Journal of Contemporary Educational Research, 2026, Volume 10, Issue 10
https://bbwpublisher.com/journal/JCER
Online ISSN: 2208-8474
Print ISSN: 2208-8466

Optimization of Teaching Mode and Path Exploration of University Algorithm Courses Guided by Computational Thinking

Zhantao Li

School of Computer Application, Guilin University of Technology, Guilin 541004, Guangxi, China

Copyright: © 2026 Author(s). This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY 4.0), permitting distribution and reproduction in any medium, provided the original work is cited.

Abstract: Algorithm courses are core foundational courses for computer-related, artificial-intelligence-related, big-data-related, and other relevant majors in universities. They undertake the important mission of cultivating students' capabilities in logical reasoning, problem decomposition, logical modeling, and efficient problem-solving, and serve as a key carrier for students to build core professional literacy in computer science. Guided by computational thinking, this paper clarifies core principles for optimizing algorithm-course teaching. Based on existing pain points in current university algorithm teaching, it explores systematic and implementable teaching-optimization paths. Meanwhile, it prepares to break traditional limitations and pursue innovations, promotes the transformation of algorithm courses from knowledge-imparting-oriented teaching to thinking-cultivation and capability-empowering teaching, helps students form regular algorithm-thinking patterns, consolidates their core professional competencies, and provides theoretical references for teaching reform and quality improvement of talent cultivation for university algorithm courses.

Keywords: Computational thinking; Algorithm course; Teaching mode; Teaching optimization; Talent cultivation

Online publication: November 19, 2026

1. Principles for optimizing the teaching mode of university algorithm courses guided by computational thinking

1.1. Principle of thinking-oriented and capability-priority

Under the guidance of computational thinking, the core of teaching optimization for university algorithm courses is to abandon the traditional single teaching objective centered on knowledge-point mastery, algorithm memorization, and high problem-solving scores, and take thinking cultivation and capability improvement as the core teaching orientation. Rooted in the core connotation of computational thinking, including abstraction, decomposition, modeling, and optimization, this principle regards the teaching of algorithm knowledge as a carrier for thinking cultivation rather than the ultimate teaching goal. In the construction of the teaching system, the cultivation of students' algorithm-logical thinking, problem-solving

thinking, and innovation-optimization thinking shall always be put in the first place. The weight of assessment on mechanical knowledge memorization shall be reduced, and importance shall be attached to fostering students' ability to independently analyze complex computational problems, build algorithm-solving models, and troubleshoot logical flaws in algorithms [1–3]. Moreover, this principle runs through the whole process of curriculum teaching design, classroom teaching, and assessment and evaluation. It fundamentally changes the traditional teaching mode in which students only grasp superficial knowledge without understanding underlying principles. Students can not only master the core logic and application scenarios of various algorithms, but also develop a thinking mode of independently applying computational thinking to solve various complex engineering and computational problems. It realizes the transformation from knowledge learning to thinking-ability development, and conforms to the core requirements for cultivating application-oriented and innovative talents in universities.

1.2. Principle of in-depth integration of theory and practice

The in-depth integration of theory and practice constitutes the main guarantee for implementing computational thinking in algorithm-course teaching, and it is also a key principle for solving the problems of empty theories and disconnection from practice in university algorithm teaching. Algorithm courses feature strong theoretical and practical attributes. Theory serves as the logical foundation for algorithm design, while practice realizes computational thinking and consolidates algorithm capabilities. The two promote each other and are inseparable. This principle holds that teaching optimization shall break the separation between theoretical classrooms and practical training sessions, discard the traditional segmented teaching mode of "delivering concentrated theoretical lectures first and conducting concentrated experiments afterward," and realize the simultaneous implementation of theoretical explanation, thinking guidance, and practical verification. Theoretically, elaborate explanations shall be given of algorithm design ideas, logical principles, scope of application, advantages, and disadvantages to consolidate students' theoretical understanding. Practically, on the basis of theoretical knowledge, students shall be guided to solve problems with computational thinking and convert thinking procedures such as problem decomposition, model building, algorithm implementation, and iterative optimization into practical activities. Through the integration of theory and practice, a closed-loop teaching system is constructed, in which theory provides guidance, practice delivers verification, thinking acts as the link, and thinking deepens theoretical understanding [4].

1.3. Principle of stratified teaching and teaching students in accordance with their aptitude

University students show great individual differences in programming foundations, logical-thinking capabilities, aptitude for algorithm learning, and learning demands. Unified and homogenized teaching modes cannot meet the growth needs of all students. The principle of stratified teaching and teaching students in accordance with their aptitude serves as the premise for targeted cultivation of computational thinking. Based on individual differences in students' capabilities, this principle takes cultivating computational thinking for students at different levels as the main objective, and abandons "one-size-fits-all" teaching content, teaching progress, and teaching requirements. Teaching optimization shall take into account the different learning demands of students with weak foundations, intermediate-level students, and top innovative students, so as to build stratified knowledge systems, thinking-training systems, and capability-evaluation systems [5].

For students with weak foundations, the focus is on cultivating algorithm thinking and consolidating their capabilities in problem decomposition and basic modeling. For intermediate-level students, emphasis is laid on cultivating their algorithm-application and optimization thinking, and improving their proficiency and accuracy in problem solving. For top-performing students, efforts shall be made to cultivate their innovative thinking and design-oriented thinking for complex-system algorithms, and encourage them to pursue algorithm innovation and cross-domain application. By adopting stratified and adaptive teaching modes, every student can achieve gradual improvement in computational thinking and algorithm capabilities on the basis of their original level, maximizing teaching effectiveness.

2. Optimization paths for the teaching mode of university algorithm courses guided by computational thinking

2.1. Reconstruct gradient-oriented curriculum-content system and anchor the core objective of computational-thinking cultivation

Reconstructing the teaching-content system of algorithm courses based on the core connotation of computational thinking serves as the premise for teaching-mode optimization. It is necessary to completely break the traditional content-arrangement mode of simply listing algorithms by type, and build a hierarchically-progressive and inter-connected curriculum-content structure centered on four core capabilities of computational thinking: abstraction, decomposition, modeling and optimization [6,7]. A general-education module on computational thinking shall be added at the beginning of the course. Combined with the characteristics of algorithm courses, systematic explanations shall be delivered on the core connotation and application logic of computational thinking, as well as its relationship with algorithm design, so that students can form an understanding of "guiding algorithm design with thinking" and change their mindset of learning algorithms mechanically. Next, original curriculum knowledge points shall be decomposed and reorganized, and content modules shall be reconstructed following the gradient logic of basic-thinking cultivation, core-capability improvement, and innovation-capability expansion. The basic module mainly covers enumeration, recursion, greedy, and other basic algorithms, and focuses on cultivating students' capabilities in problem decomposition and simple modeling. The core module centers on dynamic programming, divide-and-conquer, graph-theory algorithms, and emphasizes the cultivation of students' capabilities in logical reasoning, scenario adaptation, and algorithm optimization. The expansion module designs complex, comprehensive algorithm problems combined with engineering realities and disciplinary frontiers to foster students' systematic thinking and innovative-design thinking. Supporting curriculum resources shall be optimized simultaneously: outdated and practice-detached theoretical content shall be removed, and cutting-edge algorithm-application cases related to artificial intelligence, big data, and engineering practice shall be added, so that the cultivation of computational thinking is integrated into the process of knowledge-point learning [8]. In addition, in accordance with the principle of stratified teaching, different learning contents shall be matched for students at different levels. Students at the basic level focus on training in basic algorithm thinking and understanding of classic algorithms. Students at the improvement level focus on training in algorithm variants and scenario application. For top-level students, extended content such as algorithm innovation and algorithm-integration design shall be added. In this way, curriculum contents can accurately correspond to the growth demands of each student's computational thinking, realizing precise alignment between knowledge teaching and thinking cultivation.

2.2. Innovate immersive classroom-teaching mode and promote the implementation of computational thinking in classrooms

Aiming at the drawbacks of traditional algorithm classrooms such as dull content, emphasis on indoctrination rather than guidance, and low student participation, immersive, heuristic, and inquiry-oriented classroom-teaching modes shall be innovated under the fundamental guidance of computational-thinking cultivation. The roles of teachers and students in classrooms shall be completely transformed to build a student-centered classroom ecology focused on thinking training. At the classroom-introduction stage, instead of directly lecturing on algorithm principles, practical complex computational problems shall be put forward combined with real-life scenarios and engineering cases. Students are prompted to observe and decompose problems by themselves, extract core requirements, voluntarily strip irrelevant parts of problems, and build simplified mathematical and computational models, so as to stimulate students' computational thinking from the very beginning of class [9]. At the knowledge-explanation stage, the mode of teachers stating principles while students memorize formulas shall be abandoned. Heuristic teaching driven by problems and progressive questioning shall be adopted. Progressive questions shall be raised starting from the core logic of algorithm design, enabling students to independently deduce algorithm principles, analyze the advantages and disadvantages of algorithms, and think about applicable scenarios of algorithms. They can master the thinking logic of algorithm design in the process of independent inquiry, instead of passively accepting existing knowledge. Mechanical exercises such as copying code and applying templates shall be eliminated, and inquiry-oriented classroom tasks shall be designed. Students shall independently finish problem decomposition, model construction, algorithm design, code implementation, and bug troubleshooting. Teachers only offer targeted guidance on students' thinking misunderstandings and logical loopholes. Meanwhile, group-based inquiry modes shall be introduced. Complex algorithm problems are divided into several subtasks, and group members conduct modeling, design, optimization, and verification separately. Students' systematic thinking and logical-critical-thinking capabilities are exercised in the course of cooperation. After full-process immersive thinking training, classrooms will be transformed from "knowledge-lecturing venues" into "thinking-cultivation positions," realizing the regular implementation of computational thinking in classroom teaching [10].

2.3. Build theory-practice-integrated practical platform to consolidate and improve computational-thinking capabilities

The cultivation of computational thinking cannot be separated from continuous practical training. In view of the problems of theory-practice disconnection, single practical-training forms and fragmented practical-training contents in traditional algorithm courses, a systematic, gradient-oriented and scenario-based theory-practice-integrated practical-teaching platform shall be constructed, forming a comprehensive and closed-loop practical-training system to help students consolidate and enhance their computational-thinking capabilities. Integrate online and offline practical resources to build a three-tier practical platform covering basic training, comprehensive training, and innovative training. Online, independent-training modules shall be built relying on algorithm-training platforms and online question banks, with gradient-oriented practical exercises to meet students' demands for after-class independent thinking training. Offline, immersive algorithm-practice scenarios shall be created in professional laboratories, with complete training equipment and case resources to support the development of comprehensive and innovative algorithm-practice projects. Second, practical-teaching content shall be reconstructed. Single verification-based experiments shall be

discarded, and design-oriented, comprehensive and innovative practical projects shall be greatly increased. Basic verification experiments are only used for consolidating fundamental algorithm principles. Students are mainly guided to verify their own thinking logic through practice. Design-oriented experiments require students to independently design exclusive algorithm schemes and complete code implementation and optimization. Comprehensive experiments set multi-dimensional and multi-constraint complex computational problems combined with real-world industrial scenarios, requiring students to finish systematic algorithm design by adopting thinking modes of decomposition, abstraction, and optimization. A full-process practical-guidance mechanism shall be established. Teachers shall provide whole-process tracking and guidance for students' thinking misunderstandings, design defects, and optimization weaknesses in practice, and record the development trajectory of students' thinking [11–13]. Furthermore, practical-exchange platforms shall be built, and algorithm-training sharing sessions shall be held regularly. Students can share their problem-solving ideas, modeling methods, and optimization logics with others. Thinking can be expanded through communication and idea collision, improving the application capabilities of computational thinking and realizing effective transformation from theoretical thinking to practical capabilities.

2.4. Construct diversified whole-process evaluation system and establish educational guidance oriented toward computational thinking

Traditional evaluation for algorithm courses mainly adopts final written examinations and code assessments, laying stress on students' knowledge memorization and problem-solving results while ignoring their thinking processes and capability development. It cannot adapt to the teaching objectives of computational-thinking cultivation. Therefore, it is necessary to establish a diversified, whole-process evaluation system that emphasizes thinking and capabilities, and set up educational-evaluation guidance centered on computational thinking. First, core evaluation dimensions shall be adjusted. Single result-oriented evaluation shall be transformed into multi-dimensional evaluation covering thinking process, knowledge mastery, practical capabilities, and innovative literacy. The proportion of final written-exam scores shall be reduced, while content including classroom thinking performance, practical-inquiry procedures, algorithm-design ideas, and problem-optimization innovation shall be added. Key assessment shall be conducted on students' computational-thinking capabilities reflected in problem decomposition, model construction, algorithm optimization, and logical critical thinking. Second, whole-process dynamic evaluation shall be implemented to break the evaluation mode of "determining grades by a single examination." Evaluation shall be integrated into all links, including pre-class preview, classroom inquiry, after-class practical training, phased training, and final assessment. Pre-class assessment focuses on students' capabilities of independent thinking and problem prediction. In-class assessment examines their capabilities of inquiry-based thinking, logical expression and collaborative critical thinking. After-class assessment evaluates their capabilities of practical implementation, iterative optimization, and independent expansion. Through phased dynamic records, the growth process of students' computational thinking can be fully understood [14,15]. Diversified evaluation subjects shall be introduced to build a multi-source evaluation system combining teacher evaluation, student self-evaluation, and group mutual evaluation. Teachers mainly evaluate students' thinking and capabilities. Students sort out their own thinking deficiencies and growth experiences. In group mutual evaluation, attention shall be paid to thinking performance and task completion in the process of cooperation, so that evaluation results become more comprehensive and objective. In addition, assessment question types and

contents shall be optimized. Questions requiring mechanical memorization in written examinations and practical assessments shall be reduced. More question types such as complex computational-problem decomposition, algorithm-scheme design, multi-algorithm comparative optimization and adaptation for innovative scenarios shall be added. Students are prompted to attach importance to the cultivation and application of computational thinking, so that evaluation can give full play to its guiding, motivating and empowering functions.

3. Conclusion

To sum up, the core characteristics of computational thinking, including problem abstraction, hierarchical decomposition, logical modeling, iterative optimization, and systematic coordination, are highly consistent with the teaching essence of algorithm courses for solving complex computational problems and cultivating students' logical and innovative capabilities. It provides scientific theoretical guidance and core value orientation for the teaching-mode reform of university algorithm courses. Algorithm-teaching reform guided by computational thinking can effectively eliminate inherent drawbacks of traditional teaching, promote the transformation of course teaching from passive knowledge imparting to active thinking cultivation, from unified teaching to personalized education, and from result-oriented assessment to whole-process capability empowerment, and practically improve the pertinence and effectiveness of algorithm-course teaching. Meanwhile, this teaching mode can systematically cultivate students' core computational-thinking and problem-solving capabilities, help students adapt to complex industrial technical scenarios, provide solid teaching support for universities to cultivate high-quality innovative computer-science talents, and offer valuable theoretical references for the teaching reform of similar courses.

Disclosure statement

The author declares no conflict of interest.

References

[1] Ren L, Han X, Luo C, 2025, Curriculum Reform and Exploration of "Programming Design" Oriented to the Cultivation of College Students' Computational Thinking. Industry and Information-Technology Education, (10): 50–53.

[2] He Z, Luo J, Yourong H, et al., 2025, Teaching Reform of the Course Introduction to Computation and Artificial Intelligence Under the Background of Large-Scale AI Models. Computer Education, (9): 178–184.

[3] Tang Q, Zhang H, Hui X, et al., 2025, Teaching Reform of Computer-Major Courses Oriented Toward Computational Thinking. Computer Education, (6): 130–134.

[4] Zhang C, Tan H, 2025, Research on Teaching Reform of Python Course Oriented Toward Computational-Thinking Cultivation. Computer Knowledge and Technology, 21(16): 167–169.

[5] Gao Q, 2025, Research on Teaching Reform of "Algorithm Design and Analysis" Course in Application-Oriented Universities for Cultivating Digital-Intelligent Engineering Talents — A Case Study of Hefei Normal University. Journal of Hefei Normal University, 43(6): 118–122.

[6] Qin L, 2024, Intelligent Classification Method for University Career-Planning Course Teaching Materials Based

on Multiple Clustering Algorithms. Journal of Hebei North University (Social Science Edition), 40(3): 86–89 + 109.

[7] Bai S, Li J, Gao C, 2023, Analysis and Recommendation Model for University Elective Courses Based on Apriori Algorithm. Journal of Kaifeng University, 37(4): 70–75.

[8] Zhang L, Tong Y, Ma H, 2023, Resource-Sharing and Mutual-Aid System for University Curriculum Ideology-Politics Based on Chinese Word-Segmentation Algorithm and Crowdsourcing Collaboration. Modern Computer, 29(14): 91–95.

[9] Jiang M, Hu S, Meng X, 2023, Research on University Student Teaching-Evaluation Indicators Based on Random Forest Algorithm — Taking the Course Fundamentals of Programming Design as an Example. Journal of Kaili University, 41(3): 74–81.

[10] Dong J, Zhang Y, Hu Y, 2022, Design and Application of Learning-Outcome Prediction Model Based on Decision-Tree Algorithm — Taking the Calculus Course of a Certain University as an Example. Journal of Guangzhou Radio & TV University, 22(1): 39–46 + 109.

[11] Zhang Q, Zhang L, Wei M, 2021, Research on Algorithm and Implementation of University Computer-Aided Course-Scheduling System. China Electric Power Education, (S1): 256–257.

[12] Zhang X, Jiang H, 2021, Research on Construction of Online-Video Library for University Computer Courses Based on Scrapy Technology. Computer Knowledge and Technology, 17(36): 37–39.

[13] Xiang Y, 2021, Research on Innovation of Ideological-Political Theory Courses in Universities in the Era of Algorithm Recommendation. Social Sciences, (12): 70–80.

[14] Yang H, Cai J, Zhao X, 2021, Core Teaching Philosophy of Algorithm and Data-Structure Courses in Contemporary Universities — Selection and Design. Journal of Higher Education, 7(29): 108–111.

[15] Shang C, Dong Z, Wen W, 2021, Research on Artificial-Intelligence Algorithm Teaching for Application-Oriented Universities. Computer Knowledge and Technology, 17(27): 198–200.

Publisher's note

Bio-Byword Scientific Publishing remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.