

# A Comparative Teaching Experiment of Restricted VS Unrestricted AIGC Programming Tools in Data Structure Course Design

Lingyan Yang\*, Yong Wei, Meng Li

*School of Computer Science, Guangdong University of Science and Technology, Dongguan  
Guangdong, China*

*\*Corresponding Author*

**Abstract:** Post-training questionnaires for a unified multi-file C++ data-structure project were distributed and processed for data cleaning, yielding 103 valid samples in total. The experimental group contained 54 students who were only permitted to use the VS Code CodeGeeX and Tabnine plugins, while the control group comprised 49 students granted full access to all generative-AI tools without usage restrictions. This study adopts a self-designed questionnaire consisting of five-point Likert scales, single/multiple-choice behavior items and open-ended text questions to carry out mixed quantitative and qualitative analysis. Reliability analysis, independent samples t-test and chi-square baseline balance test are conducted on scale data; meanwhile, frequency statistics of AI usage behaviors and thematic coding of open feedback are supplemented. Statistical results show that only the dimension of perceived project completion efficiency presents a significant inter-group difference, and the two cohorts hold distinct usage habits and learning pain points. This research provides empirical evidence for establishing hierarchical standardized AIGC management schemes in programming practical courses.

**Keywords:** AIGC; CodeGeeX; Tabnine; Data Structure Teaching; Comparative Experiment

## 1. Introduction

In recent years, intelligent code assistants represented by CodeGeeX, Tabnine and general large language models have been embedded into VS Code and other mainstream editors, greatly lowering the coding threshold of data structure course design. This curriculum covers abstract logical structures including linked lists, stacks, queues, trees and graphs, which brings heavy debugging pressure to students with weak

programming foundations. AIGC can rapidly generate code frameworks and resolve compilation errors, so most learners rely on AI to finish training tasks.

While various AIGC tools have been widely applied in programming teaching, unlimited and unsupervised use of these tools brings significant hidden risks to classroom instruction. Many students directly copy complete algorithm codes generated by artificial intelligence without comprehending the internal logical principles, which greatly weakens their core competencies in independent algorithm analysis and hands-on code debugging. At present, the majority of universities have not formulated clear and standardized rules for the in-class use of AIGC auxiliary tools, and there is also a lack of targeted optimization schemes for the practical teaching of data structure project courses.

Most existing scholarly research mainly explores the auxiliary functions and basic risk prevention strategies of AIGC in teaching scenarios. Nevertheless, there are few controlled comparison experiments that quantitatively analyze the teaching quality differences between limited plugin-based AIGC access and fully open AIGC usage modes. In particular, empirical studies supported by unified course practice tasks and multi-dimensional questionnaire data are still scarce, which creates a prominent research gap in this domain.

To fill the above research gap, this paper develops and conducts a controlled comparative teaching experiment. Two homogeneous parallel classes of the same grade and major are divided into two groups with restricted and full AIGC access rights respectively. With identical teaching conditions including teachers, class duration and practical training projects, this study adopts standardized questionnaires to gather students' real learning feelings and actual tool usage behaviors. The core research objectives of this work are summarized

as follows:

Summarize the differentiated AIGC usage habits and behaviors of students under two different teaching management modes;

Investigate how limited and fully open AIGC usage modes affect students' algorithm mastery ability, code standardization awareness and the efficiency of completing practical projects;

Put forward feasible hierarchical application strategies of AIGC tools for the practical teaching of data structure courses.

The overall structure of this paper is arranged as follows. Section II sorts out and analyzes domestic and foreign research results on AIGC-assisted programming teaching. Section III elaborates the specific experimental design scheme, research objects and questionnaire survey framework in detail. Section IV displays all quantitative data and qualitative analysis results obtained from the experiment. Section V further discusses the practical teaching enlightenment brought by the research data. Section VI concludes the main research findings and puts forward feasible directions for subsequent in-depth research.

## 2. Related Work

### 2.1 Application of AIGC Tools in Computer Programming Teaching

Generative AI has reshaped the delivery of programming courses in higher education [1]. Reference [2] verified that intelligent coding plugins reduce time costs on grammar writing and compilation troubleshooting, which fits well with heavy coding tasks in data structure training. Multiple teaching practices prove that AIGC can act as an auxiliary tutor to interpret algorithm logic and generate standardized experimental annotations, effectively relieving students' learning anxiety during project development [3-5].

However, over-reliance on generative AI produces adverse learning outcomes. Reference [6,7] confirmed that long-term unrestricted use of large models significantly impairs students' independent algorithm reasoning and manual debugging skills. Learners tend to abandon autonomous thinking and acquire runnable code directly from AI, which contradicts the core training goal of cultivating algorithmic thinking in data structure courses. Reference [8] further sorted out hidden drawbacks of open AIGC in practical teaching, including fragmented code

styles, nonstandard writing habits and learning inertia.

### 2.2 Research on CodeGeeX and Tabnine Plug-in Teaching Scenarios

As lightweight VS Code plugins with convenient local deployment and stable offline functions, CodeGeeX and Tabnine have been widely deployed in university computer training rooms. Reference [9] carried out targeted teaching practices with CodeGeeX in data structure courses, finding that codes generated by professional plugins conform better to classroom teaching standards compared with web large models. Reference [10] analyzed the compatibility of multiple VS Code AI plugins with C++ multi-file engineering, and verified that the combination of CodeGeeX and Tabnine covers most auxiliary scenarios including code completion and error prompts.

Most existing studies only adopt single-group teaching trials based on coding plugins, without setting unrestricted control groups for horizontal comparison. Quantitative research based on paired questionnaire data to measure learning effect gaps is insufficient.

### 2.3 Empirical Research Based on Questionnaire Measurement

Five-point Likert scales are mainstream evaluation instruments for teaching effect evaluation in educational informatization research [11]. Reference [12] collected students' subjective perceptions via questionnaires and adopted reliability analysis and independent samples t-tests for inter-group comparison, providing a standardized research paradigm for this paper. Reference [13] designed comparative trials of restricted and open AI tools in programming courses, and proved that controlled AIGC usage balances learning efficiency and ability cultivation through statistical analysis, which offers theoretical support for our experimental design.

Overall, existing studies have clarified both advantages and risks of AIGC coding assistants, but comparative quantitative experiments targeting data structure course design are rare [14,15]. This paper constructs two groups with differentiated AI access rules and carries out multi-dimensional measurement including scale test, behavior frequency statistics and open-ended thematic coding to make up for the deficiency of prior empirical research.

### 3. Teaching Experiment Design

#### 3.1 Research Objects

The participants for this study were two parallel vocational-level computer-specialty classes featuring comparable admission profiles and similar grades in prior programming coursework. Fifty-four valid questionnaire returns were collected from the experimental group under AIGC-usage constraints, whereas the unrestricted-access control group yielded 49 valid replies. Questionnaires suffering from substantial blank entries or mindless repeated answers were filtered out, resulting in 103 usable samples that satisfy the sample-size threshold for large-sample statistical testing.

To secure the internal validity of this controlled experiment, all confounding factors were tightly harmonized. The two classes were taught by the same instructor with equal class hours and identical timetables, and they worked on an identical multi-file C++ comprehensive project for data structures. The sole independent variable manipulated in this experiment was the set of rules governing AIGC utilization. Dependent variables cover tool operation proficiency, autonomous algorithm comprehension, code standardization awareness, project completion efficiency and subjective learning pain points.

#### 3.2 Group Setting Rules

Experimental Group (Restricted Plugin Group, N=54)

Throughout the whole data structure course design cycle, students are only allowed to use VS Code equipped with CodeGeeX and Tabnine. All web-based AIGC platforms including Doubao, Tongyi Lingma and ChatGPT are prohibited. Teachers regularly inspect students' plugin installation status and record violations in follow-up questionnaires.

Control Group (Unrestricted AIGC Group, N=49)  
No usage limits are imposed on any generative AI tools in this class. Students can freely select VS Code built-in plugins, web large models and other intelligent coding tools according to personal demands, without unified standards or supervision.

#### 3.3 Questionnaire Structure Design

The questionnaire is divided into five modules, with consistent single/multiple-choice options for two groups to support horizontal comparison. All

participants belong to the same major, so major difference is excluded as a confounding variable.  
Module 1: Basic Learning Information (3 Single-Choice Items)

This section collects pre-course programming foundation and habitual AI dependence, providing variables for subsequent chi-square balance test.

Module 2: AIGC Usage Behavior Questions

Item settings differ slightly by group, while AI usage purpose options remain consistent:

Experimental group: CodeGeeX & Tabnine proficiency, main usage scenarios, compliance status;

Control group: Frequently adopted AIGC types, primary usage purposes and daily usage duration.

Module 3: Core Five-Point Likert Scale

Six Likert items are originally designed with 1=Strongly Disagree to 5=Strongly Agree. Item S3 is not distributed to the control group and is entirely removed from all calculations. The remaining five dimensions are listed below:

AIGC tools reduce difficulties in writing data structure codes;

AI tools fit VS Code and multi-file compilation & debugging well;

Current AI usage mode improves the completion efficiency of course design;

This AI teaching mode is worthy of promotion in subsequent data structure training;

Current AI usage mode helps form standardized coding habits.

Module 4: Cognition and Learning Pain Point Multiple-Choice Items

Differentiated single-choice cognition questions are set for two groups with matched pain-point options:

Experimental group: Whether dual plugins improve code standardization; restricted-mode pain points;

Control group: Impacts of unlimited AI on algorithm logic thinking; open-mode pain points.

Module 5: Open-Ended Suggestion Item

An open text box is reserved for students to put forward targeted optimization suggestions for the corresponding AI teaching mode, which is adopted for follow-up thematic qualitative coding.

#### 3.4 Data Collection and Analysis Plan

After finishing the unified data structure project, anonymous questionnaires are released via Wenjuanxing. Invalid questionnaires with blank content and meaningless repetitive answers are

eliminated, generating 103 valid samples. This research adopts mixed-method analysis: Cronbach's  $\alpha$  reliability test, independent samples t-test and chi-square balance test are performed on scale data; frequency statistics of single/multiple-choice items and manual thematic coding of open text responses are conducted. Complete test results and thematic summaries are presented in Section 4.

## 4. Data Analysis and Experimental Results

### 4.1 Reliability Evaluation of the Questionnaire Scale

This comparative research adopts five 5-point Likert items to measure learners' subjective perceptions under two AIGC management modes. Item S3 is excluded as relevant questions are not delivered to the control group. Valid responses include 54 experimental participants and 49 control participants, forming a total sample of 103.

**Table 1. Reliability Analysis Based on Cronbach's Alpha**

| Sample Category           | Valid Items | Cronbach's $\alpha$ | Reliability Judgment               |
|---------------------------|-------------|---------------------|------------------------------------|
| All 103 participants      | 5           | 0.907               | Excellent ( $\alpha \geq 0.90$ )   |
| Experimental Group (N=54) | 5           | 0.951               | Excellent ( $\alpha \geq 0.90$ )   |
| Control Group (N=49)      | 5           | 0.881               | Good ( $0.80 \leq \alpha < 0.90$ ) |

Cronbach's  $\alpha$  coefficient is adopted to evaluate inter-item correlation, the standard metric for questionnaire stability. Calculation results show the overall reliability coefficient reaches 0.907, falling into the excellent reliability interval. Grouped reliability results indicate the experimental group achieves  $\alpha = 0.951$  and the

control group achieves  $\alpha = 0.881$ . All reliability indicators exceed the widely recognized threshold of 0.70 proposed by Nunnally (1978), proving the scale possesses stable measurement performance to support inter-group comparisons. Detailed data are shown in Table 1.

### 4.2 Descriptive Statistics and Independent Samples T-test Comparison

Table 2 lists the mean and standard deviation of each scale item grouped by cohort. An obvious numerical gap appears on item S4: experimental students report a much higher average score ( $M = 3.981$ ,  $SD = 0.765$ ) than control students ( $M = 3.245$ ,  $SD = 0.480$ ). Means of S1, S2, S5 and S6 remain close between the two groups.

**Table 2. Descriptive Statistics of Scale Items by Group**

| Item Code | Exp. Mean | Exp. Std. Dev. | Ctrl. Mean | Ctrl. Std. Dev. |
|-----------|-----------|----------------|------------|-----------------|
| S1        | 3.759     | 0.845          | 4.041      | 0.815           |
| S2        | 3.852     | 0.737          | 3.878      | 0.832           |
| S4        | 3.981     | 0.765          | 3.245      | 0.480           |
| S5        | 4.019     | 0.765          | 3.898      | 0.872           |
| S6        | 3.889     | 0.793          | 3.735      | 0.908           |

Independent samples t-tests are conducted to quantify whether score gaps reach statistical significance, with all metrics summarized in Table 3. Only item S4 exhibits a prominent inter-group difference:  $t(101) = 5.786$ ,  $p < 0.001$ , with a large effect size of Cohen's  $d = 1.142$ . The 95% confidence interval of the mean difference [0.489, 0.984] further confirms that students under the restricted plugin mode hold far more positive evaluations on project efficiency. In contrast, p-values of S1, S2, S5 and S6 all exceed 0.05, meaning no statistically significant gaps exist on these dimensions.

**Table 3. Independent Samples T-Test Results**

| Item Code | t Statistic | df  | p Value | Mean Difference | 95% CI Lower | 95% CI Upper | Significance       |
|-----------|-------------|-----|---------|-----------------|--------------|--------------|--------------------|
| S1        | -1.717      | 101 | 0.089   | -0.282          | -0.606       | 0.043        | Not significant    |
| S2        | -0.166      | 101 | 0.868   | -0.026          | -0.335       | 0.283        | Not significant    |
| S4        | 5.786       | 101 | < 0.001 | 0.737           | 0.489        | 0.984        | Highly Significant |
| S5        | 0.748       | 101 | 0.456   | 0.121           | -0.202       | 0.443        | Not significant    |
| S6        | 0.920       | 101 | 0.360   | 0.154           | -0.181       | 0.489        | Not significant    |

Note: Cohen's  $d$ : S1=-0.339, S2=-0.033, S4=1.142, S5=0.148, S6=0.182.

### 4.3 Baseline Equivalence Verification of Samples

All students major in the same discipline, so major difference cannot serve as a confounding variable. Chi-square independence tests are carried out on two core baseline indicators: pre-

course programming foundation and habitual AI dependence before the project, to verify consistent initial learning conditions.

As listed in Table 4, both variables generate p-values higher than the 0.05 threshold. For programming foundation:  $\chi^2 = 4.162$ ,  $df = 2$ ,  $p = 0.125$ ; for pre-project AI dependence:  $\chi^2 = 2.564$ ,

df = 2,  $p = 0.278$ . These results demonstrate balanced distributions of programming ability and AI usage tendencies across cohorts. Table 5 supplements the count and percentage distribution of students with different proficiency levels.

**Table 4. Chi-Square Test for Baseline Balance**

| Test Variable                | $\chi^2$ Value | df | p Value | Balance Conclusion      |
|------------------------------|----------------|----|---------|-------------------------|
| Programming Foundation       | 4.162          | 2  | 0.125   | Balanced ( $p > 0.05$ ) |
| Pre-experiment AI Dependence | 2.564          | 2  | 0.278   | Balanced ( $p > 0.05$ ) |

**Table 5. Programming Foundation Distribution**

| Proficiency Level   | Exp. n (%) | Ctrl. n (%) |
|---------------------|------------|-------------|
| Strong foundation   | 3 (5.6%)   | 7 (14.3%)   |
| Moderate foundation | 23 (42.6%) | 13 (26.5%)  |
| Basic grammar only  | 28 (51.9%) | 29 (59.2%)  |

Note: Tiny subgroups are merged to avoid sparse-cell bias in chi-square calculation.

#### 4.4 Comprehensive Summary of Scale Test Findings

Three core conclusions are drawn from scale statistics. First, the five-scale questionnaire achieves sound internal consistency, which establishes a solid basis for subsequent inter-group comparisons. Second, only item S4 yields statistically significant differences with a large effect size, indicating that the standardized plugin configuration delivers distinct perceived advantages regarding project completion efficiency. Third, the two participant groups exhibit comparable pre-experimental learning backgrounds, which rules out confounding biases when conducting group-level comparisons.

#### 4.5 AI Usage Behavior and Learning Pain Point Analysis

To complement the scale-based outcomes, this section analyzes data obtained from single- and multiple-choice questions, and all frequency-related results are summarized in Tables 6 to 9.

**AIGC Tool Mastery and Usage Ecosystem**  
Within the experimental group ( $N = 54$ ), 33 students (61.1%) gained proficiency in advanced plugin-related features; 15 respondents (27.8%) merely relied on code auto-completion, and the remaining 6 participants (11.1%) lacked competent plugin-operating skills. For the control group ( $N = 49$ ), Doubao stood out as the most frequently-used tool with a usage rate of 77.6%, whereas CodeGeeX saw very low adoption at

only 4.1%.

**Table 6. Aigc Tool Usage Distribution**

| (a) Experimental Group Plugin Proficiency (n=54)           | Proficiency               | n  | %    |
|------------------------------------------------------------|---------------------------|----|------|
|                                                            | Master advanced functions | 33 | 61.1 |
|                                                            | Only auto-completion      | 15 | 27.8 |
|                                                            | Unable to use plugins     | 6  | 11.1 |
| (b) Control Group Adopted Tools (n=49, multiple selection) | Tool Category             | n  | %    |
|                                                            | Doubao web/VS             | 38 | 77.6 |
|                                                            | ChatGPT & web LLMs        | 15 | 30.6 |
|                                                            | General code plugins      | 11 | 22.4 |
|                                                            | CodeGeeX                  | 2  | 4.1  |
|                                                            | Rarely use AI             | 2  | 4.1  |

#### Primary AI Usage Purposes

Code completion and error troubleshooting are the two most frequent demands for both groups. Only a small proportion of students directly copy finished codes (7.4% Exp, 4.1% Ctrl). More control students utilize AI for algorithm optimization (59.2% vs 33.3%).

**Table 7. Primary Ai Usage Purposes (Multiple Responses)**

| Purpose                            | Exp. n (%) | Ctrl. n (%) |
|------------------------------------|------------|-------------|
| Data structure coding              | 29 (53.7)  | 34 (69.4)   |
| Compile error fix                  | 39 (72.2)  | 33 (67.3)   |
| Algorithm explanation & annotation | 29 (53.7)  | 23 (46.9)   |
| Algorithm optimization             | 18 (33.3)  | 29 (59.2)   |
| Direct copy assignment code        | 4 (7.4)    | 2 (4.1)     |

**Restricted Group Compliance Status**, Only 11 students (20.4%) fully comply with the plugin-only rule; 79.6% occasionally or frequently use external web AI, mainly due to insufficient plugin capacity for complex tree and graph algorithms.

**Table 8. Experimental Group Usage Compliance (N=54)**

| Compliance Level       | n  | %    |
|------------------------|----|------|
| Fully abide by rules   | 11 | 20.4 |
| Occasional external AI | 35 | 64.8 |
| Frequent external AI   | 8  | 14.8 |

**Group-Wise Learning Pain Points**, The two groups show completely different pain distributions. Experimental students mainly complain about limited plugin functions and IDE conflicts; control students are troubled by AI over-reliance and inconsistent code styles.

**Table 9. Self-Reported Learning Pain Points**

| Pain Point                           | Exp. n (%) | Ctrl. n (%) |
|--------------------------------------|------------|-------------|
| Plugins fail complex algorithms      | 34 (63.0)  | 0 (0)       |
| VS Code plugin conflicts             | 23 (42.6)  | 0 (0)       |
| Limited AI inquiry channels          | 13 (24.1)  | 0 (0)       |
| No obvious obstacles                 | 7 (13.0)   | 9 (18.4)    |
| Over-reliance weakens coding ability | 0 (0)      | 31 (63.3)   |

|                             |       |           |
|-----------------------------|-------|-----------|
| Inconsistent AI code styles | 0 (0) | 21 (42.9) |
| Scattered tool resources    | 0 (0) | 17 (34.7) |

#### F. Qualitative Analysis of Open-Ended Survey Feedback

This study performed thematic coding on valid open-ended responses, including 14 texts from the experimental group and 11 texts from the control group. Three key themes were finally identified through analysis: instructional model strengths, learning obstacles, and pedagogical optimization recommendations.

##### Recognized Advantages

Students in the restricted group learned with a unified IDE environment, which eliminates fatigue caused by frequent tool switching and works well for learners at the intermediate level. The unrestricted group, by comparison, offers flexible access to cross-platform algorithm references, and students in this setting have expressed a clear need for standardized AI-based teaching guidance.

##### High-Frequency Learning Barriers

Students with weak foundational skills within the experimental group reported that the available plugins failed to support complex-project development and fell behind mainstream web-based AI tools. Meanwhile, most participants from the control group noted that unrestricted, long-term access to free-of-charge AI services weakened their independent debugging abilities, and they advocated for reasonable constraints on AI usage.

##### Student-Oriented Optimization Suggestions

Three operable strategies are summarized: tiered AI access control matched with task difficulty, systematic AI-assisted coding literacy courses, and teacher-led formative assessment including code review and oral algorithm defense. Students prefer a moderate layered management framework rather than absolute restriction or full freedom.

## 5. Discussion

This section comprehensively interprets mixed quantitative and qualitative results, comparing merits and defects of restricted plugin mode and unrestricted AIGC mode combined with student feedback.

First, the unified VS Code + CodeGeeX + Tabnine environment standardizes coding styles, which explains the significant S4 advantage of the experimental group. However, 79.6% of experimental students bypass usage limits, as plugins lack support for tree and graph

algorithms. Rigid single-plugin bans cannot satisfy advanced learning demands.

Second, unrestricted AIGC provides rich algorithm resources for cross comparison, yet it brings prominent cognitive risks. Over 63% of control students report degraded independent thinking, and messy code styles hinder classroom evaluation. Even though direct code copying only accounts for 4.1% in the control group, the hidden lazy learning risk cannot be ignored. Most free-mode students actively request usage constraints, proving total freedom is not ideal.

Third, chi-square tests confirm balanced baseline ability for two cohorts, eliminating sample bias. Reliability coefficients guarantee scale validity, and only efficiency dimension has large inter-group difference. Other dimensions show no obvious gaps between modes.

Fourth, open-text feedback converges three student-proposed reform directions: graded AI permission, AI literacy training and regular teacher assessment, which provide operable schemes for data structure teaching reform.

In addition, this research has limitations. All data rely on student self-report without objective indicators such as code scoring and classroom operation logs. Follow-up research can integrate objective evaluation to enhance result persuasiveness.

## 6. Conclusion and Future Work

### 6.1 Conclusion

A controlled comparative experiment with differentiated AIGC access rules is designed for data structure course design. The experimental group (N=54) is limited to CodeGeeX and Tabnine, while the control group (N=49) has unrestricted AI access. Mixed quantitative and qualitative analysis based on 103 valid samples generates three core conclusions:

The five-item Likert scale has excellent internal consistency. Chi-square tests verify balanced pre-experiment backgrounds. Only perceived project efficiency shows a highly significant inter-group difference in favor of the restricted plugin group. Distinct usage behaviors and pain points exist between cohorts. Restricted students suffer insufficient plugin functions, while unrestricted students face AI over-reliance risks. Nearly 80% of experimental students bypass usage bans, proving absolute plugin prohibition is ineffective. Neither fully restricted nor fully unrestricted access matches curriculum training goals.

Students propose a layered, moderately controlled AIGC framework with systematic literacy guidance to balance auxiliary efficiency and independent algorithm thinking.

The three conclusions fully respond to the three research objectives raised in the introduction, forming a complete logical closed loop from experimental design to empirical results.

## 6.2 Future Work

Three avenues are available for follow-up research. Researchers may recruit participants from more grades and majors and run replicated experiments to test the robustness of current findings. Besides, objective indicators like code scoring and manual debugging frequency should be adopted to merge subjective perceptions with objective evidence. It is also worthwhile to build customized AIGC permission policies for students of disparate programming proficiency.

## Acknowledgments

This work is supported by the Teaching Reform Research Project of GUANGDONG UNIVERSITY OF SCIENCE & TECHNOLOGY under Grant No. GKZLGC2025062. The authors appreciate all participating students for authentic questionnaire feedback.

## References

- [1] L. Evans and K. Miller, "Design Scaffolds for AI-Aided Introductory Programming," *Computers & Education*, vol. 268, Art. 104216, 2025.
- [2] L. Liu and S. Chen, "Controlled AIGC Application in Vocational Data Structure Courses," in *Proc. 2024 IEEE International Conference on Computer Science and Education (ICCSE)*, 2024, pp. 456–461.
- [3] Y. Zhang, H. Li and J. Wang, "Research on the Application of Intelligent AI Coding Tools," *IEEE Access*, vol. 12, pp. 98765–98773, 2024.
- [4] Z. Huang and Y. Gao, "Balancing Learning Efficiency and Ability Cultivation," *IEEE Trans. Educ.*, vol. 67, pp. 512–519, 2024.
- [5] M. Zhang and L. Yang, "Application and Practice of VS Code Plug-in AI Programming Tools," in *Proc. 2025 IEEE International Conference on Computer Engineering Education (ICCEE)*, 2025, pp. 189–194.
- [6] H. Wang and M. Lin, "CodeGeeX-Assisted Programming Teaching Optimization," in *Proc. 6th International Conference on Artificial Intelligence in Education (ICAIED)*, 2025, pp. 210–215.
- [7] X. Zhao, Y. Wu and Z. Zhang, "The Negative Impact of Unrestricted Generative AI," *Comput. Educ. Res. (Chinese Journal)*, vol. 30, no. 2, pp. 89–96, 2023.
- [8] S. Wang, "Problems and Countermeasures of Unrestricted Generative AI," *Mod. Educ. Technol. (Chinese CSSCI Journal)*, vol. 34, no. 5, pp. 78–85, 2023.
- [9] T. Chen and R. Xu, "Empirical Analysis of AI Dependency," in *Proc. 2023 International Conference on Information System Educational Engineering (ISEE)*, 2023, pp. 321–326.
- [10] Q. Liu and H. Zhang, "Research on the Influence of Different AIGC Modes," *J. Vocat. Educ. Technol.*, vol. 42, no. 4, pp. 66–73, 2024.
- [11] R. Chen, "Likert Scale-Based Empirical Research," in *Proc. 5th International Conference on Statistical Analysis Educational Research*, 2023, pp. 99–104.
- [12] W. Zhou, "Risk Prevention and Standardized Application Strategy," *High. Educ. Forum*, vol. 15, no. 6, pp. 33–38, 2023.
- [13] G. Peng and X. Cao, "Reforming Data Structures Education," in *Proc. International Conference on Educational Information Technology*, Atlantis Press, 2025, pp. 128–133.
- [14] P. Zhang and S. Liu, "Comparative Experimental Study of Restricted and Open AI," *Int. J. Educ. Inf. Technol.*, vol. 9, pp. 41–48, 2024.
- [15] L. Wang, "Generative AI Integrated Teaching Reform," *Computer Education (Chinese Professional Journal)*, vol. 11, pp. 62–66, 2022.