Research on a Closed-Loop Multi-Agent Collaborative Decision-Making and Personalized Knowledge Generation System for Vertical Domains

Yonghui Zheng[1], Ruiheng Zhang[1], Chao Geng[1], Xiaojun Fan[1], Hanxiao Chang[2], Mingming Gong[3,*]

[1]College of Information Science and Engineering, Henan University of Technology, Zhengzhou, Henan, China

[2]College of Artificial Intelligence and Big Data, Henan University of Technology, Zhengzhou, Henan, China

[3]iFLYTEK Co., Ltd., Hefei, Anhui, China

**Corresponding Author*

Abstract: Traditional online learning systems still have several limitations. These limitations include inflexible learning paths, coarse-grained resource recommendations, delayed feedback, and limited intelligent interaction. This study develops a personalized learning system. The system combines multi-agent collaboration with dynamic learner profiles. The system first identifies learner requests through an intent recognition module. The intent recognition result determines which agent should handle the task. Different agents are responsible for different learning activities. These agents include the Question-Answering Agent, Path Planning Agent, Resource Recommendation Agent, Question Bank Generation Agent, Document Parsing Agent, Code Tutoring Agent, and Learning Evaluation Agent. The system updates the learner profile according to learning behaviors, knowledge mastery states, learning goals, and knowledge gaps. The learner profile is combined with the knowledge graph. This combination supports staged path planning and resource recommendation. The system uses error-feedback mechanisms, Feynman learning logs, and event-triggered mechanisms to build a closed learning loop. The loop connects diagnosis, path generation, resource recommendation, and adaptive feedback adjustment. Experiments and case studies show that the system can generate more reasonable learning paths and provide better resource recommendations. The proposed system provides a practical reference for personalized intelligent learning systems.

Keywords: Multi-Agent Systems; Personalized Learning; Learner Profile; Learning Path Planning; Resource Recommendation; Intelligent Education.

1. Introduction

Personalized learning represents a key direction in the development of intelligent education systems, capable of delivering differentiated learning support based on learners' foundational levels, learning goals, and process feedback [1-3]. At present, many online learning platforms already offer functions such as course resource browsing, exercise practice, and basic question-answering; however, they still suffer from rigid learning paths, insufficiently targeted resource recommendations, and underutilization of learning feedback [4, 5]. In computer science courses in particular, students differ considerably in their prior knowledge, practical ability, and areas of weakness, making it difficult for single-turn question-answering or fixed resource delivery to meet the demands of sustained and personalized learning [6, 7].

Personalized learning systems must rely on techniques such as learner profiling, multi-agent collaboration, and knowledge path planning to continuously model student states and to connect learning diagnosis, path generation, resource recommendation, and feedback adjustment into an integrated pipeline [8-12]. Building on this, the present paper proposes a personalized learning system based on multi-agent collaboration and dynamic learner profiling, capable of achieving intelligent automation across the entire workflow of "learning needs identification — dynamic

profile construction — path planning — resource recommendation — exercise feedback" [13-16]. The overall system architecture is shown in Figure 1.

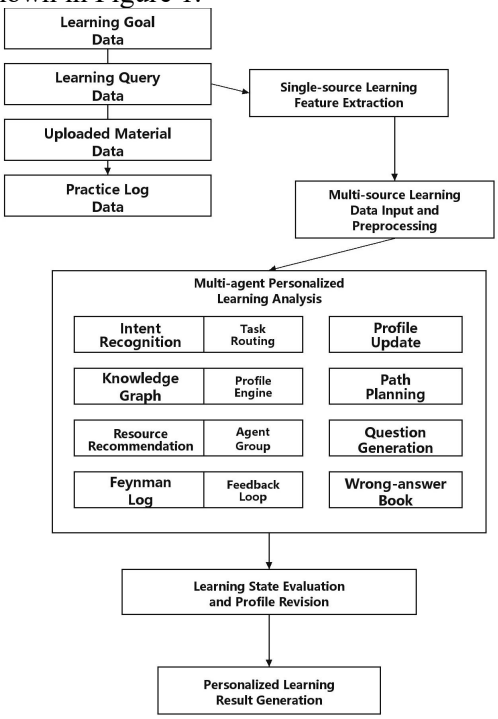

Figure 1. Overall Architecture of the Personalized Learning System

(1) Learning data collection and dynamic profile construction. The system collects data including learners' learning goals, historical dialogues, resource browsing behavior, exercise performance, error logs, learning journals, and uploaded materials, and then organizes and analyzes these heterogeneous learning data. Specifically, learning goals and historical dialogues are used to determine students' current learning needs; exercise results and error logs are used to identify weak knowledge points; and resource browsing behavior and learning journals are used to characterize learning preferences. By continuously updating learner profiles, the system enables personalized learning path planning and resource recommendation [6-9].

(2) Multiple agents collaboratively handle different types of learning tasks. The system first identifies the task category from learner inputs through the intent recognition module and then activates the corresponding agents according to the identified task. Specifically, the Question-Answering Agent is responsible for concept explanation and problem solving, while the Path Planning Agent generates staged learning routes. The Resource Recommendation Agent matches suitable learning materials, the Question Bank Generation Agent creates practice exercises, and the Document Parsing Agent processes uploaded materials. In addition, the Code Tutoring Agent performs program analysis and error localization, while the Evaluation Agent assesses learning effectiveness. Under a unified scheduling mechanism, these agents work collaboratively, enabling the system to handle diverse types of learning demands [13-15].

(3) Personalized learning service generation based on a feedback closed loop. The system generates personalized learning paths by combining learner profiles with knowledge-point association relationships, and recommends corresponding learning resources and practice content according to the current learning stage. During the learning process, the system identifies new weak areas through error records, learning journals, and stage evaluation results, further triggering profile updates, resource adjustments, and path re-planning. Through this mechanism, the system connects learning diagnosis, learning process, and learning feedback to form a continuously optimizing personalized learning closed loop [10-12, 16].

Through the above design, the system is able to provide students with continuous support spanning from learning needs identification to learning process feedback, offering a viable approach for the design and application of personalized intelligent learning systems.

2. Methods

2.1 Learner Profile Updating and Feature Construction

Learner profiles serve as the foundation of personalized learning services. The system collects data such as learning goals, historical dialogues, exercise performance, error records, and resource usage logs to continuously characterize learners' knowledge mastery, learning preferences, and weak areas, thereby constructing an updatable dynamic profile.

2.1.1 Data Description

The learning data in the system originate primarily from three types of scenarios: first, learning interaction data, such as questions, answers, and dialogue records; second, learning behavior data, such as exercise accuracy, error counts, and resource

click-through statistics; and third, process feedback data, such as review records, learning journals, and stage evaluations. Together, these data form the basis for learner profile updating. Examples of learning interaction data are shown in Figure 2, examples of learning behavior data are shown in Figure 3, and examples of error and feedback data are shown in Figure 4.

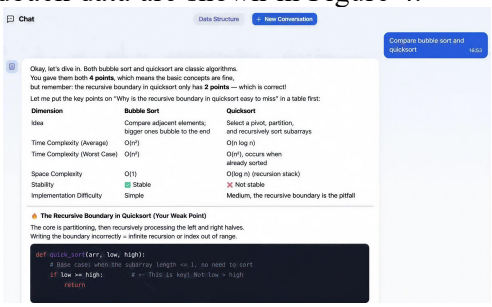

Figure 2. Examples of Learning Interaction Data

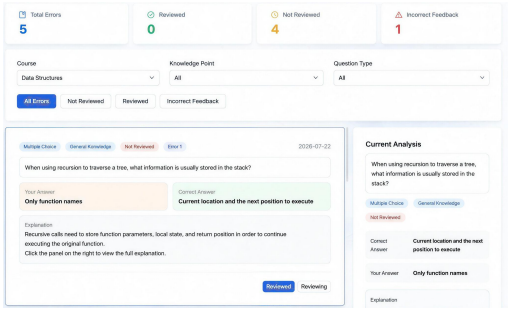

Figure 3. Examples of Learning Behavior Data

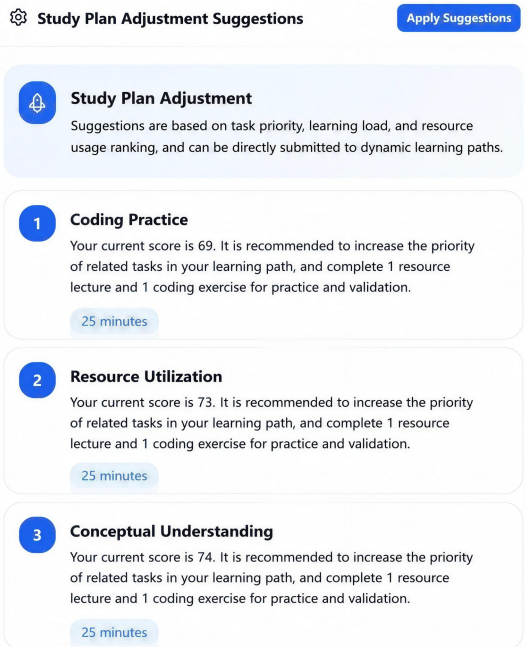

Figure 4. Examples of Error and Feedback Data

2.1.2 Profile Updating

For a knowledge point k, the system maps different learning behaviors to learning signals s_j, assigning distinct weights to actions such as correctly answering a high-difficulty question, correctly answering a medium-difficulty question, completing a learning resource, and reviewing after an error. The dynamic learner profile updating process is illustrated in Figure 5. The recent signals for the same knowledge point are averaged and then combined with the historical mastery level through a smoothing update, which can be expressed as:

$$\bar{s_k} = \frac{1}{n_k}\sum_{j=1}^{n_k} s_j \quad (1)$$

$$M_k^{(t)} = clip\left(\alpha M_k^{(t-1)} + (1-\alpha)\bar{s_k}, 1,5\right) \quad (2)$$

where $M_k^{(t)}$ denotes the learner's mastery level of knowledge point k at time t, $M_k^{(t-1)}$ denotes the historical mastery level, and α is a smoothing coefficient, set to 0.7 in this paper.

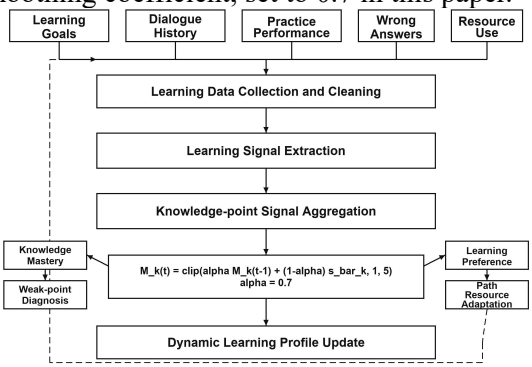

Figure 5. Flowchart of Dynamic Learner Profile Updating

2.2 Multi-Agent Collaboration and Path Planning

Building on the dynamic learner profile updating, the system further constructs a multi-agent collaboration mechanism to convert user learning requests into executable learning tasks. The system first determines the user's current needs through an intent recognition module, and then a task scheduling module selects and engages the appropriate agents — including Question-Answering, Path Planning, Resource Recommendation, Question Bank Generation, Document Parsing, Code Tutoring, and Learning Evaluation — to participate in the processing. These agents share data around a unified user profile, course knowledge graph, and learning records, enabling the system to deliver continuous services spanning from needs understanding and content generation to learning feedback. The multi-agent collaboration architecture is shown in Figure 6.

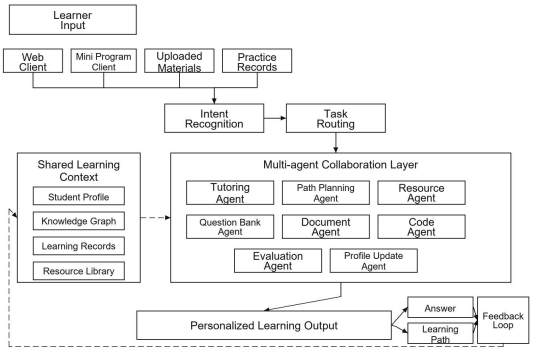

Figure 6. Multi-Agent Collaboration Architecture

2.2.1 Agent Scheduling

The system's agent architecture is constructed top-down from five layers: an input layer, an intent recognition layer, a task scheduling layer, an agent execution layer, and a result feedback layer. First, the system collects user-input learning demands, learning goals, uploaded materials, and assessment results through a web page or mini-program. The intent recognition module first classifies and parses the text to identify the core task type — knowledge-based question-answering, path planning, resource recommendation, question bank exercises, document parsing, code tutoring, or learning evaluation. Based on the intent classification results, the task scheduling module activates the target agents and provides prerequisite parameters such as the user profile, knowledge-point mastery status, and course context.

At the agent execution layer, different agents work collaboratively: the Question-Answering Agent handles knowledge analysis and heuristic guidance; the Path Planning Agent generates staged learning routes by combining the knowledge graph with the learner profile; the Resource Recommendation Agent matches relevant materials based on weak points and preferences; the Question Bank Generation Agent constructs practice exercises and stage assessment questions; the Document Parsing and Code Tutoring agents are responsible for multimodal material analysis and programming problem debugging; and the Evaluation Agent outputs learning diagnostics based on assessment results and interaction trajectories. Different agents generate responses. These responses provide immediate feedback during the current interaction. These responses are stored in dialogue logs, learner profiles, learning paths, error notebooks, and resource interaction records. The stored information is used for later strategy updates and personalized recommendations. The task scheduling mechanism across the system layers is illustrated in Figure 7.

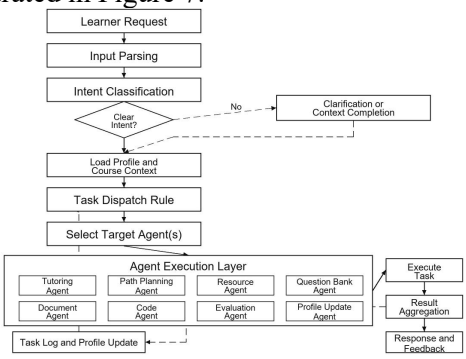

Figure 7. Agent Task Scheduling Flowchart

2.2.2 Five-Stage Path Planning

After the agents complete task identification, the Path Planning Agent generates a personalized learning path based on the learner profile and the course knowledge structure. The system abstracts course knowledge points and their prerequisite relationships into a knowledge graph, and then filters out the knowledge nodes that still require further study by considering the learner's already-mastered knowledge points. The five-stage path planning process is illustrated in Figure 8.

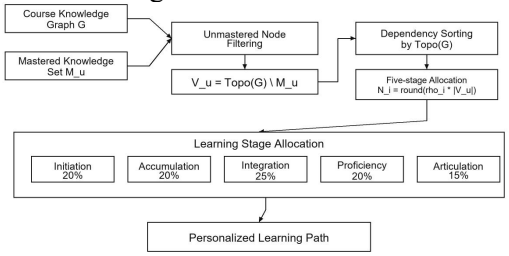

Figure 8. Five-Stage Learning Path Planning Flowchart

Let the course knowledge graph be denoted as G and the set of knowledge points already mastered by the learner be denoted as M_u; then the set of knowledge points yet to be learned can be expressed as:

$$V_u = Topo(G) \setminus M_u \tag{3}$$

where $Topo(G)$ denotes the topological sorting result of the knowledge graph, and V_u denotes the set of knowledge points that the learner still needs to study. The system does not arrange knowledge points in a simple linear order. It divides the remaining knowledge points into five stages: Enlightenment, Accumulation, Integration, Proficiency, and Articulation. These stages help learners gradually move from basic understanding to practical application. The number of knowledge points in each stage is

calculated as follows:

$$N_i = round(\rho_i \cdot |V_u|) \quad (4)$$

where $\rho = [0.20,0.20,0.25,0.20,0.15]$, corresponding to the five stages of Enlightenment, Accumulation, Integration, Proficiency, and Articulation, respectively.

2.2.3 Star-Level Threshold and Feedback Adjustment

A star-level threshold mechanism is introduced during path generation. It prevents the learning sequence from following a completely fixed order. The mechanism determines learning requirements based on knowledge-point difficulty, weak knowledge points, and current mastery levels. Figure 9 shows the star-level threshold and feedback adjustment process.

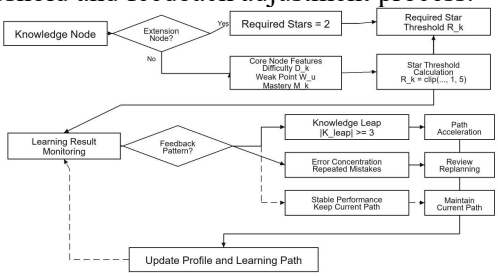

Figure 9. Star-Level Threshold and Feedback Adjustment Flowchart

The system sets a learning threshold for each knowledge point. The threshold is determined by the difficulty level, weak knowledge points, and the learner's current mastery level. For extension nodes, the default required star level is set to 2. For other nodes, the required star level is calculated as follows:

$$R_k = clip(D_k + 1 - \mathbb{I}(k \in W_u) - \mathbb{I}(M_k \geq 4), 1,5) \quad (5)$$

where R_k represents the required star level of knowledge point k. D_k represents the difficulty level of knowledge point k. W_u represents the learner's weak knowledge points. $I(\cdot)$ represents an indicator function. When a learner makes rapid progress on several knowledge points within a short period, the learning path is accelerated. For consecutive errors, the task difficulty is reduced and the review stage is rearranged. The trigger condition for a knowledge leap is given as follows:

$$K_{leap} = \left\{k | M_k^{(t-1)} \leq 2, M_k^{(t)} \geq 3.5\right\} \quad (6)$$

When $|K_leap| \geq 3$, the system increases the path difficulty or accelerates the learning pace. This condition indicates that the learner has gained enough knowledge to move to the next stage. When consecutive errors appear in adjacent knowledge points or within the same learning stage, the difficulty of later nodes is reduced. The system also adds review sessions or supplementary resources. These adjustments update the learning path according to changes in learner states. They also form a closed loop of "path generation — learning feedback — profile updating — path re-planning."

2.2.4 Database Storage Design

Django ORM is used to model and manage the core business data. These data are related to learner profile updates, agent task scheduling, and personalized path planning. The database contains four main categories of data: learners, course knowledge structures, learning paths, and process feedback. The database records learning activities during the learning process. It also updates related states and stores feedback information.

The database stores four main categories of information. Learner profile data record knowledge mastery, weak knowledge points, learning preferences, and stage evaluation results. Course knowledge structure data describe knowledge points and their prerequisite relationships. Learning path data record path stages, node sequences, required star levels, and completion statuses. Process feedback data store exercise results, error logs, learning journals, and resource usage statistics. These data are linked with the profile module and path planning module. Learning behaviors update the learner profile and influence later path planning and resource recommendation. This process forms a closed loop of "data recording — profile updating — path adjustment — learning feedback." The database E-R diagram is shown in Figure 10.

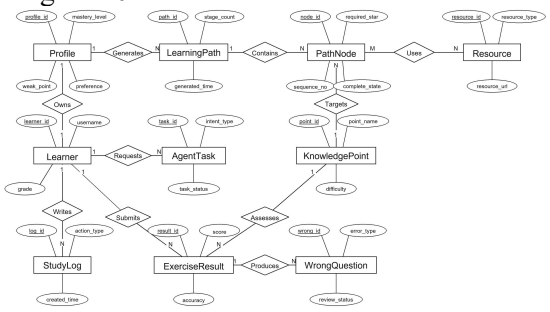

Figure 10. Database E-R Diagram

2.2.5 Learning Resource Retrieval and Generation Control

To improve the alignment between agent responses and the learner's current state, the system introduces a learning resource retrieval and context organization mechanism when generating learning suggestions,

question-answering content, and practice tasks. This mechanism takes the learner's question as input and, by combining the user profile, course knowledge points, historical learning records, error information, and resource usage statistics, filters out content relevant to the current learning task. The filtered content is then fed as context into the generation module, thereby reducing the problem of large-model-generated content deviating from the learning objectives. The learning resource retrieval and generation process is illustrated in Figure 11.

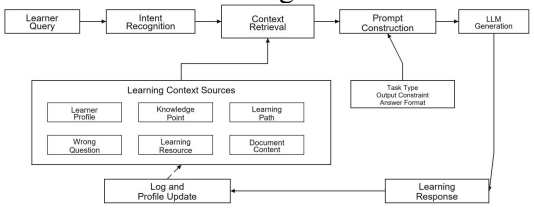

Figure 11. Learning Resource Retrieval and Generation Flowchart

The system first performs intent recognition on the learner's input to determine whether it belongs to task types such as knowledge-based question-answering, resource recommendation, path planning, question generation, document parsing, or learning evaluation. Based on the recognition result, the corresponding agent is invoked, and relevant context is extracted from the learner profile, knowledge-point structure, learning path nodes, error logs, and resource repository. For knowledge-based question-answering tasks, the system focuses on incorporating relevant knowledge points, historical dialogues, and learning stage information. For resource recommendation tasks, the system filters learning resources by combining weak knowledge points with the current path nodes. For question generation tasks, the system produces practice content according to knowledge-point difficulty, mastery level, and required star level.

At the generation stage, the system constrains the model output through prompt engineering to ensure that the generated responses remain consistent with the learning scenario. The prompts primarily include information such as the learner's current goals, knowledge-point names, mastery status, weak areas, task type, and output format requirements. In this way, the system is able to transform scattered learning data into structured context that can be understood by large models, and generate question-answering, recommendation, and feedback content that better aligns with individual learning needs.

Unlike a system that relies only on generic large-model responses, this mechanism allows agent outputs to better reflect the learner's current learning progress. The generated results are recorded in the learning journal, practice records, or profile updating module. These records support subsequent path adjustment and resource recommendation and complete the closed loop of "learning request — resource retrieval — content generation — result feedback — profile updating."

3. Application, Validation, and Analysis

3.1 Experimental Data and Validation Setup

The experimental dataset was built from actual system interaction data and simulated learning tasks, including learner interaction records, dynamic learner profiles, course knowledge-point graphs, test errors, and agent task logs, representing the learning process from needs articulation to feedback adjustment.

For the experimental setup, this paper uses six representative scenarios for system validation: knowledge-based question-answering, path planning, resource recommendation, exercise generation, error feedback, and academic evaluation. The validation follows the system workflow: "intent recognition — agent execution — learning status feedback — strategy optimization."

The system performance is evaluated using four metrics: profile updating accuracy, path planning rationality, resource recommendation relevance, and agent response effectiveness. The results reflect the system's usability and stability in personalized learning assistance and multi-agent collaboration.

3.2 System Functional Validation and Results Analysis

Four metrics are used to evaluate system performance: profile updating accuracy, path planning rationality, resource recommendation relevance, and agent response effectiveness. The results of these evaluations are presented in the corresponding charts, as shown in Figure 12. To further examine the effectiveness of the proposed system, a baseline system was introduced. The baseline adopts a single-agent learning interaction framework, where resource recommendation relies on fixed-path rules without dynamic learner profiles or the

closed-loop error feedback mechanism.

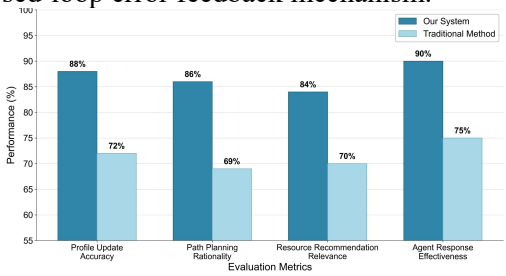

Figure 12. System Functional Validation Metrics

Profile updating accuracy: The learner profile is dynamically updated based on exercise results, error logs, review behaviors, and historical interaction records. By comparing knowledge-point mastery and weak-point labels before and after the update, the results indicate that the profile can reflect changes in the learner's state and provide guidance for later path planning.

Path planning rationality: When generating learning paths, the system combines the course knowledge graph, mastered knowledge points, and weak knowledge points to perform path planning. The results indicate that the five-stage path planning method preserves the prerequisite relationships among knowledge points and adjusts the learning nodes according to the learner's mastery status, thereby endowing the learning path with strong continuity and personalized characteristics.

Resource recommendation relevance: The system filters learning resources according to the learner profile, current path node, and weak knowledge points. An analysis of the matching between recommended resources and target knowledge points shows that the recommendation results are centered around the learner's current learning task, providing the learner with well-targeted resource support.

Agent response effectiveness: The system performs agent scheduling for tasks such as knowledge-based question-answering, path planning, resource recommendation, question generation, document parsing, and learning evaluation. The results show that the multi-agent collaboration mechanism covers the major learning scenarios and returns reasonably complete processing results across different types of learning requests.

3.3 Personalized Learning Case Analysis

Taking a learner with basic programming ability who wishes to study data structures systematically as an example: the learner first inputs the learning goal of "systematically studying data structures and improving algorithmic problem-solving ability." Based on the learner's historical interaction records, baseline test results, and learning preferences, the system generates an initial learner profile and identifies mastery differences across knowledge points such as linear lists, stacks , queues, tree structures, and graph structures. Subsequently, the Path Planning Agent combines the course knowledge graph with the learner profile, filters out already-mastered knowledge points, and generates a personalized learning path according to the five stages of Enlightenment, Accumulation, Integration, Proficiency, and Articulation. Figure 13 presents the personalized data structure learning path generated by the system.

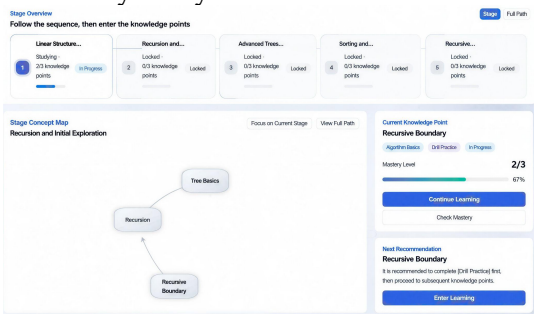

Figure 13. Personalized Learning path Generation Results for Data Structures

During the learning process, the system continuously updates the profile information based on the learner's post-exercise accuracy, error types, and review records. When the learner makes consecutive errors on knowledge points such as linked list insertion and deletion, recursive traversal, and binary tree properties, the system marks these knowledge points as weak areas and recommends corresponding explanatory resources and supplementary exercises. When the learner's mastery of knowledge points such as stacks, queues and binary tree traversal improves significantly, the system adjusts the required star levels of subsequent path nodes or accelerates the learning pace. Figure 14 shows the profile updating results after learning feedback.

From this case, it can be seen that the system is able to perform profile construction, path planning, resource recommendation, exercise feedback, and profile updating around the learner's learning goal. Compared with a fixed curriculum sequence, the personalized learning path dynamically adjusts the learning content

according to changes in the learner's state, making the learning process better aligned with individual differences. This case validates the system's usability in the data structures learning scenario and also demonstrates that the multi-agent collaboration mechanism is capable of supporting a continuous service closed loop throughout the learning process.

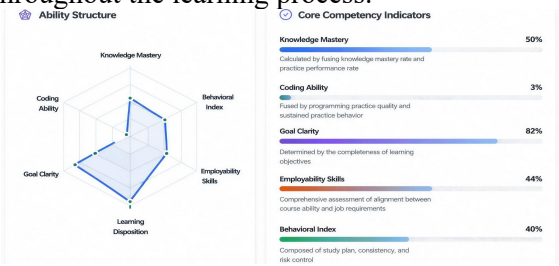

Figure 14. Learning Feedback and Profile Updating Results for Data Structures

3.4 System Application Effectiveness Simulation

3.4.1 Learner Profile Updating Simulation

To further verify the system's dynamic feedback capability during the learning process, this paper takes the data structures learning task as an example and conducts a simulation analysis of the learner profile updating process. The simulation subject is a learner with basic programming ability whose initial learner profile indicates a certain foundation in linear lists and stacks and queues, but a relatively low mastery level in knowledge points such as linked list operations, binary tree traversal, and graph search.

After the learner completes a set of data structure exercises, the system updates the learner profile according to exercise accuracy, error distribution, and review records. The updated profile shows clear improvement in basic topics, such as linear lists and stacks and queues. Linked lists and binary trees also show some improvement. However, graph search remains difficult for the learner. Figure 15 compares the mastery levels of key knowledge points before and after the exercises.

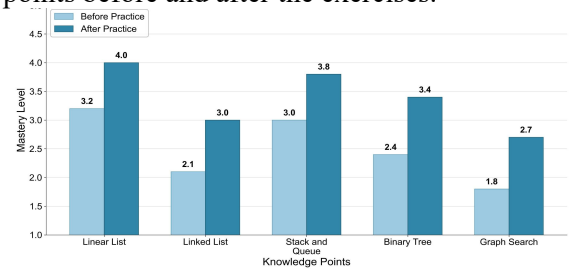

Figure 15. Learner Profile Updating Simulation Results

Figure 15 presents the changes in knowledge-point mastery based on learner exercise feedback. For knowledge points with improved mastery, the system reduces review frequency or increases the required star level. For knowledge points with limited improvement, the system keeps them as weak areas. It also recommends supplementary explanations and targeted exercises. These changes verify that the learner profile updating mechanism reflects learner-state changes. The updated profile provides information for later path adjustment and resource recommendation.

3.4.2 Personalized Path Adjustment Simulation

To verify the system's effectiveness in dynamically adjusting learning paths, this paper takes the data structures learning task as an example and conducts a simulation analysis of the personalized path adjustment process. The system continuously receives exercise results, error records, and profile updating results during the learning process, and determines whether subsequent learning paths need to be adjusted based on changes in knowledge-point mastery levels. When the learner's mastery of multiple knowledge points improves significantly within a short period, the system determines that a knowledge leap has occurred and triggers path acceleration; when the learner makes consecutive errors on knowledge points of the same category, the system triggers review re-planning or reduces the difficulty of subsequent nodes.

In the simulation scenario, after the learner completes exercises on linear lists, stacks and queues, and binary tree traversal, the mastery levels of the relevant knowledge points all improve. Several fundamental nodes meet the mastery requirement. The learner then moves to later stages of the learning path. Repeated errors appear in graph search and recursive application exercises. These knowledge points are marked as weak areas. The system provides additional explanations and targeted exercises for them. Figure 16 presents the system's path adjustment results under different learning feedback conditions.

It can be observed from Figure 16 that the system is able to dynamically process the path according to changes in the learner's state. For knowledge points with significant mastery improvement, the system raises the learning stage or shortens repetitive exercises; for knowledge points where errors cluster, the

system reduces task difficulty and arranges review nodes. These simulation results indicate that the personalized path adjustment mechanism can avoid the problem of fixed learning paths being unable to accommodate individual learner differences, making the learning arrangement more consistent with the learner's actual state.

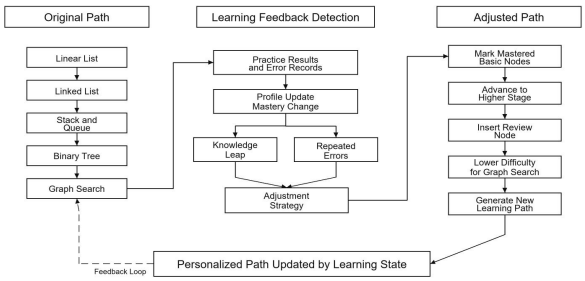

Figure 16. Personalized path adjustment simulation results

4. Conclusion

This study addresses the application of artificial intelligence technologies in personalized learning scenarios and presents a multi-agent personalized learning system that covers the entire learning process. The system integrates learner profile updating, agent task scheduling, five-stage path planning, and learning resource retrieval and generation control into a closed-loop service workflow of "learning request — profile construction — path generation — resource recommendation — exercise feedback — dynamic adjustment," providing a practical reference for applying AI technologies in educational assistance scenarios

At the system design level, learner profiling is integrated with a multi-agent collaboration mechanism, allowing the system to dynamically characterize learning states based on learners' goals, historical interactions, exercise performance, and error records. Through intent recognition and task scheduling, the system is able to assign diverse tasks — including knowledge-based question-answering, path planning, resource recommendation, question generation, document parsing, code tutoring, and learning evaluation — to the corresponding agents, which improves the system's adaptability to multiple types of learning demands.

The path planning module uses a five-stage learning path generation method based on knowledge graphs and learner profiles. The module selects learning content according to the learner's mastered knowledge points, weak areas, and knowledge-point difficulty. The selected content forms a five-stage learning path: Enlightenment, Accumulation, Integration, Proficiency, and Articulation. The star-level threshold and feedback adjustment mechanism modify the learning path according to learner changes. Knowledge leaps and consecutive errors trigger different path adjustments. The system accelerates the learning path after knowledge leaps. It replans review stages or reduces task difficulty when consecutive errors occur. These adjustments make the learning path change with the learner's state instead of following a fixed sequence.

The system was tested in a data structures learning scenario through functional tests, case studies, and simulation experiments. The experiments show changes in learner profiles, learning paths, resource recommendations, and feedback strategies under different learner conditions. The system provides personalized learning assistance based on learner conditions. Fixed learning workflows use predefined task arrangements. In contrast, the proposed approach adjusts later tasks according to learner performance. This adjustment makes the learning process closer to individual learning needs and progress.

Future work will collect more real-world learning data and test the system across different courses and learner groups. The system will also analyze learning behaviors in more detail. This analysis will help improve learner profile modeling and path adjustment stability. The collaboration strategies among agents and the quality evaluation of generated content will also be studied. These improvements aim to make the system more practical for real teaching assistance scenarios.

Acknowledgments

The authors thank iFLYTEK Co., Ltd. for its technical support and valuable resources during this research. The authors also thank the anonymous reviewers and editors for their valuable comments and suggestions. These suggestions helped improve the presentation and quality of this paper.

References

[1] Crompton H, Burke D. Artificial intelligence in higher education: the state of the field. International Journal of Educational Technology in Higher Education, 2023, 20: 22. DOI: 10.1186/s41239-023-00392-8.

[2] Zhang K, Aslan A B. AI technologies for education: Recent research & future directions. Computers and Education: Artificial Intelligence, 2021, 2: 100025. DOI: 10.1016/j.caeai.2021.100025.

[3] Ouyang F, Jiao P. Artificial intelligence in education: The three paradigms. Computers and Education: Artificial Intelligence, 2021, 2: 100020. DOI: 10.1016/j.caeai.2021.100020.

[4] Luo J, Zheng C, Yin J, et al. Design and assessment of AI-based learning tools in higher education: a systematic review. International Journal of Educational Technology in Higher Education, 2025, 22: 42. DOI: 10.1186/s41239-025-00540-2.

[5] Kabudi T, Pappas I, Olsen D H. AI-enabled adaptive learning systems: A systematic mapping of the literature. Computers and Education: Artificial Intelligence, 2021, 2: 100017. DOI: 10.1016/j.caeai.2021.100017.

[6] Brusilovsky P, Millán E. User models for adaptive hypermedia and adaptive educational systems//Brusilovsky P, Kobsa A, Nejdl W. The Adaptive Web. Berlin: Springer, 2007: 3-53. DOI: 10.1007/978-3-540-72079-9_1.

[7] Khosravi H, Shum S B, Chen G, et al. Explainable Artificial Intelligence in education. Computers and Education: Artificial Intelligence, 2022, 3: 100074. DOI: 10.1016/j.caeai.2022.100074.

[8] Piech C, Bassen J, Huang J, et al. Deep Knowledge Tracing//Advances in Neural Information Processing Systems 28. 2015: 505-513.

[9] Abdelrahman G, Wang Q, Nunes B. Knowledge Tracing: A Survey. ACM Computing Surveys, 2023, 55(11): 1-37. DOI: 10.1145/3569576.

[10] Hogan A, Blomqvist E, Cochez M, et al. Knowledge graphs. ACM Computing Surveys, 2021, 54(4): 71:1-71:37. DOI: 10.1145/3447772.

[11] Drachsler H, Verbert K, Santos O C, et al. Panorama of recommender systems to support learning//Ricci F, Rokach L, Shapira B. Recommender Systems Handbook. Boston: Springer, 2015: 421-451. DOI: 10.1007/978-1-4899-7637-6_12.

[12] Tarus J K, Niu Z, Mustafa G. Knowledge-based recommendation: a review of ontology-based recommender systems for e-learning. Artificial Intelligence Review, 2018, 50: 21-48. DOI: 10.1007/s10462-017-9539-5.

[13] Wooldridge M, Jennings N R. Intelligent agents: theory and practice. The Knowledge Engineering Review, 1995, 10(2): 115-152. DOI: 10.1017/S0269888900008122.

[14] Dorri A, Kanhere S S, Jurdak R. Multi-agent systems: A survey. IEEE Access, 2018, 6: 28573-28593. DOI: 10.1109/ACCESS.2018.2831228.

[15] Kasneci E, Sessler K, Küchemann S, et al. ChatGPT for good? On opportunities and challenges of large language models for education. Learning and Individual Differences, 2023, 103: 102274. DOI: 10.1016/j.lindif.2023.102274.

[16] Lewis P, Perez E, Piktus A, et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks//Advances in Neural Information Processing Systems 33. 2020.

