. Conversely, the debugging pretest predicted the debugging posttest, $F(1, 65) = 60.430$, $p < .001$, partial $\eta^2 = 0.482$, but did not predict the problem-solving posttest, $F(1, 65) = 0.779$, $p = 0.381$. These findings support the decision to treat problem solving and debugging as interrelated but distinct skills and reinforce the importance of including both baseline measures as covariates.

Overall, the results show that students in the AI-assisted adaptive PBL condition achieved higher problem-solving and debugging scores than students in the Case Method condition after both pretest scores were controlled. The multivariate effect size of 0.456 indicates that the difference between conditions was substantial in the estimated model. At the same time, the strong effects of both pretest scores show that entry-level ability remained important to final achievement. Thus, the findings do not support the view that technology alone determines learning outcomes or that entry-level ability completely determines student achievement. Instead, they are more appropriately understood as evidence that an instructional design combining data-informed personalization, PBL, and structured AI use was associated with better outcomes in the context of this study. This interpretation is consistent with research showing that the benefits of AI in programming education are strongly influenced by instructional design and learner characteristics (Huang et al., 2025; Li et al., 2025; Omeh et al., 2025).

For problem-solving skill, the between-group score difference can be explained theoretically by the characteristics of Arends' PBL, which require students to engage in the entire problem-solving process rather than merely respond to a structured case. Students must interpret an open-ended problem, determine what information they need to learn, design an algorithm, implement a solution, conduct tests, and justify the result. Research on PBL in programming contexts likewise emphasizes the importance of authentic problems and student responsibility for constructing solutions (Aires et al., 2023; Liu et al., 2022; Manuaba et al., 2022), while meta-analytic evidence indicates that PBL can improve higher-order thinking and problem solving when facilitation and assessment are aligned with learning objectives (Liu et al., 2022; Manuaba et al., 2022). Nevertheless, this study did not measure the contribution of each PBL phase separately or manipulate adaptive grouping as an independent factor. The mechanisms explaining the improvement in problem solving therefore require further testing through designs capable of separating the contribution of each component.

The debugging score difference was descriptively larger than the problem-solving difference. This pattern is understandable because debugging requires a relatively rapid and iterative cycle, from formulating a hypothesis about the cause and tracing program flow to applying a correction and retesting the result. Recent research indicates that novice programmers benefit from systematic, evidence-based debugging practice, while explicit

debugging instruction remains comparatively limited (Whalley et al., 2023; Yang et al., 2024). Generative AI can provide explanations, indicate possible error locations, and support diagnosis; students in programming courses have also used chatbots for these purposes (Groothuijsen et al., 2024). In this study, however, every AI suggestion had to be tested, compared with relevant course materials or documentation, and re-explained by the students. This verification procedure was important because AI-generated code may contain errors and still requires users to evaluate it (Kohen-vacs et al., 2025). Because the two effect sizes were not compared directly through a statistical test and no mediation analysis was conducted using AI interaction logs, the larger difference in debugging should be viewed only as a descriptive pattern.

The adaptive component of the intervention should likewise be understood as pedagogical rather than algorithmic adaptation. The instructor used pretest scores and formative evidence to create heterogeneous groups, rotate roles, determine the intensity of scaffolding, and provide enrichment challenges. This approach differs from automated adaptive systems that recommend exercises or generate feedback based on learner models. Recent research indicates that hybrid systems combining adaptive mechanisms and generative AI can produce better results than adaptive support alone in programming education (Na Nongkhai et al., 2025). Meanwhile, research on dynamic grouping has reported mixed effects on learning performance, although such grouping can benefit participation and the collaborative experience (Chen et al., 2020; Zhong et al., 2024). This study extends that discussion by embedding adaptation within PBL phases, but it cannot establish that regrouping or role rotation independently improved learning outcomes. Likewise, using ChatGPT, Gemini, and Claude under a common protocol standardized their pedagogical function, not the content or quality of each platform's responses.

Using the Case Method as an active comparator is both a strength and an interpretive limitation. The comparison was not simply between active learning and passive lecturing, but between two instructional conditions that both required student engagement. The experimental class used more open-ended PBL problems, adaptive grouping, tiered support, and generative AI assistance, whereas the Case Method provided more structured cases and did not permit generative AI during assessed activities. Because several components changed simultaneously, the observed effect should be understood as the effect of an integrated instructional package. The study's contribution lies in developing a design that combines data on students' entry-level abilities, the phases of Arends' PBL, and verification-oriented use of generative AI while assessing problem solving and debugging as two separate skills. These findings complement earlier research on AI-assisted PBL (Omeh et al., 2025) without claiming that any single component was independently responsible for the entire result.

This study has several limitations. First, PBL, adaptive grouping, and AI support were implemented simultaneously, so their individual and interactive effects cannot be separated. Second, the use of two intact classes leaves open the possibility of selection bias even though entry-level ability was controlled through covariates. Third, the intervention lasted only eight sessions in one course and one department, so the findings should be generalized cautiously. Fourth, the use of three continually evolving AI platforms may have produced response variation that could not be fully controlled through a single protocol. In addition, two regression-slope interaction tests for problem solving yielded p values close to .05, indicating the need for replication with a larger sample. In practical terms, the findings suggest that instructors can combine diagnostic assessment, flexible grouping, structured AI use, verification processes, and process-oriented assessment. Future research should use factorial or multigroup designs to separate the contributions of PBL, adaptive grouping, and AI; expand samples across institutions; measure retention and transfer; and analyze students' interaction processes with AI.

Conclusion

In this study of 69 students in two programming classes, students who participated in AI-assisted adaptive PBL achieved higher problem-solving and debugging skills than those who participated in the Case Method after both pretest scores were controlled. The multivariate analysis indicated a substantial effect size, while the univariate analyses showed significant differences in both skills. Descriptively, the between-group difference was larger for debugging. These findings suggest that integrating open-ended PBL tasks, data-informed pedagogical adaptation, and verification-oriented use of generative AI has the potential to support programming instruction in classes with heterogeneous entry-level abilities.

Nevertheless, the findings must be interpreted within the limits of the research design. PBL, adaptive grouping, and AI support were implemented simultaneously, so the study cannot determine the effect of each component separately. It also did not compare the effectiveness of ChatGPT, Gemini, and Claude. Future research should therefore use factorial or multigroup designs, involve larger samples from several institutions, measure learning retention and transfer, and analyze students' interactions with AI so that the mechanisms contributing to learning outcomes can be explained more robustly.

References

Ahmadzadeh, M., Elliman, D., & Higgins, C. (2007). The impact of improving debugging skill on programming ability. Innovation in Teaching and Learning in Information and Computer Sciences, 6(4), 72–87. https://doi.org/10.11120/ital.2007.06040072

Aires, J. P., Aires, S. B. K., Pereira, M. J. V., & Alves, L. M. (2023). Using the methodology problem-based learning to teaching programming to freshman students. International Journal of Information and Education Technology, 13(3). https://doi.org/10.18178/ijiet.2023.13.3.1825

Arends, R. I. (2012). Learning to teach (9th ed.). In McGraw-Hill.

Chen, B., Hwang, G.-H., & Lin, T.-S. (2020). Impacts of a dynamic grouping strategy on students' learning effectiveness and experience value in an item bank-based collaborative practice system. British Journal of Educational Technology, 51(1), 36–52. https://doi.org/https://doi.org/10.1111/bjet.12794

Dickey, E., Bejarano, A., & Garg, C. (2024). AI-Lab: A Framework for Introducing Generative Artificial Intelligence Tools in Computer Programming Courses. SN Computer Science, 5. https://doi.org/10.1007/s42979-024-03074-y

Dochy, F., Segers, M., Van den Bossche, P., & Gijbels, D. (2003). Effects of problem-based learning: A meta-analysis. Learning and Instruction, 13(5), 533–568. https://doi.org/10.1016/S0959-4752(02)00025-7

Fitzgerald, S., McCauley, R., Hanks, B., Murphy, L., Simon, B., & Zander, C. (2008). Debugging: Finding, fixing and flailing, a multi-institutional study of novice debuggers. Computer Science Education, 18(2), 93–116. https://doi.org/10.1080/08993400802114508

Groothuijsen, S., Van den beemt, A., Remmers, J., & Leeuwen, L. (2024). AI chatbots in programming education: Students' use in a scientific computing course and consequences for learning. Computers and Education: Artificial Intelligence, 7, 100290. https://doi.org/10.1016/j.caeai.2024.100290

Holmes, W., Porayska-Pomsta, K., Holstein, K., Sutherland, E., Baker, T., Shum, S. B., Santos, O. C., Rodrigo, M. T., Cukurova, M., Bittencourt, I. I., & Koedinger, K. R. (2022). Ethics of AI in education: Towards a community-wide framework. International Journal of Artificial Intelligence in Education, 32(3), 504–526. https://doi.org/10.1007/s40593-021-00239-1

Hooshyar, D., Weng, X., Sillat, P. J., Tammets, K., Wang, M., & Hämäläinen, R. (2024). The effectiveness of personalized technology-enhanced learning in higher education: A meta-analysis with association rule mining. . . Computers & Education. https://doi.org/10.1016/j.compedu.2024.105169

Huang, A., Lin, C., Su, S., & Yang, S. (2025). The impact of GenAI-enabled coding hints on students' programming performance and cognitive load in an SRL-based Python course. British Journal of Educational Technology, 56, 1942–1972. https://doi.org/10.1111/bjet.13589

Hwang, G.-J., Xie, H., Wah, B. W., & Gašević, D. (2020). Vision, challenges, roles and research issues of artificial intelligence in education. Computers and Education: Artificial Intelligence. https://doi.org/10.1016/j.caeai.2020.100001

Kabudi, T., Pappas, I., & Olsen, D. H. (2021). AI-enabled adaptive learning systems: A systematic mapping of the literature. Computers and Education: Artificial Intelligence. https://doi.org/10.1016/j.caeai.2021.100017

Kasneci, E., Sessler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., Krusche, S., Kutyniok, G., Michaeli, T., Nerdel, C., Pfeffer, J., Poquet, O., Sailer, M., Schmidt, A., Seidel, T., … Kasneci, G. (2023). ChatGPT for good? On opportunities and challenges of large language models for education. Learning and Individual Differences. https://doi.org/10.1016/j.lindif.2023.102274

Kazemitabaar, M., Chow, J., Ma, C. K. T., Ericson, B. J., Weintrop, D., & Grossman, T. (2023). Studying the effect of AI code generators on supporting novice learners in introductory programming. Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, 1–23. https://doi.org/10.1145/3544548.3580919

Kohen-vacs, D., Usher, M., & Jansen, M. (2025). Integrating Generative AI into Programming Education : Student Perceptions and the Challenge of Correcting AI. International Journal of Artificial Intelligence in Education, 3166–3184. https://doi.org/10.1007/s40593-025-00496-4

Li, H.-J., Huang, Q.-R., Wen, L.-P., Chen, W., & Xu, Z.-Z. (2025). Generative Artificial Intelligence Supported Programming Learning: Learning Effectiveness and Core Competence. SAGE Open, 15. https://doi.org/10.1177/21582440251377986

Liu, X., Liu, S., Chen, S., Wang, Y., Zhang, B., & Kang, J. (2022). Effects of problem-based learning instructional intervention on critical thinking in higher education: A meta-analysis. Thinking Skills and Creativity. https://doi.org/10.1016/j.tsc.2022.101069

Manuaba, I. B. A. P., No, Y., & Wu, C.-C. (2022). The effectiveness of problem based learning in improving critical thinking, problem-solving and self-directed learning in first-year medical students: A meta-analysis. PLOS ONE, 17(11). https://doi.org/10.1371/journal.pone.0277339

Mirata, V., & Bergamin, P. (2023). Role of organisational readiness and stakeholder acceptance: An implementation framework of adaptive learning for higher education. Educational Technology Research and Development, 71, 1567–1593. https://doi.org/10.1007/s11423-023-10248-7

Mirata, V., Hirt, F., Bergamin, P., & van der Westhuizen, C. (2020). Challenges and contexts in establishing adaptive learning in higher education: Findings from a Delphi study. International Journal of Educational Technology in Higher Education, 17(32). https://doi.org/10.1186/s41239-020-00209-y

Mogavi, R. H., Deng, C., Kim, J. J., Zhou, P., Kwon, Y. D., Metwally, A. H., Tlili, A., Bassanelli, S., Bucchiarone, A., Gujar, S., Nacke, L. E., & Hui, P. (2024). ChatGPT in education: A blessing or a curse? A qualitative study exploring early adopters' utilization and perceptions. Computers in Human Behavior: Artificial Humans, 2(1). https://doi.org/10.1016/j.chbah.2023.100027

Na Nongkhai, L., Wang, J., Wynn, A., & Mendori, T. (2025). Evaluating Adaptive and Generative AI-Based Feedback and Recommendations in a Knowledge-Graph-Integrated Programming Learning System. Computers and Education: Artificial Intelligence, 10, 100526. https://doi.org/10.1016/j.caeai.2025.100526

Normadhi, N. B. A., Shuib, L., Md Nasir, H. N., Bimba, A., Idris, N., & Balakrishnan, V. (2019). Identification of personal traits in adaptive learning environment: Systematic literature review. Computers & Education, 130, 168–190. https://doi.org/10.1016/j.compedu.2018.11.005

Omeh, C. B., Ayanwale, M. A., Mnguni, L. E., & Olelewe, C. J. (2025). Fostering programming skill and critical thinking through AI - assisted PBL integration. 0.

Plooy, D., E., Casteleijn, D., & Franzsen, D. (2024). Personalized adaptive learning in higher education: A scoping review of key characteristics and impact on academic performance and engagement. Heliyon, 10(21). https://doi.org/10.1016/j.heliyon.2024.e39630

Qian, Y., & Lehman, J. (2018). Students' misconceptions and other difficulties in introductory programming: A literature review. ACM Transactions on Computing Education, 18(1), 1–24. https://doi.org/10.1145/3077618

Sun, D., Boudouaia, A., Zhu, C., & Li, Y. (2024). Would ChatGPT-facilitated programming mode impact college students' programming behaviors, performances, and perceptions? An empirical study. International Journal of Educational Technology in Higher Education. https://doi.org/10.1186/s41239-024-00446-5

Tlili, A., Shehata, B., Adarkwah, M. A., Bozkurt, A., Hickey, D. T., Huang, R., & Agyemang, B. (2023). What if the devil is my guardian angel: ChatGPT as a case study of using chatbots in education. Smart Learning Environments, 10(15). https://doi.org/10.1186/s40561-023-00237-x

Whalley, J., Settle, A., & Luxton-reilly, A. (2023). A Think-Aloud Study of Novice Debugging. 23(2).

Yang, S., Baird, M., O'Rourke, E., Brennan, K., & Schneider, B. (2024). Decoding Debugging Instruction: A Systematic Literature Review of Debugging Interventions. ACM Transactions on Computing Education, 24, 1–44. https://doi.org/10.1145/3690652

Yew, E. H. J., & Goh, K. (2016). Problem-based learning: An overview of its process and impact on learning. Health Professions Education, 2(2), 75–79. https://doi.org/10.1016/j.hpe.2016.01.004

Zawacki-Richter, O., Marín, V. I., Bond, M., & Gouverneur, F. (2019). Systematic review of research on artificial intelligence applications in higher education where are the educators? International Journal of Educational Technology in Higher Education, 39(16). https://doi.org/10.1186/s41239-019-0171-0

Zhang, X., Zhang, B., & Zhang, F. (2022). Student-centered case-based teaching and online-offline case discussion in postgraduate courses of computer science. International Journal of Educational Technology in Higher Education. https://doi.org/10.1186/s41239-022-00374-2

Zhong, B., Liu, X., & Huang, S. (2024). Can dynamic grouping improve students' collaborative ability and learning performance in robotics education? Technology, Pedagogy and Education, 34, 257–273. https://doi.org/10.1080/1475939X.2024.2412663