Item-Difficulty-Aware Learning Path Recommendation: From a Real Walking Perspective

Haotian Zhang
State Key Laboratory of Cognitive Intelligence, University of Science and Technology of China
Hefei, China
sosweetzhang@mail.ustc.edu.cn

Shuanghong Shen
State Key Laboratory of Cognitive Intelligence, University of Science and Technology of China & Institute of Artificial Intelligence, Hefei Comprehensive National Science Center
Hefei, China
closer@mail.ustc.edu.cn

Bihan Xu
State Key Laboratory of Cognitive Intelligence, University of Science and Technology of China
Hefei, China
xbh0720@mail.ustc.edu.cn

Zhenya Huang*
State Key Laboratory of Cognitive Intelligence, University of Science and Technology of China & Institute of Artificial Intelligence, Hefei Comprehensive National Science Center
Hefei, China
huangzhy@ustc.edu.cn

Jinze Wu
iFLYTEK AI Research
Hefei, China
hxwjz@mail.ustc.edu.cn

Jing Sha
iFLYTEK AI Research
Hefei, China
jingsha@iflytek.com

Shijin Wang
State Key Laboratory of Cognitive Intelligence & iFLYTEK AI Research
Hefei, China
sjwang3@iflytek.com

ABSTRACT

Learning path recommendation aims to provide learners with a reasonable order of items to achieve their learning goals. Intuitively, the learning process on the learning path can be metaphorically likened to walking. Despite extensive efforts in this area, most previous methods mainly focus on the relationship among items but overlook the difficulty of items, which may raise two issues from a real walking perspective: (1) The path may be rough: When learners tread the path without considering item difficulty, it's akin to walking a dark, uneven road, making learning harder and dampening interest. (2) The path may be inefficient: Allowing learners only a few attempts on very challenging items before switching, or persisting with a difficult item despite numerous attempts without mastery, can result in inefficiencies in the learning journey. To conquer the above limitations, we propose a novel method named Difficulty-constrained Learning Path Recommendation (DLPR), which is aware of item difficulty. Specifically, we first explicitly categorize items into learning items and practice items, then construct a hierarchical graph to model and leverage item difficulty adequately. Then we design a Difficulty-driven Hierarchical Reinforcement Learning (DHRL) framework to facilitate learning paths with efficiency and smoothness. Finally, extensive experiments on three different simulators demonstrate our framework achieves state-of-the-art performance.

CCS CONCEPTS

• **Applied computing** → **E-learning**; • **Information systems** → **Recommender systems**.

KEYWORDS

Learning Path Recommendation, Reinforcement Learning

ACM Reference Format:
Haotian Zhang, Shuanghong Shen, Bihan Xu, Zhenya Huang, Jinze Wu, Jing Sha, and Shijin Wang. 2024. Item-Difficulty-Aware Learning Path Recommendation: From a Real Walking Perspective. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3637528.3671947

*Zhenya Huang is the corresponding author.

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 979-8-4007-0490-1/24/08
https://doi.org/10.1145/3637528.3671947

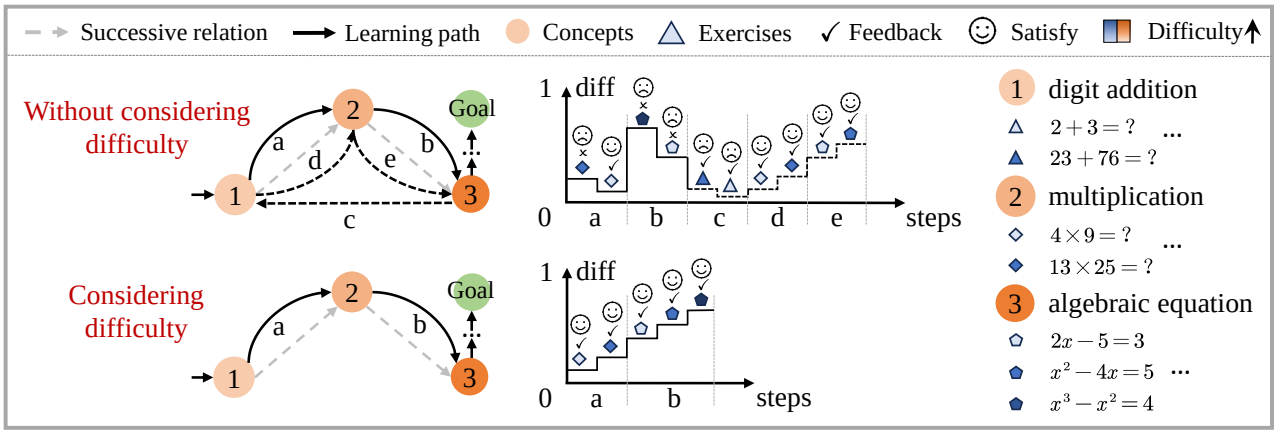


Figure 1: Contrasting Learning Path Recommendation: Considering vs. Without Considering Item Difficulty. When mastering concept 3 (i.e. "algebraic equation"), the LPR method arranges the learning path for learners to first master prerequisite concepts (i.e., path $a \rightarrow b$ from concept 1 to 3) and recommends exercises for practice corresponding to these concepts at each step.

1 INTRODUCTION

Learning path recommendation (LPR) as an essential component of adaptive learning has seen significant adoption in recent years [25, 38]. Unlike traditional education which employs a "one-size-fits-all" strategy, learning path recommendation aims to generate personal learning paths that consider individual differences [2, 21, 28].

In the literature, many works have been devoted to the LPR task. Existing methods can be categorized into two approaches for path generation: complete generation, where a fixed path is provided to learners at once, and step-by-step generation, where a dynamic path is generated based on real-time feedback from learners [2, 29]. Among them, the complete generation approach is often considered inflexible as it may disregard the evolving knowledge states (i.e. mastery level) of learners [7, 28]. In contrast, the step-by-step based methods, which better account for the dynamic interaction between learners and items (e.g. concepts, exercises, etc.), have increasingly gained prominence and are the main focus of this paper. In this branch, some works use traditional recommendation algorithms or deep learning-based methods to recommend similar learning paths to comparable users [9, 27]. On the other hand, since learning path recommendation can be regarded as a sequential decision making problem, some works adopt advanced Reinforcement Learning (RL) methods by formulating the problem as a Markov Decision Process (MDP) [21, 25, 33].

Despite the significant success achieved by these methods, there are still some underlying issues that need to be addressed. Intuitively, the learning process on the learning path can be metaphorically likened to walking. From a real walking perspective, most existing methods tend to exhibit the following two issues: 1) The path may be rough: In the learning path, the included items often present a range of difficulty levels, with the difficulty levels fluctuating unpredictably. This dynamic nature of learning can be likened to walking a dark road with uneven terrain. As illustrated in Figure 1, when the difficulty of items sharply rises or falls, e.g., recommend item $x^3 - x^2 = 4$ before item $2x - 5 = 3$ is correctly solved or recommend items similar to $2+3$ after item $23+76$ has been solved successfully, it can lead to a decrease in learners' learning interest and satisfaction. 2) The path may be inefficient: Treating different difficulty levels of items in the learning path with equal effort is akin to taking the same number of steps regardless of the distance of the journey. As depicted by the dashed backtracking path of $c \rightarrow d \rightarrow e$ in the higher half of Figure 1, allowing learners only a few practice attempts on very challenging items before switching to others (e.g., only two attempts on concept 3, same as the easier concept 2), or persisting with a difficult item despite numerous attempts without mastery, can result in inefficiencies in the learning journey. The root cause of these issues lies in the disregard for item difficulty, which consequently hinders their ability to recommend efficient and smooth learning paths that meet learners' satisfaction.

In this paper, to conquer the above issues, we propose a novel method named Difficulty-constrained Learning Path Recommendation (DLPR) to achieve a more satisfactory learning path with efficiency and smoothness by considering the difficulty of items. Specifically, unlike most previous methods that primarily focused on the granularity of single items (e.g. concepts), to adequately capture the difficulty characteristics of different items and leverage higher-order information, we first explicitly categorize items into learning items (e.g. concepts or skills) and practice items (e.g. exercises or questions). Then construct a hierarchical graph where each learning item is associated with one or more practice items. Information from multiple levels of the hierarchy is aggregated using Hierarchical Graph Neural Network (HGNN) [33]. Next, we develop the Difficulty-driven Hierarchical Reinforcement Learning (DHRL) framework, which consists of two agents with item difficulty awareness. The high-level L-Agent operates at the learning item level, responsible for selecting subsequent learning items for practice. The low-level P-Agent operates at the practice item level, selecting practice items related to the learning item chosen by the L-Agent. Additionally, a Knowledge State Estimation module is introduced to generate agents' states with difficulty awareness. Further, we devise a Communication Mechanism between the two agents to facilitate the generation of satisfactory paths in environments with fluctuating item difficulty. This mechanism allows for the exchange of difficulty-aware information, enabling informed decision-making and effective coordination between the agents. By leveraging this communication mechanism, the agents can adapt

their strategies and ensure the generation of optimal learning paths that align with the appropriate item difficulty.

Our main contributions are summarized as follows:

- We explicitly consider item difficulty, analyzing and addressing existing learning path issues from a real walking perspective. This approach resolves the existing "rough" and "inefficient" aspects of learning paths, further enhancing both the efficiency and smoothness of the learning journey.
- To fully leverage item difficulty information, we constructed a hierarchical graph of learning and practice items. Further, a Difficulty-driven Hierarchical Reinforcement Learning framework (DHRL) is devised. Synchronous consistency of path generation can be achieved through the division of labor and collaboration between two agents with item difficulty awareness.
- We validate our model in three simulators based on two benchmark datasets. Our method achieves state-of-the-art performance with efficiency.

2 RELATED WORK

2.1 Learning Path Recommendation

In online education, Learning Path Recommendation (LPR) stands as a crucial undertaking, which refers to planning and designing a structured learning path for learners, enabling them to systematically and orderly acquire knowledge and skills [21, 25].

Researchers have proposed various methods for the LPR task. The existing LPR methods can be summarized into two categories according to the approach of path generation [2, 29]: (1) Complete generation, a complete path of a specified length is generated and provided to learners at once. (2) Step-by-step generation, a dynamic path of a varying length is generated in real-time by considering the feedback from learners' interactions at each step with the next item recommended [21, 25].

For branches of complete generation, many methods have been proposed using different algorithms and techniques, such as decision tree classifier [22], depth first traversal algorithm [49], recurrent neural network [48], etc. One of the most representative works is proposed by Chen et al. [2], which formulated the recommendation task under a set-to-sequence paradigm and used an encoding-decoding structure to achieve efficient Ranking-Based Concept-Aware Learning Path Recommendation. Although complete generation methods are widely used by researchers to generate learning paths, they have several drawbacks. One of the main disadvantages is ignoring users' performance and their cognitive changes during the learning process, which may lead to users wasting time on inappropriate or unmanageable paths [28]. Due to its ability to better consider the dynamic interaction between learners and items, the step-by-step based methods are rapidly gaining prominence. In this branch, some works use traditional recommendation algorithms or deep learning-based methods, e.g., using evolutionary algorithms [12], matrix factorization [27], bayes theorem [41] etc. On the other hand, since learning path recommendation can be regarded as a sequential decision making problem, some works adopt advanced reinforcement learning methods. For example, Liu et al. [25] used the actor-critic framework with cognitive navigation as a recommender. Further, Li et al. [21] implemented efficient goal planning and achieving through hierarchical reinforcement learning. Despite their success, current methods often neglect item difficulty, leading to unsatisfactory learning path recommendations.

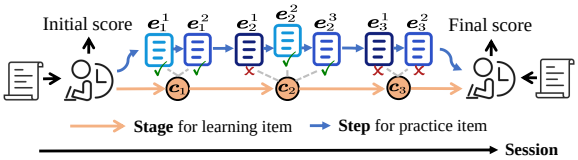


Figure 2: Illustration of Learning Process in One Session.

2.2 Hierarchical Reinforcement Learning in Education

Hierarchical Reinforcement Learning (HRL) is a reinforcement learning method designed to solve decision-making problems in complex tasks [16, 33]. In the educational data mining area, Zhou et al. [46, 47] exploited hierarchical reinforcement learning to make decisions at different levels of granularity for effective pedagogical policy induction. Lin et al. [23] and Zhang et al. [44] introduced HRL to course recommendation for capturing learner's different preferences and interests. It is worth mentioning that, in the field of learning path recommendation, GEHRL proposed by Li et al. [21] implemented efficient goal planning and achieving through Hierarchical Reinforcement Learning. Based on the demonstrated effectiveness of HRL in the field of education and the hierarchical graph constructed in this paper, we have chosen to employ HRL for the task of learning path recommendation. Specifically, our work differs from GEHRL in two significant aspects: 1) GEHRL exclusively employs hierarchical reinforcement learning within the same learning item graph, with both two agents sharing the same action space. In contrast, our approach extends hierarchical reinforcement learning to operate across two item graphs, resulting in distinct action spaces for each of the two agents. 2) Furthermore, our approach incorporates a mutual communication mechanism between the two agents, in contrast to the one-way guidance from the higher-level agent to the lower-level agent employed in GEHRL.

3 PROBLEM AND FRAMEWORK OVERVIEW

In this section, we first formalize the learning path recommendation problem and then introduce the overview of our framework.

3.1 Problem Statement

We focus on the issue of step-by-step recommendations for session-based learning paths based on real-time interactions [21, 25]. As mentioned above, the learner's learning process typically involves two types of items: learning items (e.g. concepts or skills) and practice items (e.g. questions or exercises). Without loss of generality, we denote the learning item set as $\mathcal{LI} = \{c_1, c_2, ..., c_M\}$ and practice item set as $\mathcal{PI} = \{e_1, e_2, ..., e_N\}$. The learner's learning goals are denoted as $\mathcal{G} = \{g_1, g_2, ...\}$, where $g_i \in \mathcal{LI}$.

A typical session-based (e.g. chapters) learning path is developed as shown in Figure 2. Before starting learning, a learner is tested on his learning goals and gets an initial score E_s. Then the learning can be cut into several stages. In each stage, the learner will study a certain learning item c_i, and at each step of the stage, the learner is presented with a practice item e_i^j to comprehend c_i. Subsequently, feedback $score_i^j \in \{0, 1\}$ is provided after practice, where 1 indicates

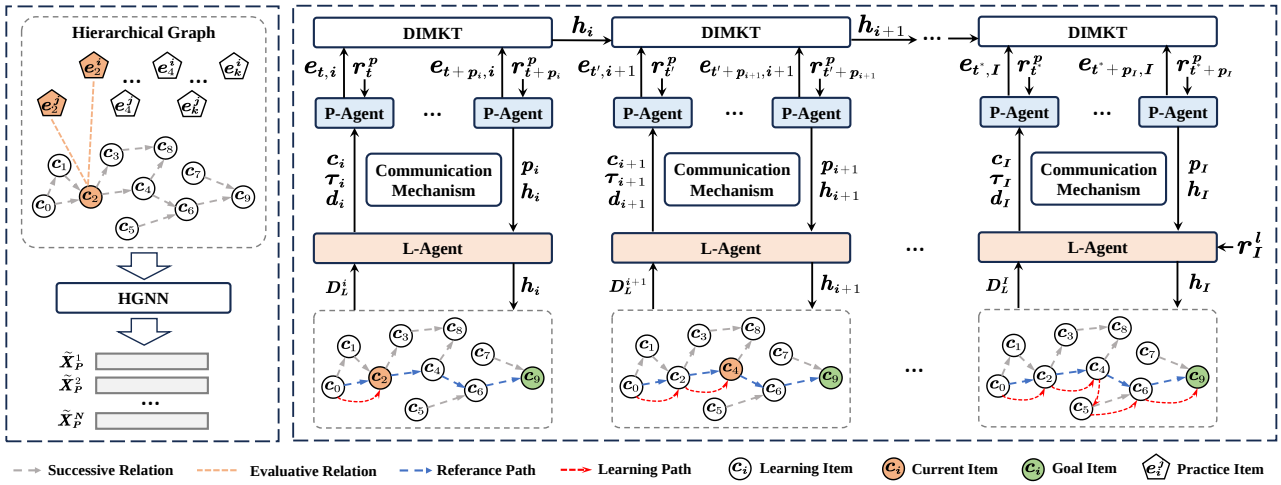


Figure 3: Framework Overview. On the left is the Hierarchical Graph (HG) module, which establishes item relationships and uses HGNN to extract enriched representations, emphasizing item difficulty information. The right is the DHRL module, which employs reinforcement learning with L-Agent and P-Agent to generate efficient and smooth learning paths.

the learner has mastered the item e_i^j and 0 for the opposite. All item-score pairs constitute the historical learning record, denoted as $\mathcal{H} = (e, score)$. After time t, $(e_t, score_t)$ is added to the historical record, i.e., $\mathcal{H}_t = \mathcal{H}_{t-1} \cup (e_t, score_t)$. Finally, a learning path is generated step-by-step as $\mathcal{P} = (\rho_1, \rho_2, \rho_3, ...)$, where ρ_i represents a learning item or practice item. Here, we set ρ_i as a practice item to align with previous works. After completing the entire learning path, a final test is taken on the learning goals to obtain a final score E_e. Then we can calculate the learning effectiveness E_p [25]:

$$E_p = \frac{E_e - E_s}{E_{sup} - E_s}, \tag{1}$$

where E_{sup} is the full score of the examination, which equals to the number of learning goals. Our goal is to maximize the E_p by providing an effective learning path.

3.2 Framework Overview

Figure 3 presents an overview of DLPR, which consists of two primary modules. The left Hierarchical Graph module (Section 4) establishes relationships between learning and practice items, using a hierarchical graph neural network to extract enriched item representations, emphasizing item difficulty. The right DHRL module (Section 5), uses hierarchical reinforcement learning with a high-level L-agent for learning items and a low-level P-agent for practice items, collaboratively generating efficient and smooth learning paths. The Hierarchical Graph provides the structural and representational foundation for the DHRL module.

4 HIERARCHICAL GRAPH ENHANCED ITEM REPRESENTATION

In this section, we present a concise overview of the construction and representation of the Hierarchical Graph for items. This module thoroughly captures the successive relationships among learning items and their associations with practice items while considering item difficulty, serving as a crucial foundation for subsequent item selection and recommendations.

4.1 Graph Construction

To comprehensively model and consider the difficulty of different items and their relationships, we categorize items into learning and practice items and construct a hierarchical graph where one learning item is associated with one or more practice items.

Specifically, the hierarchical graph of items is defined as $HG = \{V, E\}$, where V is the set of items and E is the set of edges. As shown in Figure 3, V consists of learning items and practice items and E contains successive edges between learning items and evaluative edges between learning items and practice items. To be specific, a successive edge (c_i, c_j) represents that c_i (e.g. multiplication) is logically the learning basis for c_j (e.g. algebraic equation), while an evaluative edge (e_m, c_n) indicates that the practice item e_m (e.g. 4×9) assesses the learning item c_n (e.g. multiplication) [24, 39].

4.2 Graph Representation

For more comprehensive and effective modeling and utilization of item difficulty, we first effectively represent item nodes by integrating difficulty information using MLP, and then aggregate high-order and structural information through HGNN [42, 45].

Specifically, following some previous works [14, 26, 36, 43], we calculate the difficulty of the practice item e_j as follows:

$$DP_j = \frac{\sum_{j=1}^{|S_j|} a_j == 0}{|S_j|} \times \lambda_p, \tag{2}$$

where S_j is the number of learners who answer the practice items e_j. $a_j == 0$ indicates learners that answered incorrectly and λ_p represents the predefined level of item difficulty. Similarly, learning item difficulty DL is calculated following a similar process.

Further, we represent all of the items and their difficulty with embeddings. Specifically, we use an embedding matrix $\mathbf{E}_p \in \mathbb{R}^{|\mathcal{PI}| \times d_p}$ to represent all practice items, where d_p is the dimension. Besides, we represent the difficulty embedding of practice items by an embedding matrix $\mathbf{E}_{dp} \in \mathbb{R}^{\lambda_p \times d_{dp}}$, where λ_p represents the predefined level of item difficulty and d_{dp} is the dimension. The $\mathbf{E}_l$ and $\mathbf{E}_{dl}$ of learning items are represented similarly. Then we can obtain

the difficulty-aware embedding $\mathbf{X}_L^i$ of learning item c_i and $\mathbf{X}_P^j$ of practice item e_j calculating as follows:

$$\mathbf{X}_L^i = \mathbf{W}_L^T[\mathbf{E}_l^i \oplus \mathbf{E}_{dl}^i] + \mathbf{b}_L, \tag{3}$$

$$\mathbf{X}_P^j = \mathbf{W}_P^T[\mathbf{E}_p^j \oplus \mathbf{E}_{dp}^j] + \mathbf{b}_P, \tag{4}$$

where $\mathbf{W}_L$, $\mathbf{W}_P$ are the weight matrices, $\mathbf{b}_L$, $\mathbf{b}_P$ are the bias terms and $\oplus$ is the concatenation operation.

Finally, for each practice item e_j, it employs the simplified mean aggregation [5, 42] to aggregate features from learning item neighbors and denoted as:

$$\tilde{\mathbf{X}_P^j} = \mathbf{X}_P^j \oplus \frac{1}{|\mathcal{N}_j|} \sum_{i \in \mathcal{N}_j} \mathbf{X}_L^i, \tag{5}$$

where $\tilde{\mathbf{X}_P^j}$ represents aggregated embedding of practice item e_j and $\mathcal{N}_j$ indicates the learning item neighbors of e_j in the hierarchical graph HG.

5 DIFFICULTY-DRIVEN HIERARCHICAL REINFORCEMENT LEARNIN

To create an effective learning path, we use a Knowledge State Estimation module to track learners' evolving knowledge states. This supports two hierarchical agents: a high-level L-Agent, which recommends learning items considering prerequisite relationships, and a low-level P-Agent, which suggests practice items related to those chosen by the L-Agent and manages item difficulty fluctuations for a smooth learning path. To ensure efficiency, the P-Agent limits practice attempts based on difficulty-aware information from the L-Agent. Together, the L-Agent and P-Agent collaborate to generate an efficient and smooth learning path. The pseudo-code of the DHRL module can be found in Appendix A.

5.1 Knowledge State Estimation

Comprehending learners' knowledge state is a prerequisite for recommending suitable learning resources [21, 25]. Knowledge tracing algorithms have been extensively researched to track the evolving knowledge states of learners based on their learning sequences [37]. Previous works on LPR utilized the Deep Knowledge Tracing [34] for assessing learners' knowledge states. However, given our comprehensive utilization of item difficulty, we employ DIfficulty Matching Knowledge Tracing (DIMKT) that considers the item difficulty's influence on the learner's cognitive change [36].

Specifically, we retain the original DIMKT while solely replacing the practice item embedding $\mathbf{x_j}$ with our aggregated embedding $\tilde{\mathbf{X}_P^j}$ through HGNN obtained from Section 4.2. This approach allows us to derive the current learner's knowledge state h_t from their historical learning records $\mathcal{H}_{t-1}$ while taking into full consideration both the items' difficulty and structural correlations. Following [21], we utilize DIMKT's prediction to estimate the h_t rather than directly utilize the hidden vector used in DIMKT.

5.2 L-Agent

As illustrated in Section 3.1, session-based learning paths involve planning specific items for each stage. Therefore, we developed the L-Agent to organize these learning items.

5.2.1 State Encoder. The state of the L-Agent s_i^l at learning stage i contains learning goals $\mathcal{G}$ and the learner's current knowledge state h_{i-1} at the end of stage i-1. Specifically, we use multi-hot encoding to represent learning goals as $\mathcal{G} = \{0, 1\}^M$, where the learning goals' indexes are set 1 and others 0 and M is the number of learning items. The final state of the L-Agent is encoded as:

$$s_i^l = h_{i-1} \oplus \mathcal{G}. \tag{6}$$

5.2.2 Adaptive learning action space. The action of the L-Agent c_i refers to recommending the learning item at learning stage i. If the action space is the whole item set, the search space is very large and inefficient. To constrain the action space, existing methods often employ neighbor sampling or embedding similarity sampling [21, 25]. However, these methods are shortsighted, focusing only on items near the current focal item. They work well when the focal item is close to the target, but when they are far apart, shortsightedness may cause navigation to deviate, reducing learning path efficiency. To overcome these limitations, we propose a reference-path-assisted adaptive method to help the agent determine the general search direction and adaptively expand toward the goal item. Specifically, we first use the A* algorithm [6] to generate the shortest path between the starting item and the goal item as a reference path. When generating the learning path, we adjust it based on the learner's real-time learning progress. If the learning is smooth, we continue forward along the reference path. Otherwise, we search with the current item as the root node and then learn progressively from nearby to distant items. Whether the learning is smooth is determined by assessing whether the learner has achieved the expected mastery state after completing a certain amount of practice items. Hence, we can dynamically ascertain the candidate action space D_L. The pseudo-code can be found in Appendix B.

5.2.3 Policy. After obtaining the learner's knowledge state and the action space, we still need to determine the optimal candidate learning item, which is most beneficial for achieving our learning goals. We opt to employ the Proximal Policy Optimization (PPO) [35] to generate actions among D_L given by Section 5.2.2. Specifically, we use a policy network as the actor to output an action probability from distribution $\pi^L(c_i|s_i^l;\theta_L)$ and a value network $\mathcal{V}^L(s_i^l;\phi_L)$ as the critic to estimate the expected return from each state. where θ_L and ϕ_L are the corresponding network's parameters. The probability distribution of learning items and expected return from each state are calculated as follows:

$$c_i = Softmax(FC(s_i^l)), \tag{7}$$

$$\mathcal{V}^L(s_i^l;\phi_L) = FC(s_i^l), \tag{8}$$

where FC is the fully connected layer.

For the training, we use classical mean squared loss (MSE) to train the critic and train the actor based on the PPO-clip loss function, which is similar to GEHRL [21].

5.2.4 Reward. As mentioned in Section 3.1, our final goal is to maximize the improvement of learners on the learning goals, and we can't avoid learning just because a certain learning item may be difficult, which may lead to failure in achieving our learning goals. Therefore, following previous works [21, 25], the reward is set as the learning effectiveness after the completion of the entire learning

path in Eq. (9), without considering item difficulty variation.

$$r_i^l = \begin{cases} E_p, & \text{if } i \text{ is the last learning stage} \\ 0, & \text{otherwise,} \end{cases} \tag{9}$$

where E_p is obtained using Eq. (1).

5.3 P-Agent

Given that the L-Agent has selected the learning item for the stage, we then build the P-Agent to recommend suitable practice items while managing item difficulty to ensure a smooth learning path.

5.3.1 State encoder. The state of the P-Agent $s_{t,i}^p$ contains learning item c_i in the current learning stage i and the learner's knowledge state h_t at current learning step t. Specifically, we use one-hot encoding to represent the learning item as $c_i = \{0, 1\}^M$, where the learning item's index is set 1 and others 0 and M is the number of learning items. The final state of the P-Agent is encoded as:

$$s_{t,i}^p = h_t \oplus c_i. \tag{10}$$

5.3.2 Relation-constrained practice action space. The action of the P-Agent $e_{t,i}$ is to choose the next practice item for a learner to study learning item c_i. In contrast to L-Agent, P-Agent does not require making action selections within an extensive item space. Instead, it solely concentrates on the practice items associated with the current learning item c_i at learning stage i. Specifically, the candidate action space D_P^i is got as follows:

$$D_P^i = \Psi(c_i) \in \mathcal{PI}, \tag{11}$$

where Ψ is a mapping function to obtain the set of practice items associated with c_i from HG.

Taking item difficulty into account, the P-Agent starts by favoring items near the initial difficulty d_i obtained using Eq. (15) and later considers the entire D_P^i as the stage progresses.

5.3.3 Policy. The policy of P-Agent is how to select the next practice item for a learner. Here we use an actor-critic framework [19] similar to L-Agent in Section 5.2.3. The main difference between actor-critic and PPO is in the actor's training algorithm. The actor-critic trains based on the simple policy gradient [21].

5.3.4 Reward. As discussed before, learners usually learn knowledge gradually, and the dramatically varying difficulty levels of recommended practice items may lead to a rough learning path thus decreasing learners' interest [15]. Therefore, we design a reward r_t^{p1} applying a method proposed by Huang et al. [15] to control the difficulty fluctuation as:

$$r_t^{p1} = \mathcal{L}(DP_t, DP_{t-1}) = -(DP_t - DP_{t-1})^2, \tag{12}$$

where DP_t obtained in Eq. 2 can be retrieved from the attribute of the selected practice item $e_{t,i}$.

In addition, to assess the effectiveness of the current practice items, we design the reward r_t^{p2} by examining the changes in learners' knowledge states h, which were obtained from DIMKT.

$$r_t^{p2} = \begin{cases} h_i - h_{i-1}, & \text{if } h_i > Thre \\ 0, & otherwise, \end{cases} \tag{13}$$

where h_i and h_{i-1} denote the knowledge state at the end and beginning of the current stage, respectively. $Thre$ is the predetermined threshold for the knowledge state.

Finally, the reward of P-Agent at each step is merged with α_1, α_2 balance coefficients as:

$$r_t^p = \alpha_1 \times r_t^{p1} + \alpha_2 \times r_t^{p2}, \alpha_1, \alpha_2 \in [0, 1]. \tag{14}$$

5.4 Communication Mechanism

To ensure effective coordination between the L-Agent and P-Agent and achieve efficient learning path construction, we implement a communication mechanism between the two agents. The P-Agent organizes practice items' initial difficulty and controls the learner's maximum practice attempts based on item-difficulty information passed from the L-Agent.

5.4.1 Initial difficulty control. Initial difficulty control makes L-Agent provide P-Agent with the initial difficulty information of practice items as a reference based on the learning item chosen by L-Agent in the current stage i and the learner's current knowledge state h_i. This enables the P-Agent to have a good starting point for recommending practice items that are suitable in difficulty for the learner. Specifically, inspired by Rasch method [11], we calculate the initial difficulty for the next learning item as:

$$d_i = \dot{h_i} + ln(\frac{Prob_i}{1 - Prob_i}), \tag{15}$$

where d_i represents the initial reference difficulty level of the practice items for learning item c_i. $\dot{h_i}$ represents the learner's mastery level of c_i, which is retrieved from h_i using indexing, and $Prob_i$ represents the probability of a learner answering an item correctly, which is flexible in practice.

5.4.2 Practice tolerance control. When learners learn learning items through interactions with practice items, this process should be finite. In other words, learners cannot practice the same item indefinitely to master it, as it is inefficient and unreasonable. However, since each learner has different learning levels and paces, the tolerance for the number of practice items should be individualized and dynamically adjusted. Specifically, denote the tolerance at stage i as τ_i, we get the tolerance as follows:

$$\tau_i = f(p_{i-1}, h_{i-1}, \tau_{i-1}, DL_i), \tag{16}$$

where p_{i-1} and h_{i-1} represent the number of practice items attempted for learning item c_{i-1} and the mastery level at the end of the last learning stage $i-1$. As shown in Figure 3, this information originates from the P-Agent, and there is mutual communication between the two agents, which ensures smooth collaboration. DL_i is the difficulty level of c_i learning at this stage. f is an MLP that will be trained. For training of f, we initially simulate 5000 student learning paths in the simulation system without limiting tolerance. This yields actual p_i, h_i, DL_i for each learning stage i. To align tolerance with actual practice frequency, we set $\tau_i = p_i$, resulting in training data p, h, τ, DL, which is iteratively trained.

6 EXPERIMENTS

In this section, we first introduce the datasets and simulators. Then, we demonstrate the superiority of our method through extensive experimentation and evaluation.

6.1 Datasets

Our experiments are performed on two real-world public datasets: Junyi[1] and ASSIST09[2]. Both datasets contain learners' learning log data. For the Junyi dataset, we use the "topics" field as learning items, which are commonly used in education. Additionally, the Junyi dataset provides a prerequisite graph of items, and we use it to construct the HG in Section 4. However, the ASSIST09 dataset does not provide information about the knowledge structure, so we construct a transition graph [30] as an estimation of the knowledge structure [21]. The statistics of datasets can be found in Appendix C.

6.2 Simulators

For evaluation, a key issue is that existing realistic data only contains static information. This data cannot directly analyze if practice items not in a sequence can be answered correctly [15]. Hence, it's unsuitable for evaluating learning paths or training reinforcement learning agents. Following previous works [2, 21, 25], to evaluate different methods' recommending effects, we use two kinds of similar simulators built in [25]: Knowledge Structure based Simulator (KSS) and Knowledge Evolution based Simulator (KES).

KSS is a rule-based system evaluating learner performance based on Item Response Theory (IRT) [10]. KES is a data-based system utilizing the DKT model [34] to simulate knowledge state changes of learners. Considering the impact of item difficulty on learner knowledge state [31, 40], we develop DIMKT [36] to build two different simulators KES-Junyi and KES-ASSIST based on datasets Junyi and ASSIST09 respectively, and initial logs from a specific dataset are used to simulate the learner's initial state [21].

6.3 Experimental Setup

We implement our learning path recommendation framework using pytorch [32], gym [1], and the simulator code from previous works [21, 25]. In our experiments, we employed the same data partitioning, data preprocessing, and simulator settings as in [25]. However, a significant change was made by replacing the original DKT component with DIMKT, which has a 128-dimensional embedding layer, 50 difficulty levels, and a learning rate of 0.002. To align with DIMKT, we set λ_P in Eq. (2) to be 50. Additionally, we configured the parameters α_1 and α_2 in Eq. (14) to be both 0.5. Moreover, we set the mastery threshold as 0.6 in Eq. (13), which is a common passing threshold in education. In Eq. (15), we set $Prob_i$ as 0.5, indicating that the probability of a learner answering the item correctly or incorrectly is equal. We use Adam [18] as our optimizer and the learning rate is set to be 0.001. Our code is available at https://github.com/sosweetzhang/DLPR.

6.4 Baseline Approaches

- KNN: KNN [4] find similar learners based on their learning paths. The algorithm determines the next learning item for a new learner based on the paths of the nearest identified learners.
- GRU4Rec: GRU4Rec [13] is a classic model taking the session sequence as input and generating a probability distribution that predicts the learning items likely to appear in the next step.
- DQN: DQN [3] uses a neural network to assess action values and recommends the action with the highest value.
- Actor-Critic: Use a GRU encoder and vanilla actor-critic framework [19] as a recommender.
- CB: Contextual Bandits [17] is a learning path recommendation method that treats the recommendation process as a contextual bandit problem.
- RLTutor: RLTutor [20] is an adaptive tutoring system that combines a model-based RL approach with DAS3H [8] for learning item recommendation.
- CSEAL: CSEAL [25] is a method using an actor-critic framework with cognitive navigation as a recommender.
- GEHRL: GEHRL [21] implements efficient goal planning and achieving through Hierarchical Reinforcement Learning. Specifically, GEHRL-EB is compared for its better performance.

Following previous works [2, 21, 25], we evaluate these methods based on the promotion E_p (Eq. (1)) given by simulators.

6.5 Overall Performance Comparison

Table 1 presents the average E_P values of all models across the three simulators, revealing several important insights. Firstly, our proposed DLPR outperforms all baselines in all three simulators, highlighting the necessity and value of considering item difficulty in learning path recommendation. Secondly, Reinforcement Learning methods, such as CSEAL, GEHRL, and DLPR, exhibit superior performance due to real-time interactive feedback, long-term cumulative rewards, and consideration of cognitive structure and action space constraints. Thirdly, a general decline in performance is observed as the number of recommended steps decreases in the simulated environment, underlining the need for improved recommendation performance to enhance learning path efficiency. Finally, as the number of items increases and the structure becomes more complex, the diminishing returns of specific learning steps align with intuition. This suggests that learning a larger domain of knowledge requires more steps to reach distant learning goals, indicating the need for additional time and effort to grasp new concepts and skills as knowledge expands.

It should be clear that negative values in KES-Junyi and KES-ASSIST09 are due to the presence of a large number of items in these simulators, some of which never occurred in the training data. This can lead to unstable predictions by the KT model, resulting in negative rewards [21].

6.6 Learning Path Efficiency and Smoothness

To evaluate the effectiveness and smoothness of DLPR in the learning path recommendation task, we compared it with baselines using three metrics while achieving learning goals. To be specific, we let these methods recommend items for a learner to master three specific concepts random selected (i.e. $E_e = E_{sup}$ in Eq. (1) which means the learner should correctly answer all questions in the final test). The **Learning-Steps** metric captures the average number of steps required to achieve learning goals, while the **Cog-Gap** (Cognitive Gap) metric measures the mean difference between the normalized difficulty levels of practice items and learners' knowledge state values calculated as Eq. (17). Considering that recommending difficult items to beginners or simple items to experts will increase

[1] https://pslcdatashop.web.cmu.edu/DatasetInfo?datasetId=1198
[2] https://sites.google.com/site/assistmentsdata/home/2009-2010-assistment-data

Table 1: Performance comparison for learning path recommendation methods. Existing state-of-the-art results are underlined and the best results are bold. Our DLPR is compared with the SOTA GEHRL and * indicates a p-value < 0.05 in the t-test.

		KNN	GRU4Rec	DQN	Actor-Critic	CB	RLTutor	CSEAL	GEHRL	**DLPR**
KSS	step=5	0.1005	0.1124	0.1559	0.1437	0.0852	0.1999	0.2095	0.2321	**0.5583***
	step=10	0.3133	0.2767	0.2836	0.4072	0.2643	0.4008	0.4233	0.5644	**0.7294***
	step=20	0.2972	0.1998	0.3236	0.4931	0.2614	0.5303	0.5716	0.7426	**0.8305***
KES-Junyi	step=5	-0.0902	-0.0047	0.0299	0.1004	0.0666	-0.0007	0.0975	0.1198	**0.2049***
	step=10	-0.1455	-0.0721	-0.1058	0.1671	0.1451	-0.0379	0.2021	0.2278	**0.3835***
	step=20	0.1343	0.0993	0.1536	0.1916	0.2098	-0.1034	0.2505	0.4206	**0.6124***
KES-ASSIST09	step=5	-0.0549	-0.0536	-0.0495	-0.0004	-0.0563	-0.0611	0.0482	0.0751	**0.0807***
	step=10	-0.0731	-0.1003	-0.0934	-0.0327	-0.1294	-0.1096	0.0637	0.0918	**0.1544***
	step=20	-0.0932	-0.1344	-0.0267	0.0676	0.0038	0.0784	0.1009	0.1971	**0.3283***

Table 2: Learning path efficiency and smoothness for learning path recommendation methods. It should be noted that "-" in the table indicates that the method cannot achieve absolute promotion and meet the learning goals.

		KNN	GRU4Rec	DQN	Actor-Critic	CB	RLTutor	CSEAL	GEHRL	**DLPR**
KSS	Learning-Steps	54	72	48	41	56	34	29	14	**8**
	Cog-Gap	0.3216	0.3889	0.3709	0.3921	0.3514	0.3336	0.3639	0.2436	**0.1017**
	Diff-MAD	0.3886	0.3267	0.3574	0.3749	0.3309	0.3696	0.3231	0.2704	**0.1200**
KES-Junyi	Learning-Steps	-	-	-	-	219	-	128	96	**33**
	Cog-Gap	-	-	-	-	0.3413	-	0.3898	0.3641	**0.1313**
	Diff-MAD	-	-	-	-	0.3687	-	0.2791	0.3144	**0.1614**
KES-ASSIST09	Learning-Steps	-	-	-	-	-	536	378	185	**69**
	Cog-Gap	-	-	-	-	-	0.4197	0.3823	0.3562	**0.1391**
	Diff-MAD	-	-	-	-	-	0.2947	0.2893	0.2998	**0.1577**

the Cog-Gap, leading to unnecessary practice and more Learning-Steps, these two metrics are designed to reflect the efficiency of the learning path. Additionally, the **Diff-MAD** (Difficulty Mean Absolute Deviation) metric quantifies the average absolute difference in difficulty levels of practice items throughout the learning process calculated as Eq. (18). This metric measures the variation in item difficulty, reflecting the smoothness of the learning path.

$$\text{Cog-Gap} = \frac{\sum_{j=1}^{n} |h_j - DP_j|}{n}, \tag{17}$$

$$\text{Diff-MAD} = \frac{\sum_{j=1}^{n-1} |DP_{j+1} - DP_j|}{n-1}, \tag{18}$$

where h_j represents the learner's knowledge state on practice item e_j and DP_j indicates the difficulty level of e_j calculated using Eq. (2). n is the number of learning steps.

Table 2 presents the experimental results, indicating that DLPR effectively recommends practice items that align with the learner's knowledge state while controlling difficulty variations. It significantly reduces the number of learning steps and improves the efficiency and smoothness of the learning path. Please note that "-" in the table indicates that the method cannot achieve significant improvement and meet the learning goals, therefore making it impossible to calculate the metrics used to assess learning paths that achieve the learning goal.

6.7 Ablation Study

In this section, we perform an ablation study on Junyi to analyze the impact of some key elements in DLPR. We consider four variants of DLPR, where each variant removes one element from the original DLPR. The details of these variants are as follows:

Table 3: Results of ablation experiments.

	Learning Steps	Cog-Gap	Diff-MAD
w/o ACS	154	0.1707	0.2099
w/o Init-diff	86	0.2133	0.1796
w/o Torlerance	94	0.2654	0.1832
w/o Diff-reward	61	0.3516	0.3235
DLPR	**33**	**0.1313**	**0.1614**

- DLPR w/o ACS, which refers to DLPR without adaptive learning action space in section 5.2.2.
- DLPR w/o Init-diff, which eliminates the guidance of initial difficulty control in section 5.4.1.
- DLPR w/o Torlerance, which excludes the practice tolerance control in section 5.4.2.
- DLPR w/o Diff-reward, which removes the reward r_t^{p1} that controls difficulty variations in section 5.3.4.

From Table 3, several key findings can be drawn. Firstly, the complete model achieved the best overall performance. Secondly, ACS had the most significant impact on the number of learning steps, emphasizing the importance of correct choices and learning direction. Selecting the right path is crucial for success. Diff-reward had the greatest influence on cognitive differences and difficulty smoothness, validating the effectiveness of our reward design in controlling difficulty variations. Init-diff and Tolerance played varying roles in all three factors. The collaborative synergy of the components leads to optimal results, as the absence of any component results in a decline in performance.

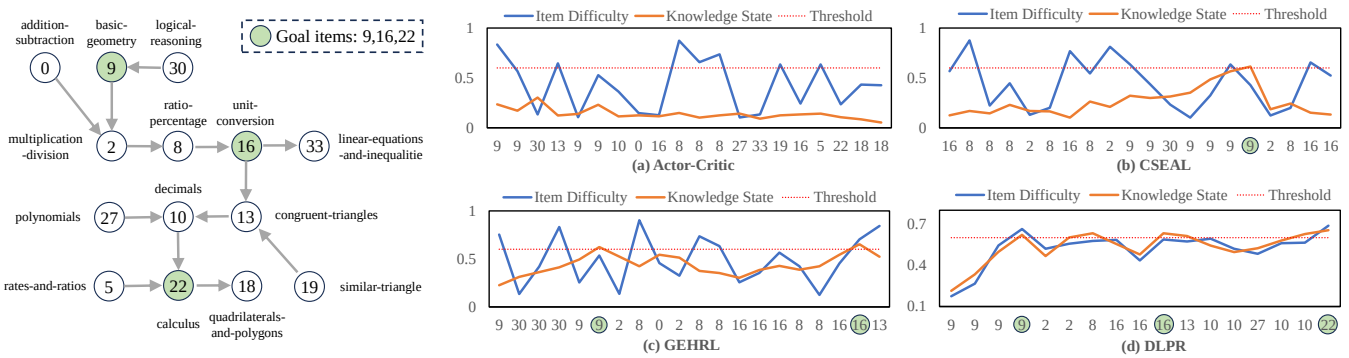


Figure 4: Visualization of different learning paths recommended by four selected methods for the same learning goal items.

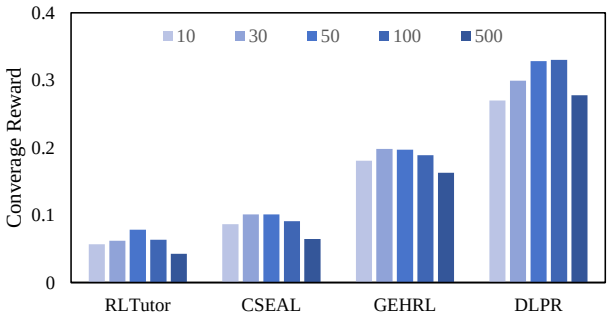


Figure 5: The performance of some methods under different predefined difficulty levels.

6.8 Impact of Difficulty Level Segmentation

As our main concern is the item difficulty, to investigate the impact of item difficulty level on learning path efficiency and smoothness, we conducted experiments using the ASSIST09 dataset, which offers richer items. Similar to the common practice of categorizing difficulty levels into three tiers: easy, medium, and hard [39], here we aim for a more granular analysis of difficulty by classifying it into 10, 30, 50, 100, and 500 levels (λ_P in Eq.(2)). These levels represent different subdivisions of the same difficulty. Further, we evaluated the performance of selected RLTutor, CSEAL, GEHRL, and DLPR under different difficulty levels with a fixed number of steps set at 20.

Figure 5 illustrates the results and highlights the importance of properly defining difficulty level segmentation. When the segmentation is too low, the distinguishability between items decreases, leading to poorer performance in modeling learner interactions and knowledge state changes. This also reduces the selectivity in recommending suitable learning items, resulting in inferior outcomes. On the other hand, excessive segmentation also deteriorates the performance of the models. Additionally, with excessively fine-grained segmentation, there may be a scarcity of appropriately difficult practice items for the same learning item. Interestingly, lower difficulty segmentation may favor methods that do not consider difficulty (e.g. CSEAL), as reduced granularity reduces difficulty fluctuations.

6.9 Case Study

Figure 4 shows the learning path recommendations of four models in the KES-junyi simulator for a learner with specific goals. The left side of Figure 4 provides a partial knowledge structure diagram of the KES-junyi environment for better understanding. On the right side, four images depict the recommended paths by the selected models, along with changes in practice item difficulty and the learner's knowledge states on the learning item. To be specific, the horizontal axis represents the recommended items in the learning path, where item ids may repeat due to multiple practices of the same item. The vertical axis denotes the normalized knowledge mastery and difficulty levels, ranging from 0 to 1. The figure reveals that: (1) The Actor-Critic method ignores knowledge structure and item difficulty, resulting in disorganized recommendations and failure to achieve goals. (2) CSEAL considers knowledge structure but lacks planning between goals, resulting in poor learning outcomes with only one item being accomplished. (3) GEHRL plans for multiple goals but overlooks item difficulty, leading to detours and incomplete goal achievement. (4) Our method, considering both knowledge structure and item difficulty, efficiently recommends paths with smoothness and achieves all goals.

7 CONCLUSION

In this paper, we addressed two special issues "rough" and "inefficient" of learning paths from a real walking perspective and proposed an effective and smooth learning path recommendation method considering item difficulty. To be specific, we constructed a hierarchical graph of learning and practice items to capture their difficulty and higher-order correlations. Then we designed a Difficulty-driven Hierarchical Reinforcement Learning framework to generate learning paths smoothly and efficiently through two agents' collaboration. Extensive experimental results validate the superiority of our framework in providing highly satisfactory learning path recommendations by thoroughly considering item difficulty. Nevertheless, as we primarily conducted the experiments in the simulated environments, further research will develop the system and test the model in practical settings in the real-world environments to investigate broader impacts.

ACKNOWLEDGMENTS

This research was partially supported by grants from the National Science and Technology Major Project (No. 2022ZD0117103), the National Natural Science Foundation of China (No. 62106244), and the University Synergy Innovation Program of Anhui Province (No. GXXT-2022-042).

REFERENCES

[1] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *arXiv preprint arXiv:1606.01540* (2016).

[2] Xianyu Chen, Jian Shen, Wei Xia, Jiarui Jin, Yakun Song, Weinan Zhang, Weiwen Liu, Menghui Zhu, Ruiming Tang, Kai Dong, et al. 2023. Set-to-sequence ranking-based concept-aware learning path recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 5027–5035.

[3] Yunxiao Chen, Xiaoou Li, Jingchen Liu, and Zhiliang Ying. 2018. Recommendation system for adaptive learning. *Applied psychological measurement* 42, 1 (2018), 24–41.

[4] Thomas Cover and Peter Hart. 1967. Nearest neighbor pattern classification. *IEEE transactions on information theory* 13, 1 (1967), 21–27.

[5] Ziqiang Cui, Haolun Wu, Bowei He, Ji Cheng, and Chen Ma. 2024. Diffusion-based Contrastive Learning for Sequential Recommendation. *arXiv e-prints* (2024), arXiv–2405.

[6] František Duchoň, Andrej Babinec, Martin Kajan, Peter Beňo, Martin Florek, Tomáš Fico, and Ladislav Jurišica. 2014. Path planning with modified a star algorithm for a mobile robot. *Procedia engineering* 96 (2014), 59–69.

[7] Guillaume Durand, Nabil Belacel, and François LaPlante. 2013. Graph theory based model for learning path recommendation. *Information Sciences* 251 (2013), 10–21.

[8] Pragya Dwivedi, Vibhor Kant, and Kamal K Bharadwaj. 2018. Learning path recommendation based on modified variable length genetic algorithm. *Education and information technologies* 23 (2018), 819–836.

[9] Lumbardh Elshani and Krenare Pireva Nuçi. 2021. Constructing a personalized learning path using genetic algorithms approach. *arXiv preprint arXiv:2104.11276* (2021).

[10] Susan E Embretson and Steven P Reise. 2013. *Item response theory*. Psychology Press.

[11] Gerhard H Fischer and Ivo W Molenaar. 2012. Rasch models: Foundations, recent developments, and applications. (2012).

[12] Kannan Govindarajan, Vivekanandan Suresh Kumar, et al. 2016. Dynamic learning path prediction—A learning analytics solution. In *2016 IEEE eighth international conference on technology for education (T4E)*. IEEE, 188–193.

[13] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[14] Zhenya Huang, Qi Liu, Enhong Chen, Hongke Zhao, Mingyong Gao, Si Wei, Yu Su, and Guoping Hu. 2017. Question Difficulty Prediction for READING Problems in Standard Tests. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31.

[15] Zhenya Huang, Qi Liu, Chengxiang Zhai, Yu Yin, Enhong Chen, Weibo Gao, and Guoping Hu. 2019. Exploring multi-objective exercise recommendations in online education systems. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1261–1270.

[16] Matthias Hutsebaut-Buysse, Kevin Mets, and Steven Latré. 2022. Hierarchical reinforcement learning: A survey and open research challenges. *Machine Learning and Knowledge Extraction* 4, 1 (2022), 172–221.

[17] Wacharawan Intayoad, Chayapol Kamyod, and Punnarumol Temdee. 2020. Reinforcement learning based on contextual bandits for personalized online learning recommendation systems. *Wireless Personal Communications* 115, 4 (2020), 2917–2932.

[18] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[19] Vijay Konda and John Tsitsiklis. 1999. Actor-critic algorithms. *Advances in neural information processing systems* 12 (1999).

[20] Yoshiki Kubotani, Yoshihiro Fukuhara, and Shigeo Morishima. 2021. RLTutor: Reinforcement Learning Based Adaptive Tutoring System by Modeling Virtual Student with Fewer Interactions. *arXiv preprint arXiv:2108.00268* (2021).

[21] Qingyao Li, Wei Xia, Li'ang Yin, Jian Shen, Renting Rui, Weinan Zhang, Xianyu Chen, Ruiming Tang, and Yong Yu. 2023. Graph Enhanced Hierarchical Reinforcement Learning for Goal-Oriented Learning Path Recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 1318–1327.

[22] Chun Fu Lin, Yu-chu Yeh, Yu Hsin Hung, and Ray I Chang. 2013. Data mining for providing a personalized learning path in creativity: An application of decision trees. *Computers & Education* 68 (2013), 199–210.

[23] Yuanguo Lin, Fan Lin, Wenhua Zeng, Jianbing Xiahou, Li Li, Pengcheng Wu, Yong Liu, and Chunyan Miao. 2022. Hierarchical reinforcement learning with dynamic recurrent mechanism for course recommendation. *Knowledge-Based Systems* 244 (2022), 108546.

[24] Fei Liu, Chenyang Bu, Haotian Zhang, Le Wu, Kui Yu, and Xuegang Hu. 2024. FDKT: Towards an interpretable deep knowledge tracing via fuzzy reasoning. *ACM Transactions on Information Systems* (2024).

[25] Qi Liu, Shiwei Tong, Chuanren Liu, Hongke Zhao, Enhong Chen, Haiping Ma, and Shijin Wang. 2019. Exploiting Cognitive Structure for Adaptive Learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. New York, NY, USA, 627–635.

[26] Sein Minn, Feida Zhu, and Michel C Desmarais. 2018. Improving knowledge tracing model by integrating problem difficulty. In *2018 IEEE International conference on data mining workshops (ICDMW)*. IEEE, 1505–1506.

[27] Amir Hossein Nabizadeh, Daniel Goncalves, Sandra Gama, Joaquim Jorge, and Hamed N Rafsanjani. 2020. Adaptive learning path recommender approach using auxiliary learning objects. *Computers & Education* 147 (2020), 103777.

[28] Amir Hossein Nabizadeh, José Paulo Leal, Hamed N Rafsanjani, and Rajiv Ratn Shah. 2020. Learning path personalization and recommendation methods: A survey of the state-of-the-art. *Expert Systems with Applications* 159 (2020), 113596.

[29] Amir Hossein Nabizadeh, Alípio Mário Jorge, and José Paulo Leal. 2017. Rutico: Recommending successful learning paths under time constraints. In *Adjunct publication of the 25th conference on user modeling, adaptation and personalization*. 153–158.

[30] Hiromi Nakagawa, Yusuke Iwasawa, and Yutaka Matsuo. 2019. Graph-based knowledge tracing: modeling student proficiency using graph neural network. In *IEEE/WIC/ACM International Conference on Web Intelligence*. 156–163.

[31] Zachary A Pardos and Neil T Heffernan. 2011. KT-IDEM: Introducing item difficulty to the knowledge tracing model. In *User Modeling, Adaption and Personalization: 19th International Conference, UMAP 2011, Girona, Spain, July 11-15, 2011. Proceedings 19*. Springer, 243–254.

[32] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[33] Shubham Pateria, Budhitama Subagdja, Ah-hwee Tan, and Chai Quek. 2021. Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys (CSUR)* 54, 5 (2021), 1–35.

[34] Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J Guibas, and Jascha Sohl-Dickstein. 2015. Deep knowledge tracing. *Advances in neural information processing systems* 28 (2015).

[35] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[36] Shuanghong Shen, Zhenya Huang, Qi Liu, Yu Su, Shijin Wang, and Enhong Chen. 2022. Assessing Student's Dynamic Knowledge State by Exploring the Question Difficulty Effect. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 427–437.

[37] Shuanghong Shen, Qi Liu, Zhenya Huang, Yonghe Zheng, Minghao Yin, Minjuan Wang, and Enhong Chen. 2024. A survey of knowledge tracing: Models, variants, and applications. *IEEE Transactions on Learning Technologies* (2024).

[38] Daqian Shi, Ting Wang, Hao Xing, and Hao Xu. 2020. A learning path recommendation model based on a multidimensional knowledge graph framework for e-learning. *Knowledge-Based Systems* 195 (2020), 105618.

[39] Jinze Wu, Haotian Zhang, Zhenya Huang, Liang Ding, Qi Liu, Jing Sha, Enhong Chen, and Shijin Wang. 2024. Graph-based Student Knowledge Profile for Online Intelligent Education. In *Proceedings of the 2024 SIAM International Conference on Data Mining (SDM)*. SIAM, 379–387.

[40] Bihan Xu, Zhenya Huang, Jiayu Liu, Shuanghong Shen, Qi Liu, Enhong Chen, Jinze Wu, and Shijin Wang. 2023. Learning behavior-oriented knowledge tracing. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2789–2800.

[41] Dihua Xu, Zhijian Wang, Kejia Chen, and Weidong Huang. 2012. Personalized learning path recommender based on user profile using social tags. In *2012 Fifth International Symposium on Computational Intelligence and Design*, Vol. 1. IEEE, 511–514.

[42] Xiaocheng Yang, Mingyu Yan, Shirui Pan, Xiaochun Ye, and Dongrui Fan. 2023. Simple and efficient heterogeneous graph neural network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 10816–10824.

[43] Haotian Zhang, Chenyang Bu, Fei Liu, Shuochen Liu, Yuhong Zhang, and Xuegang Hu. 2022. APGKT: Exploiting associative path on skills graph for knowledge tracing. In *Pacific Rim International Conference on Artificial Intelligence*. Springer, 353–365.

[44] Jing Zhang, Bowen Hao, Bo Chen, Cuiping Li, Hong Chen, and Jimeng Sun. 2019. Hierarchical reinforcement learning for course recommendation in MOOCs. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 435–442.

[45] Lei Zhang, Wuji Zhang, Likang Wu, Ming He, and Hongke Zhao. 2023. SHGCN: Socially enhanced heterogeneous graph convolutional network for multi-behavior prediction. *ACM Transactions on the Web* 18, 1 (2023), 1–27.

[46] Guojing Zhou, Hamoon Azizsoltani, Markel Sanz Ausin, Tiffany Barnes, and Min Chi. 2019. Hierarchical reinforcement learning for pedagogical policy induction. In *Artificial Intelligence in Education: 20th International Conference, AIED 2019, Chicago, IL, USA, June 25-29, 2019, Proceedings, Part I 20*. Springer, 544–556.

[47] Guojing Zhou, Hamoon Azizsoltani, Markel Sanz Ausin, Tiffany Barnes, and Min Chi. 2022. Leveraging granularity: Hierarchical reinforcement learning for pedagogical policy induction. *International journal of artificial intelligence in education* 32, 2 (2022), 454–500.

[48] Yuwen Zhou, Changqin Huang, Qintai Hu, Jia Zhu, and Yong Tang. 2018. Personalized learning full-path recommendation model based on LSTM neural networks. *Information sciences* 444 (2018), 135–152.
[49] Haiping Zhu, Feng Tian, Ke Wu, Nazaraf Shah, Yan Chen, Yifu Ni, Xinhui Zhang, Kuo-Ming Chao, and Qinghua Zheng. 2018. A multi-constraint learning path recommendation algorithm based on knowledge map. *Knowledge-Based Systems* 143 (2018), 102–114.

A PSEUDO-CODE FOR DHRL

The pseudo-code of the DHRL module is presented in Algorithm 1. The details are presented in Section 5. First, initialize the simulation environment and set the learning path $\mathcal{P} = \emptyset$. During the learning process, the L-Agent recommends a learning item c_i. Then its initial practice difficulty d_i and practice tolerance τ_i are passed to the P-Agent. The P-Agent then recommends practice items $e_{t,i}$ for this learning item step by step and records the practice count p_i. Feedback from each practice item is used to update the student's knowledge state. Once practice ends (upon reaching the mastery threshold or practice tolerance), the practice count p_i and updated knowledge state h_i are sent back to the L-Agent for the next recommendation. This process is repeated until the learning goals are achieved.

Algorithm 1 DHRL

Initialize simulation environment;
Initialize learning path $\mathcal{P} = \emptyset$;
while Learning goals not met **do**
  Recommend learning item c_i by L-Agent; (Section 5.2)
  Obatain d_i and τ_i according c_i ;(Section 5.4)
  Pass c_i, d_i and τ_i from L-Agent to P-Agent;
  Set current practice count p_i to 0;
  while $h_t < threshold$ and $n < \tau_i$ **do**
    if $n == 0$ **then**
      Recommend practice item $e_{t,i}$ according d_i by P-Agent; (Section 5.3)
    else
      Recommend adaptive practice item $e_{t,i}$ by P-Agent; (Section 5.3)
    end if
    Practice count $p_i += 1$;
    Add $e_{t,i}$ to learning path $\mathcal{P}$;
    Get the feedback $score_i^j$;
    Evolve and update learner's knowledge state h_t;(Section 5.1)
  end while
  Pass p_i and h_i from P-Agent to L-Agent;
end while
return $\mathcal{P}$

B PSEUDO-CODE FOR ADAPTIVE LEARNING ACTION SPACE

The pseudo-code of the adaptive learning action space is presented in Algorithm 2. In the algorithm input, G_l refers to the learning item graph, which is equivalent to the 'learning item' layer at the bottom of the hierarchical graph (HG). Additionally, the shortest path is obtained using the A* algorithm, a highly efficient and widely used algorithm. *Prerequisite(c)* represents all the predecessor neighbor nodes of c in the graph G_l. If there are no predecessor nodes, it returns empty. More details about Algorithm 2 can be found in Section 5.2.2.

Algorithm 2 Adaptive Learning Action Space

Input: C_{st}: start learning item; G_l: a learning item graph; $\mathcal{T}$: learning target; $threshold$: level of proficiency target
Output: $\forall\ d \in D_L$ has a shortest path to $\mathcal{T}$ in G_l.

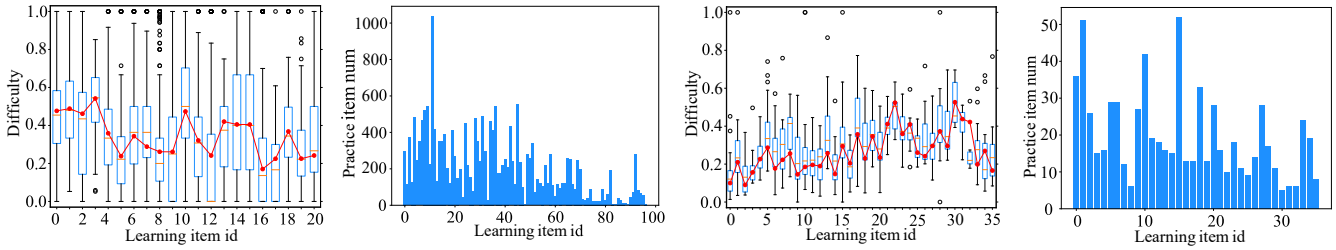
initial $D_L = \emptyset$ and $Q = \emptyset$;
obtain the shortest candidate path $P = \{c_{st}, ..., c_i, ...\mathcal{T}\}$;
add C_{st} to D_L;
$Q = P$
while $Q \neq \emptyset$ **do**
  for c in Q **do**
    add c to D_L
    Study c and get the mastery level h_c.
    if $h_c > threshold$ **then**
      $c \leftarrow Q$.pop()
    else
      add $prerequisite(c)$ to top of Q
    end if
  end for
end while
return D_L

C STATISTICS ON ITEM QUANTITY AND DIFFICULTY

The statistics of datasets are provided in Table 4. Further, as shown in Figure 6, we depict the relationship between the number of learning items and practice items in the two datasets, along with the variation in difficulty levels among different practice items. In Figure 6(a) and 6(c), the blue boxplots illustrate the difficulty distribution of all practice items associated with the learning item. The red line represents the difficulty of the learning item. As mentioned above, in real learning scenarios, a single learning item is typically assessed by multiple practice items, each with potentially different difficulty levels.

Table 4: Dataset Statistics.

Dataset	Junyi	ASSIST09
learning items	36	97
practice items	711	16,836
learners	245,511	4,092
records	25,367,573	397,235
number of edges in HG	267	683



(a) Item-diff distribution in ASSIST09 (b) Practice items in ASSIST09 (c) Item-diff distribution in Junyi (d) Practice items in Junyi

Figure 6: Statistical information on the quantity and difficulty of practice items associated with a learning item. In (a), the ASSIST09 dataset contains numerous learning items, and we randomly selected 20 of them for clarity. It is evident that in real learning scenarios, a learning item typically includes multiple practice items with varying difficulty levels.