ingegneria sismica
International Journal of Earthquake Engineering
Anno XLIII - Num. 2 - 2026

# Dynamic Generation of Personalized Adaptive Learning Paths for College English driven by Reinforcement Learning

*Min Xu*[1,*] *and Liang Yin*[2]

[1] Department of Fundamental Subjects of Sichuan University of Architectural Technology, Deyang 618000, Sichuan China
[2] Library & Campus Information Technology Center, Sichuan Institute of Industrial Technology, Deyang 618000, Sichuan, China

**SUMMARY:** *Aiming at the problems of lag of state update, rough matching of candidate resources and unstable strategy convergence in personalized path generation, an adaptive path dynamic generation model driven by reinforcement learning was constructed. The model encodes user behavior sequence, resource features, knowledge status and feedback records into a 128-dimensional state vector. GRU is used to extract historical interaction features, and knowledge dependence matrix is combined to compress the candidate action space. The reward function integrates path revenue, resource matching degree, completion feedback and load penalty, and the update range of the strategy is constrained by the PPO cutting objective function. The experiment constructed a data set based on 12864 interaction records, 420 resource nodes and 96 knowledge units. The results show that the Accuracy of Proposed PPO reaches 93.5%, NDCG@10 is 0.881, Completion Rate is 89.7%, Mastery Gain is 22.4%, and Average Reward is 0.842. Compared with the DQN model, the relative improvement rates of Accuracy, NDCG@10 and Completion Rate are 6.74%, 8.50% and 8.86%, respectively. The average reward stabilizes after round 82. Experimental results show that the proposed model can improve the accuracy of dynamic path generation, the quality of resource ordering and the stability of strategy iteration.*

***KEYWORDS:*** *Reinforcement Learning; Adaptive Path Generation; PPO Algorithm; Knowledge State Estimation; Sequential Recommendation*

## 1 Introduction

Personalized adaptive path generation is a typical dynamic sequential decision making problem. Its goal is not to sort resources once, but to generate a resource access sequence with better long-term benefits under the conditions of continuous changes in user status, obvious differences in resource attributes, and delay in feedback signals. Traditional recommendation methods mostly rely on collaborative filtering, content matching or static rules, which can complete resource screening, but lack the ability to adapt to the change of user capabilities, periodic behavior fluctuations and continuous path constraints. Reinforcement learning recommender systems incorporate state, action, reward and policy update into a unified framework, and can optimize long-term path benefits through interactive feedback [1]. Research on learning path recommendation also shows that the path generation task needs to deal with target decomposition, resource ranking, state recognition and stage feedback

*15883422864@163.com
https://doi.org/10.65102/is20261005
Published: 30 April, 2026

correction at the same time, and a single recommendation is difficult to meet the dynamic optimization requirements in continuous scenarios [2].

User state representation is the core link that affects the quality of path generation. If the state vector only contains clicks, completion records or resource tags, it is difficult for the model to judge the difference of users 'mastery on different knowledge units, and it cannot identify the ability changes behind the behaviors such as low scores, jumping out, and repeated visits. Deep knowledge tracing research encodes historical answer sequences, knowledge point labels and interaction results into continuous states, which can estimate the probability of knowledge mastery from multiple rounds of behavior, and provide more refined input for subsequent resource selection [3]. Relevant reviews have pointed out that deep networks, attention mechanisms and graph structure modeling can improve the dynamics and interpretability of knowledge state estimation, so that the model can update the user profile according to real-time interaction results [4].

The existing research on personalized path recommendation has shifted from static resource matching to procedural path modeling. Related frameworks generate personalized paths through user portraits, resource attributes and goal constraints, but some methods are still mainly based on heuristic rules or supervised learning ranking, and lack a strategy update mechanism for long-term benefits [5]. The combination of process mining and deep knowledge tracing can extract path patterns from historical trajectories, and adjust the recommendation order according to changes in knowledge mastery [6]. Multi-behavior modeling and cascaded deep Q network method jointly use browsing, answering, completion and feedback and other behaviors for path decision-making, which improves the processing ability of complex interaction sequences [7].

Reinforcement learning further enhances the ability of policy optimization in dynamic path generation. Offline reinforcement learning can learn the path policy based on historical interaction data, reduce the instability risk caused by online trial and error, and control the recommended action boundary through the constraint mechanism [8]. The graph reinforcement learning method organizes knowledge points, resources and user states into graph structures, and uses node relationships and path connectivity in strategy selection to improve the structural rationality of resource sequences [9]. Research on the fusion of dynamic knowledge tracking and reinforcement learning shows that embedding the knowledge state estimation results into the reward function and policy network can improve the real-time performance and the degree of personalization of path optimization [10]. Existing research has provided technical support for adaptive path generation, but there are still problems such as insufficient state representation, single reward goal, insufficient path coherence constraints, and unstable feedback updates.

This paper constructs a personalized adaptive path dynamic generation model driven by reinforcement learning. Taking multi-source behavior sequences, resource attributes, knowledge states and feedback results as inputs, the model established user state vector, resource candidate set, path action space and multi-objective reward function. In the policy optimization layer, PPO algorithm was introduced, and the range of policy update was controlled by cutting the objective function. In the experimental layer, rule recommendation, collaborative filtering, DQN model and ablation model are set as comparison objects, and the performance of the model is verified from the aspects of path generation accuracy, recommendation performance, knowledge state improvement, strategy convergence speed and dynamic update effect.

# 2 Computational modeling of the personalized path generation task

## 2.1 Structured coding of multi-source behavior sequence data

The input of personalized path generation is composed of behavior log, quiz record, resource attribute and feedback result. The original data sources are different, and there are differences in field scale, sampling frequency and missing conditions. If it is directly input into the policy network, it is easy to cause state space noise amplification, which affects the subsequent resource screening and action decision [11]. In this paper, we set up a structured coding link at the front end of the model, organize the scattered interaction records into unified time window samples, and map the continuous values, discrete labels and feedback states into the same quantity space, which provides the basic input for state representation, candidate action generation and reinforcement learning policy update.

Let the u-th user generate a resource interaction behavior at time t. The original observation vector is defined as follows.

$$x_{u,t}=\left[c_{u,t},d_{u,t},f_{u,t},q_{u,t},r_i,g_{u,t}\right] \tag{1}$$

where, cu,t represents the number of clicks, du,t represents the stay time, fu,t represents the completion status, qu,t represents the test score, ri represents the difficulty level of the ith resource, and gu,t represents the feedback score. The vector compresses the interaction intensity, execution result, resource attribute and feedback information into the same computing unit, which can reduce the index misalignment when multi-source fields are read separately. For the missing fields, the mean value of similar resources and the mean value of adjacent Windows of users are used to fill in, and the missing markers are retained for subsequent coding.

Because the dimensions of stay duration, quiz score, completion status, and feedback score are different, we use normalization with confidence weights to process the JTH field:

$$\hat{x}_{u,t}^{j}=\omega_j\cdot\frac{x_{u,t}^{j}-\min\left(x^j\right)}{\max(x^j)-\min(x^j)+\varepsilon} \tag{2}$$

where $\hat{x}_{u,t}^{j}$ are the standardized results, ωj are the feature confidence weights, min(xj) and max(xj) are the minimum and maximum values of this field in the training set, and ε is the smoothing term. The test score and completion status are more sensitive to the change of knowledge status, and can be set with higher weights. Subjective feedback is greatly affected by chance factors, and lower weights can be set to reduce the interference of noise fields on the state vector.

Path generation has the characteristics of continuous decision making, and it is difficult to reflect the change of user state in a single interaction. In this paper, a sliding window of length L is used to organize the normalized vector and form the behavior sequence matrix:

$$X_{u,t}^{L}=\left[\hat{x}_{u,t-L+1},\hat{x}_{u,t-L+2},\cdots,\hat{x}_{u,t}\right]\in R^{L\times m} \tag{3}$$

where $X_{u,t}^{L}$ represents the window behavior sequence of user u at time t, L represents the historical window length, and m represents the single-step feature dimension. The matrix keeps the last L interaction records in chronological order, and can retain dynamic patterns

such as continuous errors, fast completion, repeated access, and low feedback. The window length is set to 12 in the experiment, and the early sequences with insufficient length are processed by zero-padding.

To clarify the coding objects and computational purposes of different data after entering the model, Table 1 summarizes the multi-source behavior sequence data fields.

*Table 1: Multi-source Behavior Sequence Data Fields and Encoding Methods*

| Data Type | Encoding Fields | Model Role |
|---|---|---|
| Behavior Log | Click Count, Dwell Time, Completion Status | Represent Interaction Intensity and Execution Result |
| Test Record | Test Score, Accuracy, Number of Wrong Questions | Estimate Stage Ability Change |
| Resource Attribute | Resource Difficulty, Knowledge Tag, Resource Type | Construct Candidate Action Space |
| Feedback Data | Rating, Number of Relearning Times, Bounce Status | Modify Reward and State Update |

After the multi-source fields have been uniformly encoded, the model input tensors need to be generated according to the fixed link. Figure 1 shows the processing process of the original data from access, cleaning, normalization to sliding window segmentation.

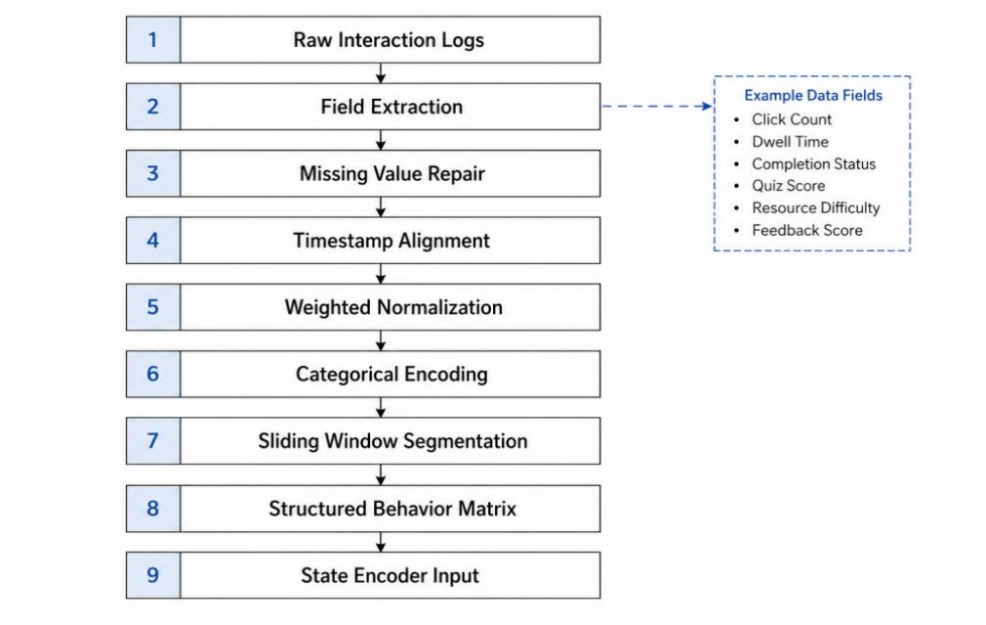


*Figure 1: Structured Coding Flowchart for Multi-source Behavioral Sequence Data*

## 2.2 User state vector and resource feature space construction

After encoding the behavior sequence, the model needs to map the user-side dynamic features and the resource-side static attributes into the same computational space. User state vector is used to describe current interaction ability, historical behavior trend and feedback change, and resource feature space is used to describe resource difficulty, knowledge label, type attribute and historical use effect [12]. If the two types of vectors are in different scales or different semantic Spaces, the reinforcement learning policy network is prone to matching deviation when selecting actions [13]. In this paper, we adopt embedding mapping and feature fusion to

uniformly represent user behavior sequences, knowledge states, feedback signals and resource attributes as computable vectors, which provide a basis for the state space and action space in the subsequent Markov decision process.

Let the behavior sequence matrix of user u at time t be $X_{u,t}^{L}$, the knowledge state vector be Ku,t, and the feedback feature be Bu,t. The user state vector is obtained by nonlinear mapping:

$$s_{u,t}=\sigma\left(W_x\cdot Pool\left(X_{u,t}^{L}\right)+W_k K_{u,t}+W_b B_{u,t}+b_s\right) \tag{4}$$

where, su,t represents the user state vector, Pool(·) represents the sequence pooling operation, Wx, Wk, Wb are trainable mapping matrices, bs is the bias term, and σ(·) is the nonlinear activation function. This representation combines short-term behavior, stage knowledge acquisition and feedback records into a unified state, so that the model can perceive the recent interaction strength and the direction of capability change simultaneously. In this paper, the user state vector dimension is set to 128, which is used to carry the state input of the subsequent policy network.

The resource-side features are composed of resource difficulty, knowledge label, resource type, historical completion rate and average feedback score. Let the original attribute vector of the ith resource be zi and its embedding be expressed as follows.

$$e_i=\phi(W_z z_i+b_z),\quad z_i=\left[\rho_i,\tau_i,\kappa_i,h_i,\bar{g}_i\right] \tag{5}$$

In the formula, ei represents the resource feature vector, ρi represents the resource difficulty, τi represents the resource type encoding, κi represents the knowledge label vector, hi represents the historical completion rate, and gi represents the average feedback score. Wz and bz are resource mapping parameters, and ϕ(.) is the embedding activation function. The dimension of the resource feature vector is set to 64 to reduce the sparsity of high-dimensional labels and enable different resources to be compared in similarity in a unified space.

The matching relationship between the user state and the resource features determines the quality of the ranking of candidate actions. This paper uses a bilinear matching function to calculate the adaptation score of resource i selected by user u at time t:

$$M_{u,i}^{t}=s_{u,t}^{T}W_m e_i+\lambda_1\left(1-\left|\alpha_{u,t}-\rho_i\right|\right)+\lambda_2 C_{u,i} \tag{6}$$

where, $M_{u,i}^{t}$ represent the matching score between users and resources, Wm is the matching matrix, αu,t represents the estimated value of user's current ability, ρi represents the difficulty of resources, Cu,i represents the coverage relationship between user status and resource knowledge labels, and λ1 and λ2 are weight coefficients. The function not only computes the vector space similarity, but also introduces the capability-difficulty difference and knowledge label coverage constraints, which can avoid the path instability caused by too easy or too hard resources or knowledge point jump.

To facilitate the illustration of the main components of user state vector and resource feature space, Table 2 presents the core parameter configuration adopted in this paper.

*Table 2: User State Vector and Resource Feature Space Parameter Configuration*

| Vector Type | Dimension/Field | Description |
|---|---|---|
| User State Vector | 128-dimensional | Fuse Behavior Sequence, Knowledge State and Feedback Features |
| Resource Feature Vector | 64-dimensional | Represent Difficulty, Type, Tag and Historical Feedback |
| Knowledge State Vector | 96-dimensional | Correspond to Mastery Probability of 96 Knowledge Units |
| Interaction History Vector | 12×6 | Save Feature of Recent 12 Interaction Windows |
| Matching Score | 1-dimensional | Output Adaptation Degree Between User and Candidate Resource |

After the unified representation of user status and resource features is completed, the mapping structure from input fields to matching scores needs to be formed. Figure 2 shows the encoding, fusion and matching process of the two types of vectors.

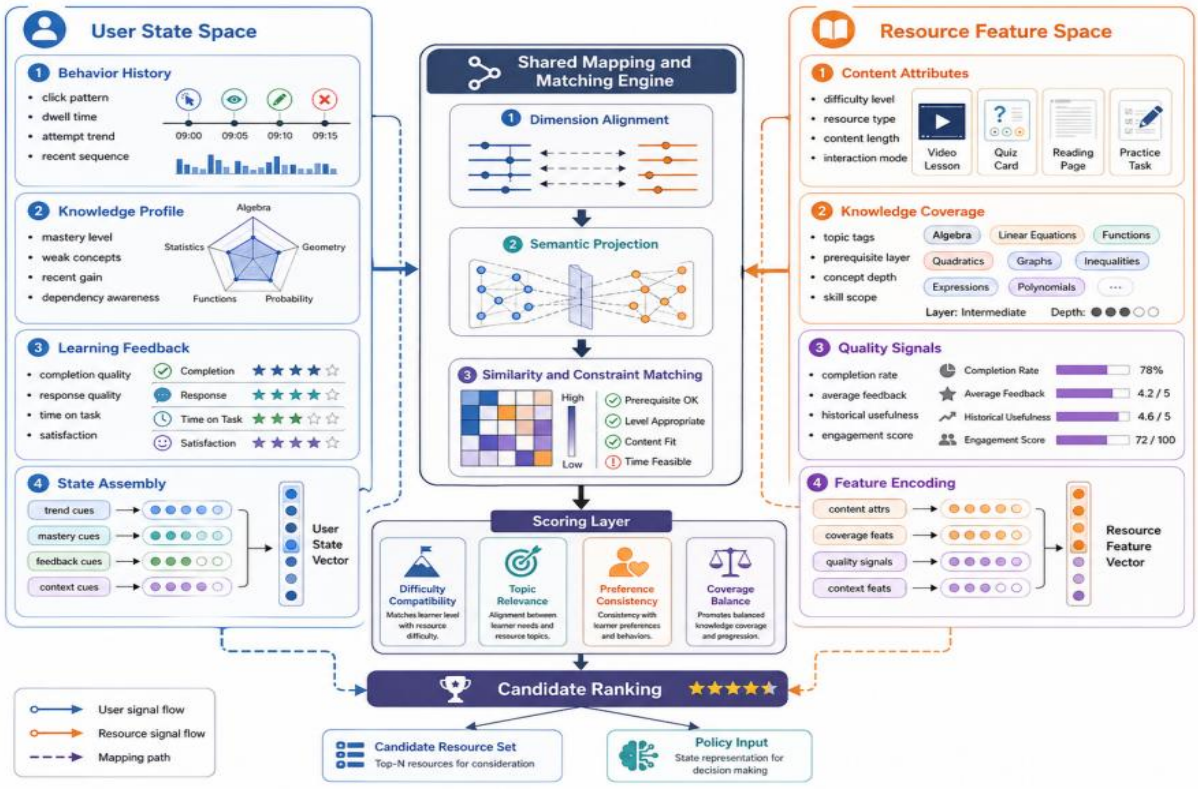


*Figure 2: Mapping structure diagram between user state vector and resource feature space*

Through the above modeling, the user-side dynamic state and resource-side attribute information are compressed into a unified vector space. The user state vector is responsible for expressing the current ability, behavior trend and feedback change, the resource feature vector is responsible for expressing the resource difficulty, knowledge coverage and history quality, and the matching function is responsible for converting the two into candidate action scores. This result can be directly used for state definition, action screening, and state transition computation in the Markov decision process modeling in the next section.

## 2.3 Markov Decision process modeling of the path generation task

The path generation task has a continuous decision attribute, and the current resource selection not only affects the feedback of the current round, but also changes the state representation and candidate resource range in the next round. If the ranking is only based on the instant matching score between users and resources, the model is easy to fall into local optimum, and it is difficult to deal with the coupling between resource precedence, state transition and long-term benefits [14]. In this paper, the path generation process is abstracted as a finite-domain Markov decision process, and state observation, resource action, state transition and revenue accumulation are unified into the reinforcement learning modeling framework, so that the path generation problem is transformed from a static sequencing problem to a trainable sequence optimization problem.

Let the path generation task be executed in a decision period of length T. The finite-time domain Markov decision process is defined as follows.

$$M_T=\left(S,A,P,R,\gamma,\mu_0,T\right) \tag{7}$$

where, MT represents the finitetime-domain path generation decision process, S represents the state space, A represents the action space, P represents the state transition probability function, R represents the reward function, γ represents the discount factor, μ0 represents the initial state distribution, and T represents the maximum number of decision steps for a single path. This definition only describes the overall decision elements of the path generation task, and the specific state content, action form and reward structure will be expanded in the subsequent model.

At each decision step, the system selects a path action from the candidate resource actions based on the current state. The state is carried by the user state vector constructed in the previous section, and the action corresponds to a certain candidate resource node or resource combination. To avoid searching directly on the whole resource set, we restrict the action space to the candidate set filtered by both matching scores and resource constraints. After the user performs an action, the system updates the next moment state according to the interaction result, the increment of knowledge state and the feedback signal. The state transition function is defined as follows.

$$s_{u,t+1}=\sigma\left(W_s s_{u,t}+W_a e_{a_t}+W_y y_{u,t}+W_\Delta \Delta k_{u,t}+W_f f_{u,t+1}+b_p\right) \tag{8}$$

where, su,t+1 represents the state vector of user u at time t+1, su,t represents the current state vector, eat represents the embedding vector of resource corresponding to action at, yu,t represents the result feature after action execution, Δku,t represents the change of knowledge state, fu,t+1 represents the updated feedback feature. Ws, Wa, Wy, WΔ, Wf are trainable mapping matrices, bp is the bias term, and σ(·) is the nonlinear activation function. The transition function integrates the original state, resource action, execution result and feedback correction into the next state generation process, which can express the impact of resource selection on subsequent path environment.

The goal of path generation is not to obtain the single-step highest matching score, but to maximize the long-term payoff over the complete decision cycle. In this paper, the path optimization objective under policy πθ is defined as follows.

$$J(\theta)=E_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1}\gamma^t\left(R(s_{u,t},a_t)-\eta C(s_{u,t},a_t)\right)\right] \tag{9}$$

where J(θ) represents the policy network optimization objective with parameter θ, τ~πθ represents the path trajectory sampled by policy πθ, γt represents the revenue discount weight at step t, R(su,t,at) represents the immediate reward obtained after performing action at, C(su,t,at) represents the path load or action cost. Let η denote the cost penalty weight. This objective function explicitly deducts the overload cost in the cumulative reward, so that the model avoids generating paths that are too long, too difficult, or with high feedback risk.

Through the above modeling, the path generation process forms a closed loop of "state observation -- action selection -- execution feedback -- state transition -- revenue accumulation". The action space corresponds to the executable resource nodes. The state transition function describes the impact of the execution result on the next round of state. The modeling results provide a formal basis for the subsequent reinforcement learning model construction. On this basis, the state fusion, candidate resource screening, reward function and PPO policy update methods will be further designed in the following paper.

# 3 Reinforcement learning driven dynamic path generation model construction

## 3.1 Multimodal behavior feature fusion and state representation

The input of the dynamic path generation model includes not only behavior sequences such as click, stay and finish, but also resource interaction results, knowledge state changes and feedback signals. The semantic density of different modal data is different, the behavior sequence reflects the user's recent operation trajectory, the knowledge state reflects the ability distribution, and the feedback signal reflects the path execution quality. If multi-class features are directly concatenated, it is easy to cause high-dimensional redundancy and noise superposition [15]. In this paper, a state representation module is constructed by combining embedding mapping, gated temporal coding and attention aggregation, and the multimodal behavior information is compressed into a state vector that can be read by the policy network.

Let the behavior sequence matrix of user u at time t be $X^{L}_{u,t}$, resource interaction embedding be Eu,t, knowledge state vector be Ku,t, feedback feature be Fu,t, and multimodal input fusion vector be defined as follows.

$$z_{u,t}=\mathrm{Concat}\left(W_x X^{L}_{u,t}, W_e E_{u,t}, W_k K_{u,t}, W_f F_{u,t}\right) \tag{10}$$

where zu,t represent the fused multimodal input vector, and Wx, We, Wk, Wf represent the mapping matrix of behavior sequence, resource interaction, knowledge state, and feedback features, respectively. The formula compreses the input from different sources to the same latent space, and reduces the influence of dimension difference on subsequent temporal coding.

In order to extract the trend of state change in continuous behavior, this paper uses GRU structure to encode the fusion sequence. GRU controls the proportion of historical information retention through the update gate and the candidate state, which is suitable for handling interactive data with the coexistence of short-term fluctuations and periodic state accumulation of users. Its hidden state update process is represented as follows.

$$h_{u,t}=\left(1-r_{u,t}\right)\odot h_{u,t-1}+r_{u,t}\odot \tanh\left(W_h z_{u,t}+U_h h_{u,t-1}+b_h\right) \tag{11}$$

where, hu,t represents the current hidden state, hu,t−1 represents the last hidden state, ru,t represents the update gate, zu,t represents the multimodal input vector, Wh and Uh are

trainable parameter matrices, bh is the bias term, ⊙ represents element-wise multiplication. The structure can retain sequential patterns such as continuous completion, continuous error and long stay, so that the state vector has the ability of time memory.

Different time steps do not contribute equally to the path decision. Serial errors, low completion rates, and high-difficulty resource failures are often a better indicator of state changes than regular clicks. In this paper, attention weight is introduced to aggregate the hidden states and obtain the final state representation:

$$\bar{s}_{u,t}=\sum_{l=t-L+1}^{t}\alpha_{u,l}\,h_{u,l},\quad \alpha_{u,l}=\frac{\exp\left(v^{T}\tanh\left(W_a h_{u,l}+b_a\right)\right)}{\sum_{j=t-L+1}^{t}\exp\left(v^{T}\tanh\left(W_a h_{u,j}+b_a\right)\right)} \tag{12}$$

where, $\bar{s}_{u,t}$ represent the state representation after attention aggregation, $\alpha_{u,l}$ represent the attention weight at the l time step, $h_{u,l}$ and hu,j represent the hidden state at the corresponding time step, and v, Wa and ba are the attention layer parameters. This formulation enables the model to automatically increase the weight of key interaction segments and reduce the interference of ordinary low-information behaviors on state representation.

To show the computational link after multimodal data enters the state representation module, Figure 3 shows the structural relationship between feature mapping, GRU encoding, attention aggregation and state output.

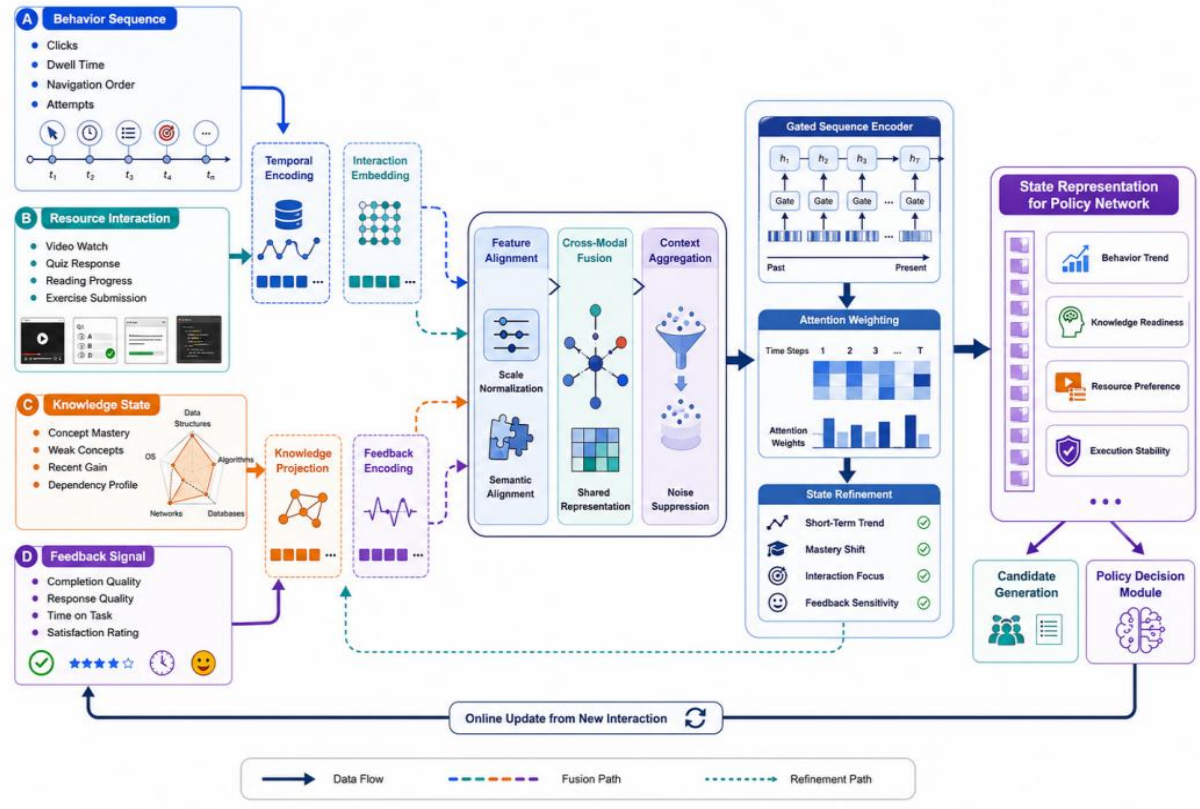


*Figure 2: Multimodal behavior feature fusion and state representation model diagram*

Through the above modeling, multimodal interaction data are transformed into state representations with temporal memory and critical segment perception capabilities. The state vector not only contains the user's recent behavior trajectory, but also integrates resource execution results, knowledge state changes and feedback signals, which can be used as a

unified input for subsequent resource candidate set generation and PPO policy network.

## 3.2 Resource candidate set generation based on knowledge state estimation

The generation of resource candidate sets determines the scale and quality of the action space of reinforcement learning. If all resource nodes are directly taken as the action set, the policy network needs to search in a large number of low correlation, too difficult or repeated resources, which is easy to cause a decrease in training efficiency and unstable action selection [16]. In this paper, a knowledge state estimation module is introduced before policy decision to filter candidate resources according to the user's current knowledge grasp probability, knowledge point dependency, resource difficulty and historical feedback results, so that the action space is compressed from the full resource set to the executable high relevant candidate set.

Let the state of user u at time t be characterized by $\bar{s}_{u,t}$. The mastery probability of the KTH knowledge unit can be expressed as follows.

$$p_{u,k}^{t}=\text{sigmoid}\left(w_k^T \bar{s}_{u,t}+b_k\right) \tag{13}$$

where $p_{u,k}^{t}$ represents the mastery probability of user u to knowledge unit k at time t, $\bar{s}_{u,t}$ represents the state representation obtained in the previous section, wk represents the mapping weight of the KTH knowledge unit, and bk represents the bias term. The lower the mastery probability is, the knowledge unit needs priority reinforcement. If the grasp probability is too high, the recommendation priority of the corresponding resource is reduced in the current stage.

Resource selection is not only based on mastery probability, but also needs to consider the predependency between knowledge units. Let $D \in R^{K\times K}$ denote the knowledge dependency matrix and Da,k=1 denote that the knowledge unit A is the antecedent node of the knowledge unit k. The learnability constraint on the target knowledge unit k is defined as follows.

$$\chi_{u,k}^{t}=\prod_{a=1}^{K}\left[1-D_{a,k}\left(1-p_{u,a}^{t}\right)\right] \tag{14}$$

where, $\chi_{u,k}^{t}$ represents the learnability weight of knowledge unit k to user u, K represents the total number of knowledge units, Da,k represents the antecedent dependency, and $p_{u,a}^{t}$ represents the mastery probability of antecedent knowledge unit a. If multiple prior knowledge points are not yet mastered, $\chi_{u,k}^{t}$ will decrease, thus inhibiting the skipping resource recommendation and ensuring the path coherence of the candidate set.

After obtaining the knowledge state and dependency constraints, we synthesize resource difficulty, label coverage and historical feedback to generate candidate resource scores:

$$G_{u,i}^{t}=\chi_{u,k(i)}^{t}\cdot\left(1-\left|p_{u,k(i)}^{t}-\rho_i\right|\right)+\beta\bar{g}_i-\delta o_{u,i}^{t} \tag{15}$$

where, $G_{u,i}^{t}$ represents the candidate score of user u for resource i at time t, k(i) represents the knowledge unit corresponding to resource i, ρi represents the difficulty of the resource, $\bar{g}_i$ represents the historical average feedback of the resource, $o_{u,i}^{t}$ represents the repeated exposure times of the user to the resource, β and δ are the weight coefficients. This score considers knowledge learnability, difficulty adaptation, resource quality and repetition penalty

simultaneously, which can reduce low-quality resources and repetitive resources entering the action space.

To clarify the role of various types of constraints in the candidate set screening process, Table 3 summarizes the resource candidate set generation rules.

*Table 3: Resource Candidate Set Generation Constraints and Screening Rules*

| Constraint Type | Calculation Basis | Screening Role |
|---|---|---|
| Mastery Constraint | Mastery Probability of Knowledge Unit | Exclude Mastered or Low-relevance Resources |
| Dependency Constraint | Knowledge Dependency Matrix | Suppress Cross-level Jump Recommendation |
| Difficulty Adaptation Constraint | Difference Between Mastery Probability and Resource Difficulty | Control Resource Difficulty Span |
| Feedback Quality Constraint | Historical Rating, Completion Rate | Improve Candidate Resource Quality |
| Repeated Exposure Constraint | Historical Recommendation Times | Reduce Proportion of Repeated Resources |

The generation process of the resource candidate set needs to connect knowledge state estimation, dependency constraints, and resource screening as continuous computational links. Figure 4 shows the process of compressing candidate resources from the full resource library to the action space.

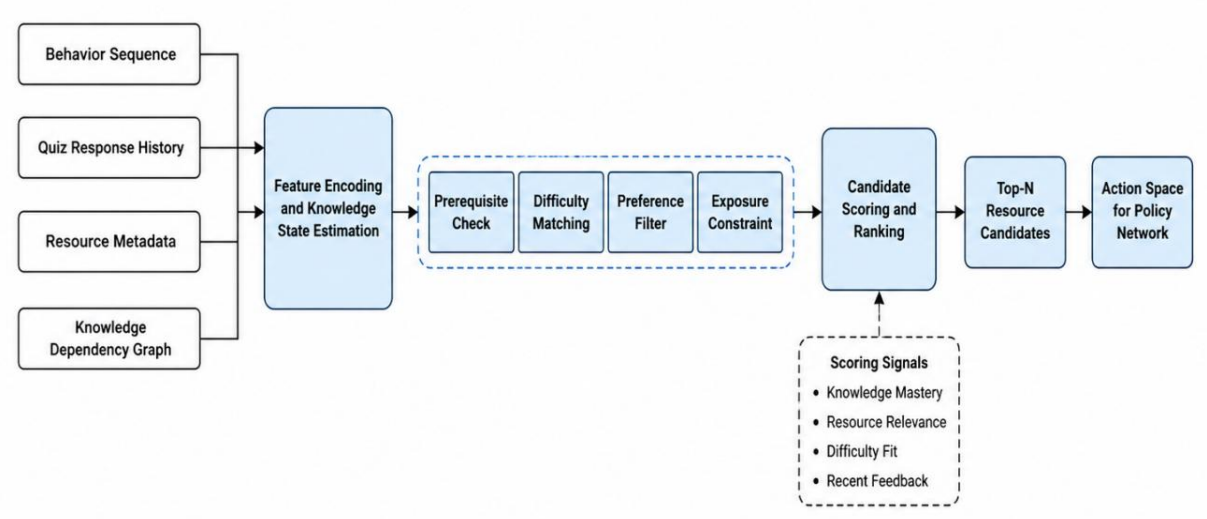


*Figure 4: Flowchart of resource candidate set generation based on knowledge state estimation*

By the above method, the full resource space is compressed into a set of candidate actions that match the current state of the user. This set not only retains the adaptability of resource difficulty, but also satisfies the precondition relationship of knowledge points and feedback quality constraints, which can reduce the complexity of policy network search and improve the stability of subsequent path action selection [17].

### 3.3 Multi-objective reward function design for long-term payoff optimization

The optimization objective of the path generation model cannot rely only on the single-step CTR or the resource matching score. If the reward function only emphasizes immediate completion results, the policy network may tend to recommend low-difficulty resources, which are difficult to promote state improvement although short-term feedback is good. If we only focus on increasing difficulty, we may cause continuous failures, higher bounce rates, and interrupted path execution [18]. In this paper, the long-term benefits are divided into five types of signals: state promotion, resource matching, path coherence, feedback quality and load penalty, so that the PPO policy network takes into account both revenue growth and execution stability in the update process.

After the user performs the action 'at', the change in knowledge state constitutes the core source of path benefits. Let the knowledge state vector of user u at time t be Ku,t, and it is updated to Ku,t+1 in the next moment. The knowledge coverage vector corresponding to the action is mat. Then, the state improvement reward is defined as:

$$R_{u,t}^{gain}=\frac{\left(K_{u,t+1}-K_{u,t}\right)^{T}m_{a_t}}{\left\|m_{a_t}\right\|_1+\varepsilon} \tag{16}$$

where, $R_{u,t}^{gain}$ represents the state promotion reward, Ku,t+1−Ku,t represents the knowledge state increment, mat represents the knowledge units covered by action at, $\left\|m_{a_t}\right\|_1$ represents the number of covered knowledge units, and ε is the smoothing term. This reward can transform the effective state change generated by the path action into a trainable signal, avoiding the model to make judgments only based on the completion state of resources.

In addition to state lifting, path actions need to satisfy difficulty adaptation and coherence. Let $p_{u,k(a_t)}^{t}$ denote the probability that the user grasps the knowledge unit to which the action at belongs, ρat denote the difficulty of the resource, Dat−1,at denote the knowledge dependency connectivity between adjacent actions, and the matching coherence reward is defined as follows.

$$R_{u,t}^{match}=\lambda_m\left(1-\left|p_{u,k(a_t)}^{t}-\rho_{a_t}\right|\right)+\lambda_d D_{a_{t-1},a_t} \tag{17}$$

where, $R_{u,t}^{match}$ represent resource matching and path coherence rewards, λm and λd are weight coefficients, $p_{u,k(a_t)}^{t}$ represents the user's current mastery probability, ρat represents the difficulty of action resources, and Dat−1,at represents the dependency strength between the previous action and the current action. This formula makes the model give priority to the resources whose difficulty is close to the current state and closely connected with the previous node, reducing too easy, too difficult and skip recommendation.

The integrated reward function further introduces feedback quality and load penalty. Let cu,t denote the completion result, gu,t denote the feedback score, lu,t denote the learning load, ou,t denote the repeated exposure intensity, and the comprehensive reward is defined as follows.

$$R_{u,t}=\alpha_1 R_{u,t}^{gain}+\alpha_2 R_{u,t}^{match}+\alpha_3 c_{u,t}+\alpha_4 g_{u,t}-\alpha_5 l_{u,t}-\alpha_6 o_{u,t} \tag{18}$$

where, Ru and tmatch represent resource matching and path coherence rewards, λm and λd

are weight coefficients, pu,k(at)t represents the user's current mastery probability, ρat represents the difficulty of action resources, and Dat−1,at represents the dependency strength between the previous action and the current action. This formula makes the model give priority to the resources whose difficulty is close to the current state and closely connected with the previous node, reducing too easy, too difficult and skip recommendation.

The integrated reward function further introduces feedback quality and load penalty. Let cu,t denote the completion result, gu,t denote the feedback score, lu,t denote the learning load, ou,t denote the repeated exposure intensity, and the comprehensive reward is defined as follows.

*Table 4: Multi-objective Reward Function Composition and Optimization Direction*

| Reward Item | Calculation Basis | Optimization Direction |
|---|---|---|
| State Improvement Reward | Knowledge State Increment, Resource Coverage Vector | Improve Long-term Return |
| Difficulty Matching Reward | Difference Between Mastery Probability and Resource Difficulty | Control Resource Adaptation Degree |
| Path Coherence Reward | Dependency Connectivity of Adjacent Resources | Maintain Sequence Structure Stability |
| Completion Feedback Reward | Completion Result, Feedback Rating | Improve Path Execution Quality |
| Load Penalty | Dwell Time, Continuous Errors, Repeated Exposure | Reduce Path Interruption Risk |

Once the reward function is designed, each path action can be translated into explicit training feedback. The state promotion reward provides the long-term optimization direction, matches the coherent reward constraint path structure, the completion feedback reward reflects the action execution effect, and the load penalty controls the risk of the strategy. Compared with single click or completion rate reward, the multi-objective function can more stably guide the PPO policy network to strike a balance between resource adaptation, path continuity and long-term state improvement, and provide interpretable optimization signals for subsequent policy updates.

## 3.4 Path dynamic generation algorithm based on PPO policy network

The PPO policy network is used to complete the probabilistic modeling and stable update of path actions. The set of candidate resources in the path generation scenario will constantly change with the user's state, knowledge mastery degree and feedback results, and the common policy gradient method is easy to cause a large deviation of action probability due to the single round reward fluctuation [19]. PPO separates action selection from value estimation through the actor-critic structure. The Actor network is responsible for output the selection probability of candidate resources, and the Critic network is responsible for estimating the current state value. The pruning objective function is used to limit the differences between the old and new strategies, so that the path generation can maintain stable iteration under the condition of dynamic feedback [20].

Let the state of user u at time t be characterized by $\bar{s}_{u,t}$ and the set of candidate resources be Au,t. The Actor network inputs the scoring function together with the state representation and the candidate resource embedding, and then obtains the selection probability of each path action through Softmax:

$$\pi_\theta\left(a_t \mid \bar{s}_{u,t}\right)=\frac{\exp\left(q_\theta\left(\bar{s}_{u,t},e_{a_t}\right)\right)}{\sum_{a_j\in A_{u,t}}\exp\left(q_\theta\left(\bar{s}_{u,t},e_{a_j}\right)\right)} \tag{19}$$

where $\pi_\theta\left(a_t \mid \bar{s}_{u,t}\right)$ represents the probability that the policy network with parameter θ selects action at under state $\bar{s}_{u,t}$, qθ(·) represents the Actor network action scoring function, eat represents the resource embedding vector corresponding to action at, and Au,t represents the current set of candidate resources. The probability distribution enables the model to complete adaptive action selection within the candidate resources and avoid inefficient search in the full resource space.

The Critic network provides value benchmarks for policy updates. The single-step reward is easily affected by the completion state, feedback score and short-term dwell time, and directly used to update the Actor will amplify the reward noise. The advantage function uses a multi-step reward structure, which combines the future returns of several steps with the state value estimation to measure the benefit increment of the current action relative to the value benchmark:

$$A_t=\sum_{l=0}^{H-1}\gamma^l R_{u,t+l}+\gamma^H V_\psi\left(\bar{s}_{u,t+H}\right)-V_\psi\left(\bar{s}_{u,t}\right) \tag{20}$$

where, At represents the estimated advantage value of the action at step t, H represents the multi-step reward length, γ represents the discount factor, Ru,t+l represents the immediate reward at step t+l, Vψ(·) represents the Critic value network with parameter ψ, $\bar{s}_{u,t+H}$ represents the state representation after multi-step execution. This design can weaken the interference of single-step feedback anomaly on policy update, and make action selection pay more attention to continuous path benefits.

In the policy update phase, the action probability ratio between the old and new policies is calculated, and then the pruning function is used to constrain the optimization objective. If the new policy changes too much compared with the old policy, the pruning term will inhibit the objective function to continue to increase, so as to avoid the action probability from overshifting on a small number of high reward samples. The PPO clipping objective function is expressed as follows.

$$L^{PPO}(\theta)=E_t\left[\min(r_t(\theta)A_t,\text{clip}(r_t(\theta),1-\epsilon,1+\epsilon)A_t)\right],\quad r_t(\theta)=\frac{\pi_\theta\left(a_t \mid \bar{s}_{u,t}\right)}{\pi_{\theta_{old}}\left(a_t \mid \bar{s}_{u,t}\right)} \tag{21}$$

where, LPPO(θ) represents the PPO pruning objective function, rt(θ) represents the probability ratio of new and old policy actions, πθold represents the policy network before updating, and ϵ\ epsipoϵ represents the pruning threshold. The clipping interval limits the policy update within a controllable range, which can improve the training stability of the path generation model under the condition of dynamic candidate set and delayed feedback.

When the algorithm is executed, the state representation module first outputs su,t, and the candidate set generation module gives Au,t. The Actor network samples path actions according to Equation (19), and the environment returns completion status, knowledge state increment, feedback score and load index. After the reward function calculates the immediate reward, the Critic network estimates the state value and generates a dominance estimate from Equation (20). Subsequently, the Actor parameters are updated using equation (21), and the

Critic parameters are updated through back-propagation of value errors. After each round of training, the old policy parameters are synchronized to the current policy parameters and enter the next batch of path samples. In the early stage of training, high sampling randomness is retained to enhance exploration, and the proportion of high probability action selection is gradually increased in the later stage, so that the model shifts from candidate resource exploration to stable path output.

In order to show the running relationship of PPO policy network in path dynamic generation, Figure 5 shows the closed-loop structure among state input, action selection, reward feedback and policy update.

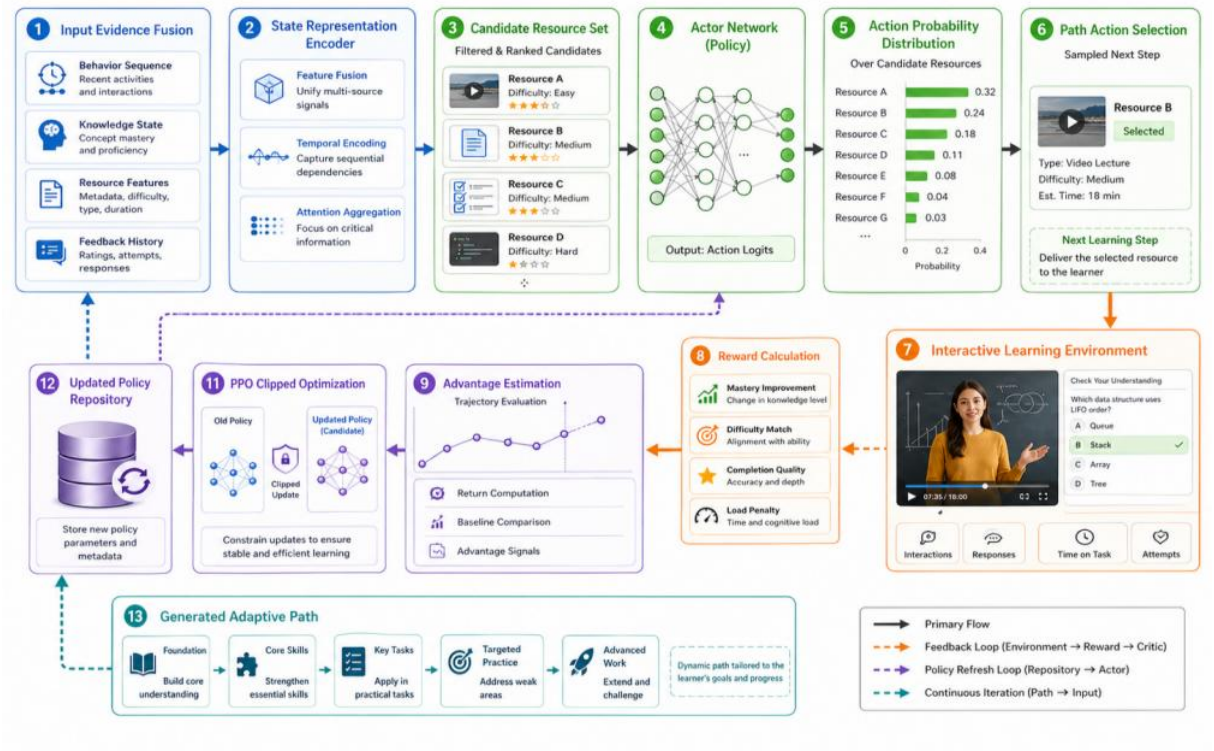


*Figure 5: Structure diagram of path dynamic generation model based on PPO policy network*

The PPO policy network organizes candidate resource selection, value evaluation and policy tailoring into a unified algorithm link. The Actor network provides path action probability, the Critic network stable value benchmark, multi-step advantage estimation reduces the impact of single feedback fluctuation, and the objective function is trimmed to suppress excessive policy update. This mechanism can balance exploration, profit and stability in the process of dynamic path generation, and provide the core algorithm foundation for subsequent feedback-driven online iteration

## 3.5 Feedback-driven policy update and online iteration mechanism

The path generation model needs to continuously revise the strategy according to the execution feedback, otherwise it is easy to stay in the fixed preference formed in the offline training phase. The feed-back driven mechanism writes the resource completion status, test change, stay time, repeat access and user rating into the next round of state, so that the policy network can update the action probability according to the real interaction results. Instead of simply append logs, the feedback signal is transformed into state modification and policy update gradient, and the closed loop of "path generation - execution feedback - state

modification - policy update - new path output" is realized.

Let the feedback vector Fu,t be obtained after the execution of action at, which contains the completion result, score change, stay duration, jump out state and score information. The state vector after feedback correction is defined as follows.

$$\tilde{s}_{u,t+1}=\sigma\left(W_o\bar{s}_{u,t+1}+W_fF_{u,t}+W_rR_{u,t}+b_o\right) \tag{22}$$

where, $\tilde{s}_{u,t+1}$represents the next state after feedback correction, $\bar{s}_{u,t+1}$ represents the original state representation, Fu,t represents the feedback vector, Ru,t represents the immediate reward, Wo, Wf, Wr are mapping matrices, bo is the bias term, and $\sigma(\cdot)$ is the activation function. The formula writes the user's real execution results into the state update process, so that the subsequent action selection can perceive the path execution quality.

In the online iteration stage, the small batch incremental update method was used to avoid the drastic fluctuation of the policy caused by a single feedback. Let the PPO target formed by the NTH batch of feedback samples be $L_n^{PPO}(\theta)$, and the policy parameters are updated as follows.

$$\theta_{n+1}=\theta_n+\eta_p\nabla_\theta L_n^{PPO}(\theta)-\eta_c\nabla_\theta\Omega(\theta_n) \tag{23}$$

where, θn+1 represents the updated policy parameters, θn represents the current policy parameters, ηp represents the policy learning rate, ηc represents the regular constraint weight, and Ω(θn) represents the parameter stability term. The first term promotes the policy update in the direction of high-payoff actions, and the second term inhibits the parameter drift too fast, ensuring that the online iteration process achieves a balance between the new feedback adaptation and the maintenance of the old policy.

To show the execution link of the feedback-driven mechanism, Figure 6 shows the online iteration process of the path strategy.

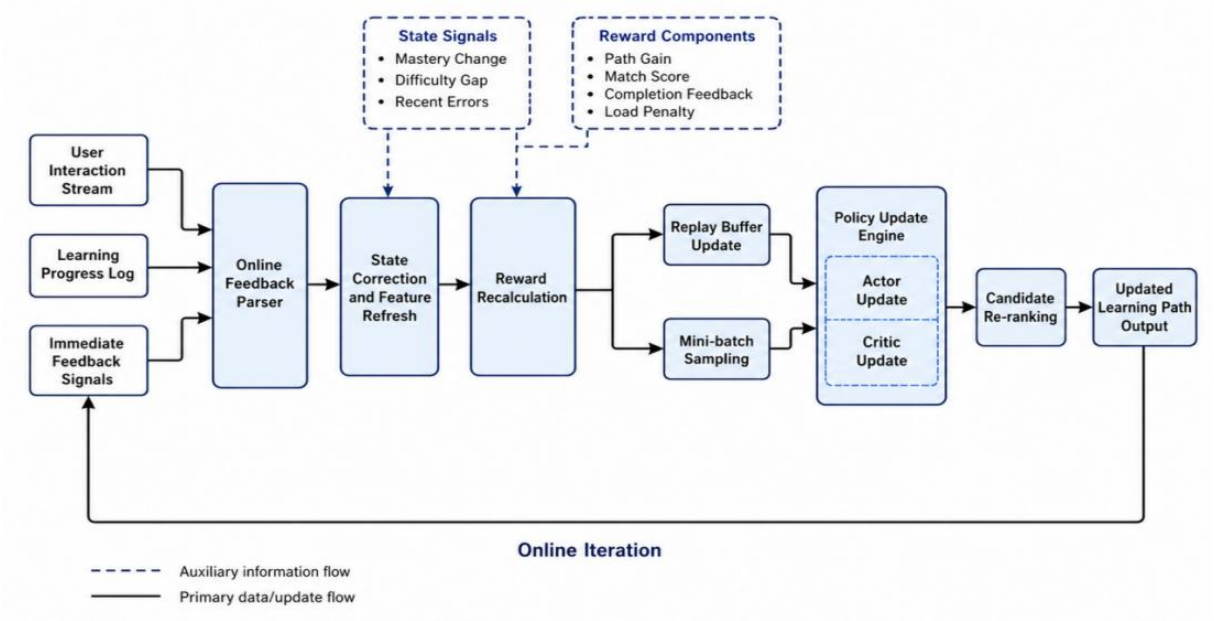


*Figure 6: Diagram of the feedback-driven online iteration mechanism of the path strategy*

Through the above mechanism, the model can convert the changes in the user's execution process into the next round of state and policy parameters in time. The resources with higher

completion rate, obvious score improvement and better feedback will improve the subsequent selection probability. Resources with high bounce rates, consecutive errors, or excessive repeated exposures are weighed down by both the reward function and the state correction. The online iteration changes the path generation from one-time offline recommendation to a continuous optimization process, which helps to improve the adaptability and stability in dynamic environments.

# 4 Model training, experimental design and performance verification

## 4.1 Experimental environment, data set construction and evaluation index

The experiment was completed in a unified software and hardware environment. The dataset consisted of online interaction logs, resource attributes, knowledge unit labels and feedback records, which contained 816 users, 12864 interaction records, 420 resource nodes and 96 knowledge units. Each record contains the user number, resource number, number of clicks, dwell time, completion status, quiz score, feedback score, and timestamp. The data is divided according to the user time series, and the proportions of training set, validation set and test set are 70%, 15%, 15% to avoid leakage of subsequent interaction information into the training phase. The evaluation indicators are selected as Accuracy, NDCG@10, Completion Rate, Mastery Gain and Average Reward. The recommendation accuracy is used to calculate the hit of the predicted action of the model with the positive feedback resource:

$$\mathrm{Acc}=\frac{1}{N}\sum_{n=1}^{N} I\left(a_n^{pred}=a_n^{pos}\right) \tag{24}$$

where N represents the number of test samples, $a_n^{pred}$ represents the predicted resource action, $a_n^{pos}$ represents the positive feedback resource action, and $I(\cdot)$ represents the indicator function. The ranking quality is evaluated by NDCG@10:

$$\mathrm{NDCG@10}=\frac{1}{N}\sum_{n=1}^{N}\frac{\mathrm{DCG}_n@10}{\mathrm{IDCG}_n@10} \tag{25}$$

where, DCGn@10 represents the cumulative gain of the top 10 recommendation results, and IDCGn@10 represents the cumulative gain under ideal ranking. To ensure the consistency between model training and result reproduction, Table 5 gives the experimental platform, data scale and training parameter configuration.

*Table 5: Experimental Environment, Dataset and Training Parameter Configuration*

| Category | Item | Configuration Value |
|---|---|---|
| Hardware | CPU | Intel Xeon Silver 4314 × 2 |
| | GPU | NVIDIA RTX 4090 24GB |
| | Memory | 128 GB |
| Software | Operating System | Ubuntu 22.04 |
| | Framework | PyTorch 2.2 |
| Dataset | Users | 816 |
| | Interaction Records | 12864 |
| | Resource Nodes | 420 |
| | Knowledge Units | 96 |
| Training | Sequence Window | 12 |
| | State Dimension | 128 |
| | Resource Dimension | 64 |
| | Batch Size | 64 |
| | Learning Rate | 0.001 |
| | Max Epoch | 150 |

## 4.2 Comparison model and ablation experimental design

The experiment set up nine groups of models, including three groups of baseline, five groups of ablation models and one group of complete models. Rule-based does not participate in parameter training, and only outputs paths in a fixed difficulty order. CF constructs a user similarity matrix based on the training set. DQN uses action-value function to complete resource selection. Ppo-type models are uniformly trained for 150 rounds, and the average training batch for a single round is 141. The number of DQN parameters is about 0.86M, the number of Proposed PPO parameters is about 1.24M, and a single round of training takes about 18.6 s. Ablation experiments remove the knowledge state estimation, feedback reward, pruning constraint, knowledge dependence matrix and attention aggregation layer, respectively, which are used to locate the influence of key modules on the performance of path generation. To clarify the structural differences of each experimental group and the purpose of verification, Table 6 gives the specific Settings.

*Table 6: Comparative Models and Ablation Experiment Settings*

| Model Name | Model Setting | Training Configuration/Validation Purpose |
|---|---|---|
| Rule-based | Fixed Rule Path Generation | No Training Parameters, Validate Static Rule Baseline |
| CF | Recommend Resources Based on Similar User Behavior | Construct User Similarity Matrix, Validate Traditional Recommendation Performance |
| DQN | Select Resources by Action Value Function | 0.86M Parameters, Validate Reinforcement Learning Baseline |
| PPO-w/o-KS | Remove Knowledge State Estimation Module | Observe Influence of Missing State Estimation |
| PPO-w/o-FB | Remove Feedback Reward Item | Observe Influence of Missing Feedback Signal |
| PPO-w/o-Clip | Remove Clipping Constraint | Observe Policy Update Stability |
| PPO-w/o-Dep | Remove Knowledge Dependency Matrix | Observe Role of Path Coherence Constraint |
| PPO-w/o-Attn | Remove Attention Aggregation Layer | Observe Role of Key Segment Identification |
| Proposed PPO | Retain All Modules | 1.24M Parameters, Validate Comprehensive Performance |

In the testing phase, the model parameters are fixed, no longer updated online, and the output Accuracy, NDCG@10, Completion Rate, Mastery Gain and Average Reward are unified.

## 4.3 Analysis of path generation accuracy and recommendation performance

The path generation performance needs to be examined simultaneously from three levels: ranking accuracy, path completion quality and long-term gain. In this section, the complete table of experimental results is reserved to present the original indicators of each model. Instead of repeating the original values, the three effect comparison charts are transformed into relative improvement rate and performance degradation, which are used to highlight the advantages of the complete model and the contributions of key modules. The improvement ratio of the model over the baseline is defined as follows:

$$\Delta P=\frac{P_m-P_{DQN}}{P_{DQN}}\times 100\% \tag{26}$$

where, ΔP represents the index improvement rate of model m relative to the DQN baseline, Pm represents the result of the current model on a certain index, and PDQN represents the result of DQN on the same index. In order to give the complete original results of each model, the experimental indicators are sorted out as follows, and Table 7 lists the statistical results of the comparison experiment and ablation experiment.

*Table 7: Results of Comparative Experiment and Ablation Experiment*

| Model | Accuracy/% | NDCG@10 |
|---|---|---|
| Rule-based | 78.4 | 0.713 |
| CF | 82.1 | 0.746 |
| DQN | 87.6 | 0.812 |
| PPO-w/o-KS | 89.0 | 0.831 |
| PPO-w/o-FB | 89.2 | 0.834 |
| PPO-w/o-Clip | 90.1 | 0.846 |
| PPO-w/o-Dep | 90.4 | 0.852 |
| PPO-w/o-Attn | 91.2 | 0.861 |
| Proposed PPO | 93.5 | 0.881 |

From the experimental results, Proposed PPO achieves the highest value in the five indicators. Compared with DQN, the Accuracy is improved from 87.6% to 93.5%, and the relative increase is 6.74%. NDCG@10 increased from 0.812 to 0.881, with a relative increase of 8.50%. Rule-based and CF have significantly lower action hit rate and ranking quality, which indicates that static rules and traditional similarity recommendation are difficult to deal with state changes in continuous path decision. In order to highlight the gains of the improved reinforcement learning model in action hitting and ranking quality, Figure 7 shows the accuracy and ranking performance improvement results of the relative DQN.

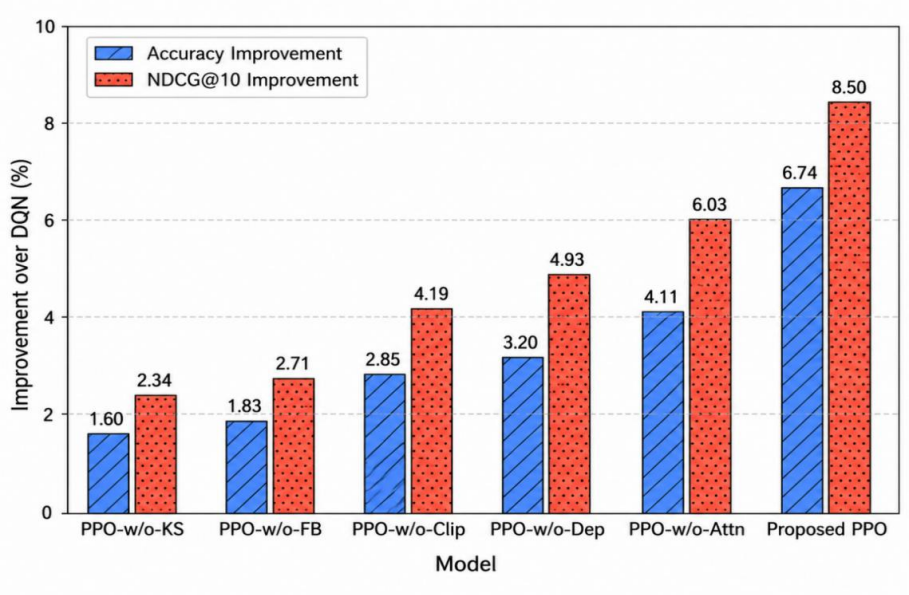


*Figure 7: Comparison chart of path generation accuracy and sorting performance improvement rate relative to DQN*

The improvement rates of Proposed PPO in Accuracy and NDCG@10 are 6.74% and 8.50% respectively, both higher than those of each ablation model. The improvement amplitude of PPO-w/o-Attn and PPO-w/o-Dep is second only to this. This indicates that attention aggregation and knowledge dependency constraints can improve state representation and resource sorting. The improvement of PPO-w/o-KS and PPO-w/o-FB is relatively low, suggesting that state estimation and absence of feedback rewards will weaken the quality of action selection.

The path generation also needs to pay attention to the completion rate and state gain. Figure 8 presents the completion rate and knowledge state improvement results relative to DQN.

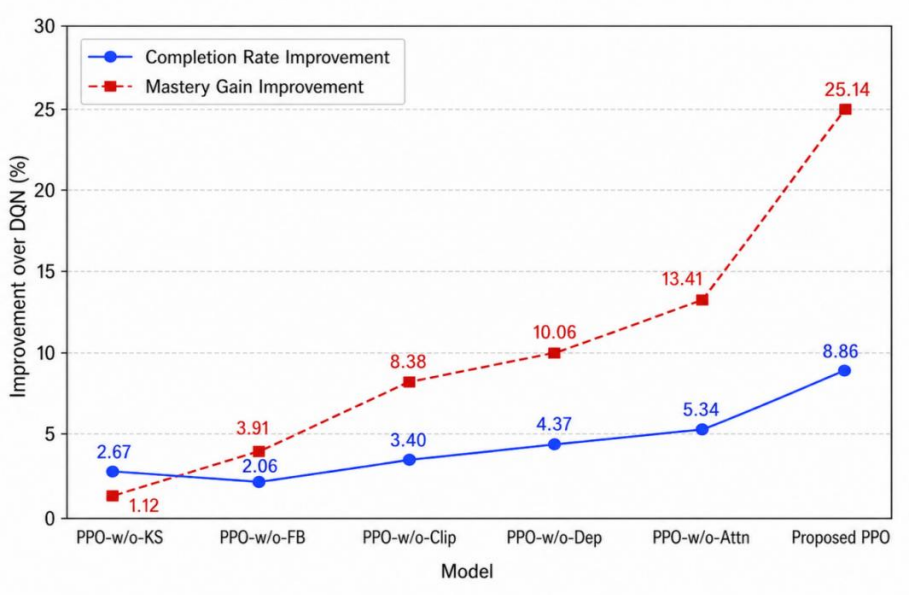


*Figure 8: Comparison of path completion rate and knowledge state improvement rate of relative DQN*

Proposed PPO has the most obvious improvement in Completion Rate and Mastery Gain, which are 8.86% and 25.14% respectively. Mastery Gain of PPO-w/o-KS had the lowest improvement rate, indicating that knowledge state estimation directly affected long-term state change. The improvement of Completion Rate of PPO-w/o-FB is low, which indicates that the feedback reward has a constraint effect on the stability of path execution.

The ablation experiment is used to judge the key module contribution, and Figure 9 shows the performance degradation of each ablation model relative to the complete model.

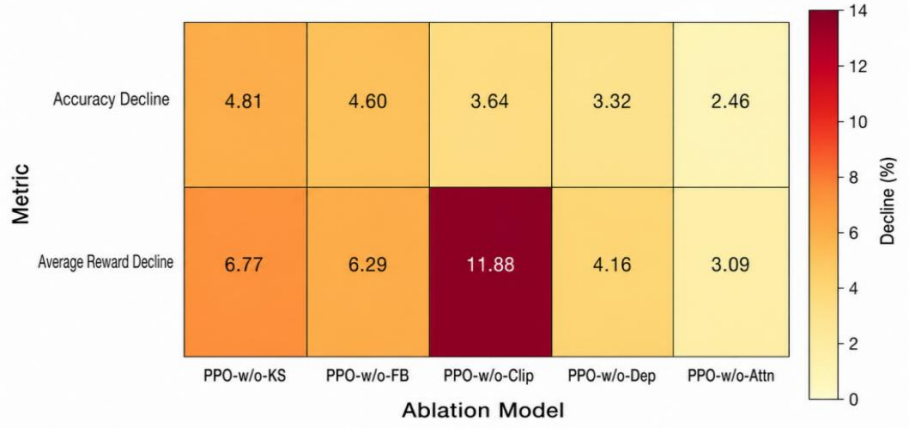


*Figure 9: Comparison of performance degradation of the ablation model with respect to the complete model*

PPO-w/o-Clip shows the largest decrease in Average Reward, which indicates that the clipping constraint has a significant impact on the long-term payoff stability. The Accuracy decrease of PPO-w/o-KS and PPO-w/o-FB is more prominent, which indicates that knowledge state estimation and feedback reward directly affect the quality of resource action selection. On the whole, the advantage of the complete model comes from the multi-module collaborative optimization.

## 4.4 Analysis of policy convergence and dynamic update effect

Policy convergence is used to test whether the model can form a stable path output in multiple rounds of training. DQN's reward increased rapidly in the first 60 rounds, but there were still fluctuations after 80 rounds. PPO-w/o-Clip declined in the 60th and 80th rounds, indicating that the policy update is easily affected by local high reward samples when there is no pruning constraint. The Proposed PPO improved rapidly in the first 40 rounds, and entered the stable interval after the 82nd round, and the final average reward reached 0.842. In order to show the reward variation trend of different reinforcement learning models in the training process, Figure 10 shows the average reward convergence curve.

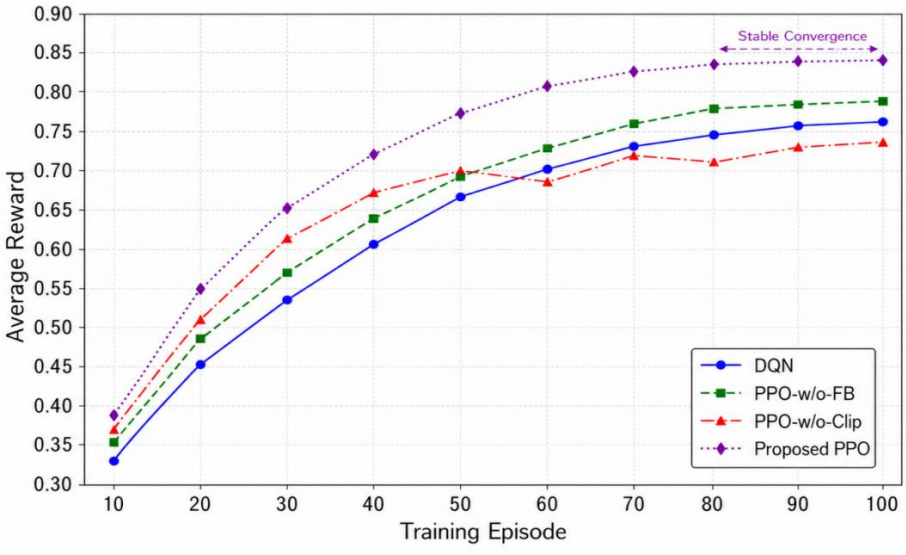


*Figure 10: Average Reward Convergence Curves of Different Reinforcement Learning Models*

The dynamic update effect is used to verify whether the model can continuously correct the path strategy based on new feedback. During the online iterative process, the path adjustment rate decreased from 31.6% in the initial stage to 9.7%, the repetition exposure rate decreased from 14.2% to 8.5%, and the average feedback correction delay remained within 42.8 ms. After the update, both the Completion Rate and Average Reward continued to improve, indicating that the feedback-driven mechanism can reduce the repeated push of invalid resources and enhance the continuous adaptability of path generation. To further compare the changes in key indicators before and after online update, Figure 11 presents the comparison results of the dynamic update effect.

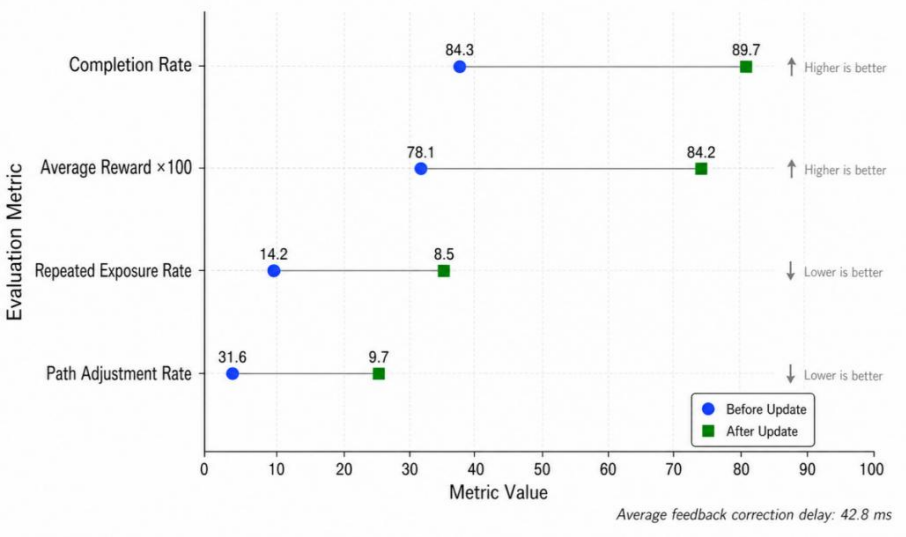


*Figure 11: omparison diagram of dynamic update effect in the online iteration stage*

Synthesizing the two groups of results, it can be seen that the Proposed PPO enters the stable interval after the 82nd round, and the final average reward reaches 0.842, which is higher than 0.761 of DQN and 0.789 of PPO-W /o-FB, indicating that cutting update and feedback rewards can improve the upper limit of strategy convergence. In the online update phase, the path adjustment rate is reduced from 31.6% to 9.7%, the repeat exposure rate is

reduced from 14.2% to 8.5%, and the average feedback correction delay is controlled within 42.8 ms, which indicates that the feedback-driven mechanism can reduce the repeated push of invalid resources and maintain a good real-time update ability.

# 5 Conclusion

Focusing on the problem of personalized adaptive path dynamic generation, this paper constructs a sequential decision model based on PPO reinforcement learning. The model encodes multi-source behavior sequences, resource attributes, knowledge states and feedback records into a state vector, compresses the candidate action space by knowledge dependence constraints, and uses a multi-objective reward function to coordinate path benefits, difficulty adaptation, feedback quality and load penalty. In the policy update phase, the Actor-Critic structure and PPO pruning objective function were introduced to reduce the interference of single-step reward fluctuation on action probability and improve the training stability under the condition of dynamic candidate set. Experimental results show that the proposed model is superior to rule-based recommendation, collaborative filtering and DQN baseline models in terms of path generation accuracy, recommendation ranking quality, completion stability and policy convergence effect. The results show that knowledge state estimation, feedback reward and tailoring update can effectively improve the long-term revenue optimization ability of adaptive path generation. In the future, Transformer sequence encoding, offline reinforcement learning constraints, and cross-scenario transfer training mechanisms can be further introduced to enhance the generalization ability of the model under large-scale resource libraries and cold-start user conditions.

# Funding

This work was supported by the Special Project of "Language Serve Vocational Education Going Global" for Vocational Colleges in 2024 by the National Advisory Committee on Teaching Foreign Languages to Majors in Higher Vocational Education under the Ministry of Education of China, titled"Research and Practice on the Construction of Blended English 'Golden Courses' in Higher Vocational Colleges under the Artificial Intelligence Environment" (Project Number:WYJZW-2025-023)

# About The Author

Min Xu was born in Luzhou, Sichuan, P.R. China, in 1983. She obtained a master's degree from Southwest University of Science and Technology in China. I am currently working at General Education Department, Sichuan University of Architectural Technology. My main research direction is language teaching and learning. 15883422864@163.com

Liang Yin was born in Deyang, Sichuan,P.R. China, in 1983. He obtained a bachelor's degree from Southwest University of Science and Technology in China. I am currently working at Library & Campus Information Technology Center, Sichuan Institute of Industrial Technology. My main research direction is information security technology. swustit1128@163.com

# References

[1] AFSAR M M, CRUMP T, FAR B. Reinforcement learning based recommender systems: A survey[J]. ACM Computing Surveys, 2022, 55(7): 145:1-145:38.

[2] RAHAYU N W, FERDIANA R, KUSUMAWARDANI S S. A systematic review of learning path recommender systems[J]. Education and Information Technologies, 2023, 28(6): 7437-7460.

[3] SONG X, LI J, CAI T, et al. A survey on deep learning based knowledge tracing[J]. Knowledge-Based Systems, 2022, 258: 110036.

[4] ABDELRAHMAN G, WANG Q, NUNES B P. Knowledge tracing: A survey[J]. ACM Computing Surveys, 2023, 55(11): 224:1-224:37.

[5] ZHENG Y, WANG D, ZHANG J, et al. A unified framework for personalized learning pathway recommendation in e-learning contexts[J]. Education and Information Technologies, 2025, 30(6): 7911-7948.

[6] ZHANG F, FENG X, WANG Y. Personalized process-type learning path recommendation based on process mining and deep knowledge tracing[J]. Knowledge-Based Systems, 2024, 303: 112431.

[7] MA D, ZHU H, LIAO S, et al. Learning path recommendation with multi-behavior user modeling and cascading deep Q networks[J]. Knowledge-Based Systems, 2024, 294: 111743.

[8] YUN Y, DAI H, AN R, et al. Doubly constrained offline reinforcement learning for learning path recommendation[J]. Knowledge-Based Systems, 2024, 284: 111242.

[9] HALDAR S, SENGUPTA S, DAS A K. Personalized learning path recommendation using graph reinforcement learning[J]. Procedia Computer Science, 2025, 258: 3480-3489.

[10] FU Z. Integrating reinforcement learning with dynamic knowledge tracing for personalized learning path optimization[J]. Scientific Reports, 2025, 15(1): 40202.

[11] LUO G, GU H, DONG X, et al. HA-LPR: A highly adaptive learning path recommendation[J]. Education and Information Technologies, 2025, 30(10): 14597-14627.

[12] LI J, YU S, ZHANG T. Learning path recommendation based on reinforcement learning[J]. Engineering Letters, 2024, 32(9): 1823-1832.

[13] ZHANG S, PU J, CUI J, et al. MLC-DKT: A multi-layer context-aware deep knowledge tracing model[J]. Knowledge-Based Systems, 2024, 303: 112384.

[14] YANG H, HU S, GENG J, et al. Heterogeneous graph-based knowledge tracing with spatiotemporal evolution[J]. Expert Systems with Applications, 2024, 238: 122249.

[15] DUAN Z, DONG X, GU H, et al. Towards more accurate and interpretable model:

Fusing multiple knowledge relations into deep knowledge tracing[J]. Expert Systems with Applications, 2024, 243: 122573.

[16] DAI H, ZHANG Y, YUN Y, et al. Adaptive meta-knowledge dictionary learning for incremental knowledge tracing[J]. Engineering Applications of Artificial Intelligence, 2024, 132: 107969.

[17] HAN D, KIM D, KIM M, et al. Temporal enhanced inductive graph knowledge tracing[J]. Applied Intelligence, 2023, 53(23): 29282-29299.

[18] XI X, ZHAO Y, LIU Q, et al. Integrating offline reinforcement learning with transformers for sequential recommendation[C]//Proceedings of the 17th ACM Conference on Recommender Systems. New York: ACM, 2023: 6 pages.

[19] GAO C, HUANG K, CHEN J, et al. Alleviating Matthew effect of offline reinforcement learning in interactive recommendation[C]//Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval. New York: ACM, 2023: 238-248.

[20] [ZHANG Y, QIU R, LIU J, et al. ROLeR: Effective reward shaping in offline reinforcement learning for recommender systems[C]//Proceedings of the 33rd ACM International Conference on Information and Knowledge Management. New York: ACM, 2024: 3269-3278.