Graph Enhanced Hierarchical Reinforcement Learning for Goal-oriented Learning Path Recommendation

Qingyao Li
ly890306@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Wei Xia
xiawei24@huawei.com
Huawei Noah's Ark Lab
Shenzhen, China

Li'ang Yin
yinla@apex.sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Jian Shen
r_ocky@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Renting Rui
ruirenting@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Weinan Zhang
wnzhang@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Xianyu Chen
xianyujun@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Ruiming Tang
tangruiming@huawei.com
Huawei Noah's Ark Lab
Shenzhen, China

Yong Yu*
yyu@apex.sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

ABSTRACT

Goal-oriented Learning path recommendation aims to recommend learning items (concepts or exercises) step-by-step to a learner to promote the mastery level of her specific learning goals. By formulating this task as a Markov decision process, reinforcement learning (RL) methods have demonstrated great power. Although extensive research efforts have been made, previous methods still fail to recommend effective goal-oriented paths due to the underutilizing of goals. Specifically, it is mainly reflected in two aspects: (1)The lack of goal planning. When learners have multiple goals with different difficulties, the previous methods can't fully utilize the difficulties and dependencies between goal learning items to plan the sequence of achieving these goals, making the path chaotic and inefficient; (2)The lack of efficiency in goal achieving. When pursuing a single goal, the path may contain learning items unrelated to the goal, which makes realizing a certain goal inefficient. To address these challenges, we present a novel Graph Enhanced Hierarchical Reinforcement Learning (GEHRL) framework for goal-oriented learning path recommendation. The framework divides learning path recommendation into two parts: sub-goal selection(planning) and sub-goal achieving(learning item recommendation). Specifically, we employ a high-level agent as a sub-goal selector to select sub-goals for the low-level agent to achieve. The low-level agent in the framework is to recommend learning items to the learner. To make the path only contain goal-related learning items to improve the efficiency of achieving the goal, we develop a graph-based candidate selector to constrain the action space of the low-level agent based on the sub-goal and knowledge graph. We also develop test-based internal reward for low-level training so that the sparsity problem of external reward can be alleviated. Extensive experiments on three different simulators demonstrate our framework achieves state-of-the-art performance.

CCS CONCEPTS

• **Applied computing** → **E-learning**.

KEYWORDS

Learning Path Recommendation; Hierarchical Reinforcement Learning; Online Education

ACM Reference Format:
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 (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3583780.3614897

1 INTRODUCTION

In online education, learners with different learning goals and knowledge background should be recommended personalized learning paths so that learners can achieve their learning goals effectively. Therefore learning path recommendation is one of the most important topics in online education [33, 38].

Several recommendation methods have been suggested to offer customised learning paths based on learner traits [10, 22, 41]. Typical solutions discover similar learners and offer similar learning paths to the target learner [10, 28]. On the other hand, since learning path recommendation is a sequential decision making problem, it is also intuitive to adopt advanced Reinforcement Learning (RL) methods to formulate the problem as a Markov Decision Process (MDP) [15, 19, 22].

*Corresponding author.

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom

ACM ISBN 979-8-4007-0124-5/23/10…$15.00
https://doi.org/10.1145/3583780.3614897

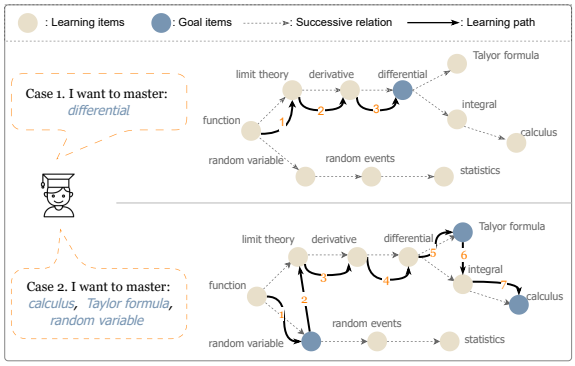


Figure 1: Illustration of a goal-oriented learning path

A learner usually engages in learning activities to complete one or more learning goals, as Figure 1 shows. To recommend an effective learning path, considering learners' learning goals is important. Unfortunately, this common learning scene has rarely been studied. This paper focuses on the goal-oriented learning path sequential recommendation problem, which recommends learning paths step-by-step to help learners achieve their goals. In this scenario, goals play a key role in deciding the path. The most representative work for this problem is proposed by Liu et al. [22], which motivates goal-oriented learning path recommendation by using a learner's promotion of learning goals as reward to train an RL recommender. The proposed CSEAL model utilizes cognitive structure to help the actor-critic algorithm [18] generate learning paths, achieving appreciated performance compared with other methods.

However, there still exists the problem of under-utilizing learners' learning goals, reflected in two aspects: (1) The lack of goal planning. A learner has multiple learning goals with varying levels of difficulty and dependencies, but the previous methods did not make reasonable plans for realizing goals while they only input the one-hot encoding of learning goals, and the agent needs to explore in a large search space to realize the goals' effect. For example, suppose the learner shown in the lower part of Figure 1 has three goals *random variable*, *Taylor formula*, and *calculus*, where *calculus* is much more difficult than *random variable* and *Taylor formula* to achieve. Here *Taylor formula* is a prerequisite item for *calculus*, so achieving *Taylor formula* ahead is helpful for mastering *calculus*. Therefore, an appropriate learning order is to achieve *random variable* and *Taylor formula* first, then to learn *calculus* so that the learner can make maximum progress in a limited time. (2)Inadequate efficiency in goal achieving, due to that the path to achieving a specific goal may contain learning items irrelevant to the goal. From the view of a learner, she wants to achieve her goals with minimal effort, and the recommended learning path should contain goal-related learning items only. As shown in the upper part of Figure 1, for a learner who wants to acquire $differential$, an efficient learning path contains $function$, $limittheory$, $derivative$, and $differential$. There is no need to put $random\ variable$ or $statistics$ in the path.

In light of the above challenges, we divide goal-oriented learning path recommendation into two parts: sub-goal selection and sub-goal achieving, where sub-goal selection aims to plan the realization sequence of multiple goals or disassemble a difficult goal into several simple sub-goals, and sub-goal achieving is to recommend learning paths to achieve the sub-goals. To make them work collaboratively, we propose Graph Enhanced Hierarchical Reinforcement Learning framework (GEHRL), which contains two agents (a high-level agent and a low-level agent) playing the roles of sub-goal selection and sub-goal achieving. 1) For sub-goal selection, the high-level agent chooses a sub-goal from all the learning items in each sub-session. When the sub-goal is one of the learners' learning goals, the high-level agent plays a role in arranging the goals' sequence (e.g., from easy to difficult) to achieve all goals more efficiently, or set the simpler prerequisite items of one hard goal as sub-goals. 2) For sub-goal achieving, the low-level agent is built to recommend learning items to the learner step-by-step. To make the path to achieve a specific goal efficiently, we develop a graph-based candidate selector to constrain the low-level agent's action. This candidate selector ensures the path for achieving a goal does not contain unrelated learning items so the learner can achieve the goal more effectively. In addition, two kinds of rewards are needed to train two agents. The environment will give an *external reward* evaluating the path's effectiveness which is used to train the high-level agent. While we need to design an *internal reward* to motivate the low-level agent to achieve each sub-goal. We design a test-based internal reward which recommend the sub-goal at the end of each sub-session to get the learner's feedback as the *internal reward*. In this way, the low-level agent receive a trustworthy signal of whether the sub-goal is achieved.

In summary, the high-level agent plans the realization of multiple goals according to the relationship between the goals, and the low-level agent is responsible for achieving each goal, with a graph-based candidate selector to make the path contain goal-related learning items only.

Our main contributions are summarized as follows:

- We analyze the challenge of the goal-oriented learning path recommendation problem and divide the task into sub-goal selection(planning) and sub-goal achieving, which is a novel perspective. We propose a Graph Enhanced Hierarchical Reinforcement Learning (GEHRL) framework for learning path recommendation, which contains a high-level policy for sub-goal selection and a low-level policy for item recommendation. To the best of our knowledge, this is the first attempt to introduce hierarchical reinforcement learning as recommendation model in learning path recommendation.
- We design a graph-based candidate selection module to select the goal-related learning items to make the path contain goal-related learning items only. In addition, we design a test-based internal reward assignment algorithm to give an accurate signal for goal-achieving.
- We validate our model in three simulators based on two benchmark datasets. Our framework consistently achieves state-of-the-art performance in experiments on three simulators.

2 RELATED WORK

2.1 Learning Path Recommendation

Learning path recommendation is an essential task in online education. Different methods have been proposed to solve this task in the past few years. In some methods, researchers try to use traditional

recommendation algorithms [2, 9, 10, 28]. For example, Elshani et al. [10] used genetic algorithms to generate and select the learning path with the best fitness value calculated by a fitness function. Traditional methods may reach considerable performance under strong assumptions in the environments with a small amount of data, but they do not comprehensively consider the long-term and short-term learning effects while generating paths. On the other hand, deep learning-based methods are studied [36, 40, 41]. Zhou et al. [41] proposed a personalized learning path recommendation model, which makes use of clustering algorithms and LSTM (long short term memory) [14] neural network to provide useful guidance to learners based on their learning process or visited materials.

Another line of promising solutions is reinforcement learning-based, since learning path recommendation can be formulated as a sequential decision-making problem. For example, Kubotani et al. [19] built an inner model to simulate the learner to train the agent with a smaller number of interactions. Liu et al. [22] defined the goal-oriented learning path recommendation problem, which use learners' promotion on learning goals as reward to train a actor-critic framework as a recommender, and built a cognitive navigation module to guide the process, so the paths are logical sequences. Despite their success, these methods cannot offer effective goal-oriented paths because goals are not used evidently to guide learning path recommendations. This drawback force earlier methods to search a large search space to realize the learning goals' guidance role and make the recommended paths contain learning items unrelated to the goal, which is inefficient for achieving learning goals.

2.2 Hierarchical Reinforcement Learning (HRL) in Recommendation

Hierarchical reinforcement learning (HRL) [32] is a branch of Reinforcement Learning (RL) that introduces temporally abstract actions. In this paper, we use the typical goal-oriented HRL framework proposed by Kulkarni et al. [20], in which a high-level agent gives sub-goals for a low-level agent to achieve. Recently, more and more works begin to utilize HRL in recommendation to capture users' preferences on different levels. Zhao et al. [39] develop multi-goals abstraction based HRL for recommendation in which the high-level agent generates multiple goals to guide the low-level agent in different sub-periods. They design an appreciated reward assignment mechanism to coordinate different goals in a consistent direction. Xie et al. [35] use HRL for integrated recommendation. They take low-level policy as a channel selector and high-level policy as an item recommender to capture user preferences on both item and channel levels.

In the educational data mining area, Zhang et al. [37] and Lin et al. [21] try to use HRL for course recommendation. However, it does not contradict our statement on the first introduction of HRL as the recommendation model in learning path recommendation. Our work differs from them in two significant ways: 1) The course recommendation recommends a single item rather than a sequential path, and without learner feedback at each step. 2) They employ HRL as an auxiliary module to clean data for the recommendation model, while HRL is the recommendation model in our work, and we use knowledge graph to enhance the representation of each item and build a candidate selector to eliminate learning items unrelated to the goal.

3 PROBLEM FORMULATION

This paper aims to solve the session-based goal-oriented learning path recommendation problem. For each learner, our job is to recommend an ordered learning item sequence step-by-step from the item set $\mathcal{I} = \{p_1, p_2, ..., p_K\}$ to maximize the learner's mastery of learning goals. The learner's learning goals are denoted as $\mathcal{G} = \{g_1, g_2, ...\}$, which is a subset of the learning item set $\mathcal{G} \subset \mathcal{I}$.

The process is shown in Figure 2. At the beginning of each session, a learner takes a test on her learning goals and gets an initial score E_{start}. Then at each step of the learning session, the learner receives an item p to learn and after being tested a feedback $score \in \{0, 1\}$ is given, where 1 stands for that the learner masters the item. Each item-score pair constitutes a response recorded in the historical learning record, denoted as $\mathcal{H} = \{(p, score)\}$. Similarly, at each subsequent step, a new learning response $(p, score)$ is added to the historical record, i.e., $\mathcal{H}t + 1 = \mathcal{H}t \cup (p_{t+1}, score_{t+1})$. Finally, a learning path with length M is generated step-by-step as $\mathcal{P} = (p_1, p_2, ..., p_M)$. In our framework, each session is divided into sub-sessions of length N, where each sub-session is dedicated to achieving one sub-goal. Typically, $M = k * N$ where k represents the number of sub-sessions. The value of M is determined by the environment, while N is set by us. After completing the entire learning path, a final test is taken on the learning goals to obtain a final score E_{end}. It is expected that the proficiency improves through the learning items in the path. Our goal is to maximize the difference between the final score (E_{end}) and the initial score (E_{start}) by providing an effective learning path.

The metric of evaluating a path for a learner is given by the promotion on the learning goals [22]:

$$E_{\mathcal{P}} = \frac{E_{end} - E_{start}}{E_{sup} - Estart} \tag{1}$$

where E_{sup} represents full marks on the learning goals which equals to the number of learning goals.

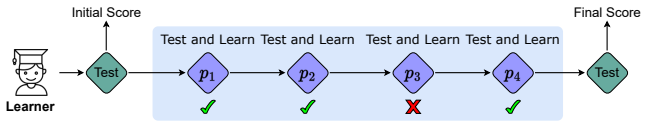


Figure 2: The Flowchart of the Learning Process

4 METHODOLOGY

4.1 Framework Overview

Figure 3 shows the overall architecture of our framework, which mainly consists of three parts: a high-level recommendation agent (HRA) as a sub-goal selector, a low-level recommendation agent (LRA) as a learning item recommender, and a graph-based candidate selector (GCS). The high-level agent gives a sub-goal at the beginning of every sub-session for the low-level agent to achieve. The high-level agent collects fewer transitions (training data) than the low-level agent in one session's interactions, so make them coordinate the training rhythm with each other, we choose the

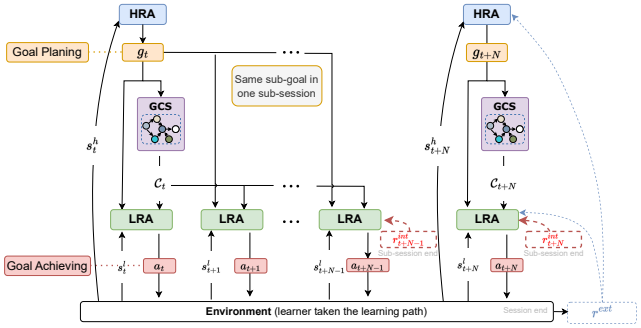


Figure 3: Framework Overview

vanilla Actor-Critic to model the low-level agent and PPO [31] (an advanced actor-critic algorithm) as the high-level agent. We shall compare different combinations of them in the experiments. The candidate selector is an appreciated algorithm to constrain the action space for the low-level agent based on the sub-goal and knowledge graph. We define the key notions as follows:

- **High-level state** s^h: s^h contains learner historical learning records $\mathcal{H}$ and the learner's learning goals $\mathcal{G}$.
- **High-level action** g: g is to choose a goal item from the learning item set. $g \in \mathcal{I}$ (any learning item can be selected as a sub-goal).
- **Low-level state** s^l: s^l contains information of learner historical learning records $\mathcal{H}$, learning goals $\mathcal{G}$ and the sub-goal g selected by the high-level agent.
- **Low-level action** a: a is to recommend a learning item for the learner to test and learn, $a \in \mathcal{I}$.
- **Internal reward** r^{int}: r^{int} is a scale to evaluate the learner's mastery level of the sub-goal g, which is only given at the last step of each sub-session, elaborated in Section 4.5.
- **External reward** r^{ext}: r^{ext} is a scale to evaluate the learner's promotion on her learning goals, which is only given at the last step of the whole session, elaborated in Section 4.5.
- **Discount factor** γ: $\gamma \in [0, 1]$. It gives a discount on the importance of future reward when calculating the cumulative reward.
- **High-level Transition Function** $p(s^h_{t+1}|s^h_t)$: As the time-varying part of the high-level state is the historical learning records $\mathcal{H}$, then once a new item is recommended to the learner and the answer is given, the state transition is record extension.
- **Low-level Transition function** $p(s^l_{t+1}|s^l_t)$: The low level state transition is also record extension. If t is integer multiple of N, the state transition contains the sub-goal change.

During the recommendation process, a session of length M is divided into multiple sub-sessions of N steps, where M is equal to k times the number of sub-sessions ($M = k * N$). As depicted in Figure 3, the high-level agent initiates each sub-session by providing a sub-goal, while the low-level agent recommends learning items sequentially to achieve this sub-goal within the sub-session.

For the high-level agent, the learner's learning records $\mathcal{H}_t$ and learning goals $\mathcal{G}$ are encoded into a vector to form as the high-level state s^h_t using a shared encoding module. Utilizing s^h_t, the high-level agent generates a sub-goal g_t to guide the low-level agent throughout the N-length sub-session. On the other hand, the low-level agent's state, denoted as s^l_t, is composed of two components. The first component is the same vector as the high-level state, encoding the learner's learning history and goals. The second component is the graph embedding representation of the sub-goal g_t. Based on s^l_t, the low-level agent recommends a learning item a^t to the learner.

At each time step of the sub-session, the internal reward is 0 until the end of the sub-session given by a test-based internal reward algorithm to evaluate the learner's mastery of the sub-goal. Similarly, the external reward is also 0 during the session until the end as an evaluation of the learner's promotion of her learning goals. The low-level agent is trained by the combination of internal reward and external reward so that it can recommend items considering both sub-goal and learning goals. The high-level agent is trained only on external reward considering that its function is to select sub-goals appropriately so that the learner can reach her learning goals faster and better. The technical details of reward mechanism will be elaborated in Section 4.5.

It should be noticed that the action space of the high-level and low-level agents is the same, but the meaning of their actions differs. The high-level agent gives an learning item representing a concept to acquire, whereas the low-level agent recommends an item reflecting a specific learning resource (e.g., exercise) related to the concept. In this paper, due to our recommendation being scaled to the level of knowledge points, we abstract all of both of them as learning items.

4.2 High-level Agent

4.2.1 High-level State Encoder. High-level state s^h_t contains information of learner historical learning records $\mathcal{H}_t = \{ \{p_1, score_1\}, \{p_2, score_2\}, ...\{p_{t-1}, score_{t-1}\} \}$ and learning goals $\mathcal{G} = \{g_0, g_1...\}$. We encode $\mathcal{H}_t$ with GRU [7] for that it has similar strength with LSTM but with simpler structure. It receives the one-hot encoded $\mathcal{H}_t$ as input and outputs the final hidden state $h^{\mathcal{H}}_t$. Then it will concatenate with the learning goals' multi-hot representation $h^{\mathcal{G}} \in \{0, 1\}^K$ where the learning goals' indexes are set 1 and others 0. The final high-level state is encoded as:

$$s^h_t = Concat(h^{\mathcal{H}}_t, h^{\mathcal{G}}_t) \tag{2}$$

4.2.2 High-level Agent Model. The objective of the high-level agent is to provide sub-goals to the low-level agent for accomplishment. We opt to employ the Proximal Policy Optimization (PPO) as the model for the high-level agent due to its recognition as one of the top-performing advanced reinforcement learning models. [31]. It contains a policy network (actor) $\pi^h(g_t|s^h_t; \theta^h)$ that outputs a action probability distribution under current state, and a value network (critic) $V^h(s^h_t; \phi^h)$ estimating the return of each state, where θ and ϕ are the corresponding network's parameters.

For the actor, we feed the s^h into a fully connected layer to output the probability distribution the learning items, and the sub-goal is sampled from it:

$$g_t \sim \pi^h(g_t|s^h_t; \theta^h) = Softmax(FC(s^h_t)) \tag{3}$$

where FC is the fully connected layer.

For the critic, it receives s^h and gives the estimate return of the state:

$$V^h(s_t^h;\phi^h) = FC(s_t^h) \tag{4}$$

The critic is trained based on the classical mean squared loss (MSE):

$$L(\phi^h) = \mathbb{E}(||\sum_{i=0}^{T-t}\gamma^i r_{t+i}^h - V^h(s_t^h;\phi^h)||)^2 \tag{5}$$

For the actor training, we first calculate a advantage value at each time step:

$$A_t^h = -V^h(s_t^h;\phi^h) + r_t^h + \gamma r_{t+1}^h + \cdots + \gamma^{T-t+1} r_{T-1}^h + \gamma^{T-t} V^h(s_T^H;\phi^h) \tag{6}$$

Then in the actor's training process, it divides one episode's data into several epochs to train. We denote π^{θ_k} as the interaction actor for the k-th epoch. The actor is trained based on the PPO-clip loss function:

$$L(\theta^h) = -\mathbb{E}_{g\sim\pi^{\theta_k}}[min(\frac{\pi^{\theta_h}}{\pi^{\theta_k}}A^{\pi\theta^k}, clip(\frac{\pi^{\theta_h}}{\pi^{\theta_k}}, 1-\epsilon, 1+\epsilon)A^{\pi\theta^k})] \tag{7}$$

where the $clip(x,l,h) = max(min(x,h),l)$ which restricted x in $[l,h]$. In this way, the training could be more stable and effective [31].

4.3 Low-level Agent

4.3.1 Low-level State Encoder. The low-level state encoder share the same GRU encoder with the high-level agent to encode the learner's learning history and learning goals. In addition, the low-level state s^l also contains the graph embedding of the sub-goal h_t^g.

$$s_t^l = Concat(h_t^{\mathcal{H}}, h_t^{\mathcal{G}}, h_t^g) \tag{8}$$

4.3.2 Low-level Agent Model. In our HRL framework, the high-level agent gets a transition every N steps(every sub-session): (s_t^h, g_t, s_{t+N}^h), however, the low-level agent gets a transition every step:(s_t^l, a_t, s_{t+1}^l). So low-level agent can train on more data than the high-level agent after the same interactions. To avoid the low-level policy to get ahead of the high-level policy too much, we choose to use vanilla actor-critic [18] framework as the low-level agent. Another reason of this setting is that for the high-level agent, the low-level agent is part of the environment since it will affect the state transition the high-level agent faces. From experiments, we find this setting shows advantage.

Actor-Critic trains critic with the same MSE loss as PPO, and calculates advantage function A_t^l the same as A_t^h only using low-level's reward. The main difference between vanilla actor-critic and PPO is in the actor's training algorithm. The actor-critic trains based the simple policy gradient:

$$L(\theta^l) = -\mathbb{E}_t[log(\pi^{\theta^l}(a_t|s_t^l) * A_t^l(s_t^l, a_t))] \tag{9}$$

4.4 Graph-based Candidate Selector

If the action space of the low-level agent is the whole item set, The path may contain learning items unrelated to the sub-goal, and the search space is very large. For example, assume we set the sub-session's length N=5, and the item set size K=10. Then for each sub-goal, the low-level agent's sub-session path search space sizes 50. Most of these paths are unreasonable or inefficient. In the human learning process, we only learn related knowledge points when we want to acquire a certain learning goal. This "related" relationship could be discovered from the knowledge graph. Due to this intuition, we develop a graph-based candidate selector to constrain the action space for the low-level agent so that the recommended goal-centered learning paths to achieve the sub-goal faster and better. This selector plays two roles. First, from the learning path recommendation's perspective, it makes the path totally goal-centered to achieve the goal. Second, from RL's perspective, it reduces the exploration space for the agent. We provide two selection schemes for this candidate selector.

4.4.1 Sub-tree based candidate selection. The knowledge graph is a digraph with learning items as nodes. An edge (i,j) in the graph means that item i is the prerequisite item for item j. From this graph, we can extract a sub-goal's prerequisite nodes as a tree rooted by the sub-goal and take the nodes in this tree as the action candidates of the low-level agent.

$$C^{subtree} = prerequisite(subgoal) \tag{10}$$

where $C^{subtree}$ is the sub-tree based candidate set; $prerequisite(x)$ means all the prerequisite nodes of node x.

4.4.2 Graph embedding based candidate selection. Graph embedding [23, 34] aims to learn a scale vector from the graph to represent each node (item) so that the structure information can be encoded. We extract related items which are closer to the sub-goal in the embedding space.

$$C^{emb} = topC_{item\in\mathcal{I}}(-||emb(subgoal) - emb(item)||_2^2) \tag{11}$$

where C^{emb} is the graph embedding based candidate set; $emb(x)$ means the graph embedding of item x; $topC_x(f(x))$ means extracting the top C with biggest value of $f(x)$.

Here, we need a graph embedding model to generate the representation of the nodes as to capture the nodes that are close to the sub-goal node and similar to it. We choose to use node2vec [11] for it considers the "homophily" and "structural equivalence" of the graph and can effectively capture the structure and distance of nodes in the graph. Any graph embedding models could be used here and we compare representative models in the experiments.

4.5 Reward Mechanism

4.5.1 External reward. Though a signal of whether the learner answering correctly or not is received after recommending a learning item, it is not appropriate to directly set this signal as the external reward because our purpose is to maximize the promotion of the learner's learning goals. The promotion is obtained at the end of the learning session, so following [22], the external reward is set to be the learner's promotion on learning goals $r_{session}^{ext} = E_{\mathcal{P}}$ in Equation 1 which is given only when the session is over. For each time step, the reward is set by:

$$r_t^{ext} = \begin{cases} r_{session}^{ext}, & \text{if } t \text{ is the last time step} \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

And the high level reward r_t^h is set to be the external reward.

$$r_t^h = r_t^{ext} \tag{13}$$

4.5.2 Test-based internal reward. The internal reward r^{int} evaluates the learner's mastery level on the sub-goal item. We utilize the Deep Knowledge Tracing model (DKT) [30] to predict the learner's mastery of all the learning items based on her learning records. However, the DKT can only be trained on the RL's interaction records with learners, which would not be so stable and accurate, so setting the internal reward as the DKT's prediction value directly could be insufficient to guide the low-level agent's training, which is also confirmed in experiments.

To get an accurate signal of whether the learner has acquired the sub-goal, we develop a test-based internal reward mechanism. The idea is that although we cannot get the learner's mastery level, we can get the feedback $score_t$ of whether she answers the learning item we recommend correctly at each step. Thus we set the sub-goal as the recommending item at the end of each sub-session, so we can get the feedback signal and know if she masters the sub-goal.

Additionally, we utilize DKT's prediction to estimate the learner's mastery level, but only when the prediction is greater than a threshold δ_{thre} that we believe the learner has achieved the sub-goal. (DKT's output is a scale in [0, 1] representing the mastery level). We set δ_{thre} larger than 0.5 for that we only want to use DKT to judge the learner's mastery when the prediction is certain.

$$r_t^{int} = \begin{cases} 1, & DKT[subgoal] > \delta_{thre} \text{ or } score_{subend} = 1 \\ 0, & otherwise \end{cases} \tag{14}$$

This test-based internal reward gives us a reliable signal on whether the learner has mastered the sub-goal. However, there might be more than one path that could lead the learner to achieve the sub-goal. The low-level agent should choose the one that could also provide more promotion on the learning goals. Therefore, different from a typical hierarchical reinforcement learning approach training the low-level agent with internal reward only, we set the low-level agent's reward as a combination of the internal reward that indicates how well the sub-goal has been mastered, and the external reward that indicates how well learning goals have been promoted. In this way, the low-level agent can achieve the sub-goal and maximize the learner's mastery of learning goals at the same time.

$$r_t^l = \alpha r_t^{int} + \beta r_t^{ext} \tag{15}$$

where α is the weight for internal reward and β is the weight for external reward, r_t^l is the low level reward at timestep t.

5 EXPERIMENTS

In this section, we first introduce the experimental setup. Then we compare our model's performance with several baselines and discuss the promotion and effects of our model.

5.1 Datasets and Simulators

5.1.1 Datasets. Our experiments are based on two datasets: *Junyi*[1] and *ASSITments2015*[2]. They both provide learners' exercising logs. *Junyi Academy Dataset* also provide a prerequisite graph [5] in which an edge $e_{i,j}$ means item i is prerequisite to j. However, the knowledge graph is not provided in *ASSISTments2015* so we build a transition graph [27] as an estimation of the prerequisite graph. We provide the statistics of these two datasets in Table 1.

5.1.2 Simulators. To evaluate different methods' recommending effects, we use the similar experimental environment built in Liu et al. [22], where two kinds of simulators are developed: Knowledge Structure based Simulator (KSS) and Knowledge Evolution based Simulator (KES). Specifically, we develop three simulators: KSS, KES-junyi and KES-ASSIST. KES-junyi and KES-ASSIST are both KES and they are based on datasets *Junyi* and *ASSITments2015* respectively.

KSS is a rule-based simulator containing 10 learning items where the learner's performance on the knowledge item is measured based on Item Response Theory (IRT) [24, 25]. KES is a data-based simulator with a Deep Knowledge Tracing model (DKT) [30] inside to simulate the learner's change of knowledge state and the performance on learning items. Since DKT is trained on a specific dataset, the initial logs from the dataset would be used to simulate the learner's initial state.

It should be noted that in the original implementation of KSS, each learner's learning behavior is subject to a strict constraint: the item learned at time step t must be close to the previous item at time step $t-1$. (Following the cognitive navigation the author proposed). We alleviate the constraint by imposing a discount on the learner's promotion when the learner learns a knowledge graph item that far from from the previous one. This treatment is more appropriate for the situation than the hard constraint. Furthermore, we use the entire dataset to train the simulator's DKT and only 50% of it to train our model's DKT (for calculating the internal reward) because the DKT cannot predict the learner's mastery so accurately in real situation.

5.2 Compared Methods

To demonstrate the effectiveness of our approach, we compare it with the following methods:

- KNN [6]: A traditional classification algorithm that each sample can be represented by its nearest K adjacent values. In our environment, we use the cosine similarity between the learning paths to measure the similarity between two learners and recommend similar items.
- GRU4Rec: Proposed by Hidasi et al. [13] for session-based recommendation. The session is one-hot encoded and embeded before processed by a GRU network and return the probability of every item being recommended.
- DQN: One of the basic reinforcement learning algorithms [26]. It estimates the value of each action (learning item in our case) at each state by a neural network and recommend with the one with highest value.
- SAC: Soft Actor-Critic [12]. An advanced off-policy maximum entropy reinforcement learning algorithm.
- Actor-Critic: Use a GRU encoder and vanilla actor-critic [18] as recommender.

[1] https://www.kaggle.com/datasets/junyiacademy/learning-activity-public-dataset-by-junyi-academy
[2] https://sites.google.com/site/assistmentsdata/datasets/2015-assistments-skill-builder-data

Table 1: Dataset Statistics

Dataset	Junyi	ASSISTments2015
#exercises	835	100
#learners	525,061	69,675
#records	21,460,249	2,420,200
attempts per question	16.42	5.49
positive label rate	54.38%	73.17%

- CB: Contextual Bandits for learning path recommendation [15]. An learning path recommendation method which model the process as a contextual bandit as to solve with corresponding method.
- RLTutor: An Model-based RL method from [19], which presents an adaptive tutoring system that uses DAS3H [8] to model the behavior of virtual students and use RL agent to recommend learning items.
- CSEAL: Proposed by Liu et al. [22] which use a DKT as encoder and vanilla Actor-Critic as recommender with a cognitive navigation for guidance.
- (ours) GEHRL-ST: The framework we proposed with the sub-tree based candidate selector.
- (ours) GEHRL-EB: The framework we proposed with the graph embedding based candidate selector.

We evaluate these methods with the average promotion of the mastery on the learning goals after path recommendation, which is given by the converge reward from the simulators.

5.3 Implementation Details

We use MindSpore [1], gym [4], and the simulator code from [22] to implement our proposed learning path recommendation framework. Following [22], we set the embedding layer's dimension in DKT and GRU encoder to 15 in KSS , 600 in KES-junyi and 64 in KES-ASSIST. The hidden dimension in GRU is 20, 512, 128 in KSS, KES-junyi and KES-ASSIST, respectively. We use multiple-layer perceptron to implement the policy network and value network and set the two layers' hidden sizes as 128, 32 in KSS and KES-ASSIST, 512, 256 in KES-junyi. Both high-level agent and low-level agent use this setting. All the experiments run with 3 random seeds:1, 5, 10. We use Adam [16] as our optimizer and the learning rate is set to be 5×10^{-5}. And we set the reward weights α and β to be 1.0 in all three environments. For the baselines DQN and SAC, we set the learning rate to 3×10^{-4} because we find it make the models perform better. The GRU encoder's setting is the same for all the RL models. δ_{thre} is set to be 0.9 in all three simulators. The source code for our proposed GEHRL are available[3].

5.4 Overall Performance Comparison

Table 2 reports the overall performance of all methods in three simulators. Table 2 demonstrates that:

- Our proposed GEHRL-EB outperforms all baselines, and GEHRL-ST defeats other baselines in many situations too. The results demonstrate the advantages of our methods.
- Reinforcement learning methods generally outperforms non-RL methods in 20-step learning path recommendation, which gets benefit from that RL methods can give consideration to both long-term benefits.
- The best performances are achieved by CSEAL or GEHRL in all cases, demonstrating that constraining the action space in a reasonable way could help RL methods get better solution.

The results show that introducing hierarchical reinforcement learning into learning path recommendation is effective. Furthermore, the proposed graph-based candidate selector and test-based internal reward make the framework work better. It should be clear that there are negative values in KES-junyi due to that there are 835 items in this simulator, and some items never occurred in the training data. When the DKT model in the simulator encounters these items, the prediction may be unstable, so the reward could be negative. We will conduct more detailed experiments in the following sub-sections.

5.5 Impact of Graph-based Candidate Selector

In our framework, we propose a graph-based candidate selector to constrain the action space for the low-level agent to achieve sub-goals. To prove the effectiveness of the candidate selector, we conduct an ablation study that removes the candidate selection step. Our framework's performance on different simulators are illustrated in Figure 4.

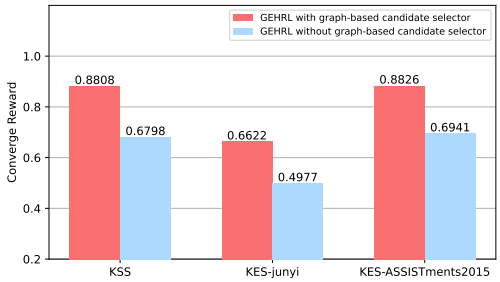


Figure 4: Ablation study for graph-based candidate selector

We can see from Figure 4 that the framework with the graph-based candidate selector has better performance. As described in Section 4.4, this selector could reduce the search space for the RL agent. To demonstrate this point, we calculate the average size of exploration space of various methods in Table 3. Our method reduces search space significantly, and RL agents could explore such a space to find a suitable policy easily.

5.6 Low-level Reward Study

To illustrate the effectiveness of our proposed low-level agent's test-based internal reward mechanism, we compare different reward settings. Specifically, we compare our setting with 1) DKT-based internal reward only, which sets the reward equal to the deep knowledge tracing model's prediction. 2) External reward only, where we remove the test-based reward and train the low-level agent with external reward only. 3) Test-based internal reward only where we train the low-level agent with test-based internal reward only. The results are demonstrated in Figure 5.

From Figure 5 we can see that test-based internal reward combined with external reward as a low-level agent's reward achieves the best result. For the internal reward, test-based reward performs

[3]Source code for model implementations: https://gitee.com/mindspore/models/tree/master/research/recommend/GEHRL

Table 2: Performance comparison for learning path recommendation methods.

		KNN	GRU4Rec	DQN	SAC	Actor-Critic	CB	RLTutor	CSEAL	GEHRL-ST	GEHRL-EB
KSS	Step = 5	0.1607	0.1792	0.2627	0.2714	0.2385	0.1355	0.2947	0.3313	0.3062	**0.3784**
	Step = 10	0.4227	0.4133	0.2701	0.3692	0.4811	0.3610	0.5108	0.5175	0.6089	**0.6241**
	Step = 20	0.3851	0.1905	0.3855	0.2665	0.5051	0.3891	0.6227	0.6833	0.8333	**0.8808**
KES-junyi	Step = 5	0.0102	-0.0974	-0.1069	-0.4647	0.2852	0.1147	-0.0797	0.2401	-0.0342	**0.2868**
	Step = 10	-0.1205	-0.1227	-0.1386	-0.2873	0.1259	0.1691	-0.1037	0.3071	0.0269	**0.3463**
	Step = 20	0.1405	0.0011	0.2105	0.2140	0.3183	0.3128	-0.1306	0.3942	0.5626	**0.6622**
KES-ASSIST15	Step = 5	0.3544	0.3592	0.4214	0.5459	0.4327	0.4986	0.5623	0.5621	0.5751	**0.5858**
	Step = 10	0.2586	0.2433	0.5230	0.4817	0.5913	0.5394	0.7069	0.6954	0.6707	**0.7508**
	Step = 20	0.1644	0.1025	0.3675	0.1543	0.7282	0.6413	0.8713	0.8015	0.8352	**0.8826**

Table 3: Various methods' average search space sizes.

	Total space	CSEAL	Ours
KSS	10^{10}	3.4×10^6	5.9×10^4
KES-junyi	835^{10}	5.3×10^{15}	5.9×10^{14}
KES-ASSIST15	100^{10}	5.5×10^{10}	1×10^{10}

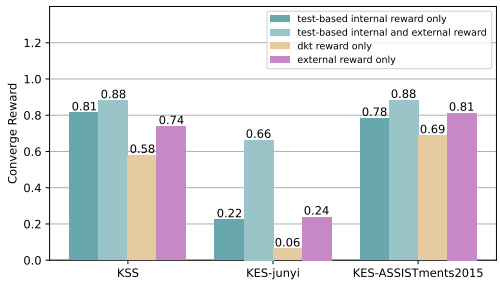


Figure 5: Compare the performances with different settings of low-level agent's reward

better than DKT-based reward, demonstrating that test-based internal reward is more trustworthy. In addition, training with only test-based internal reward or external reward could not reach an appreciable result, proving that the low-level agent should give consideration to the improvement of sub-goal and learning goals.

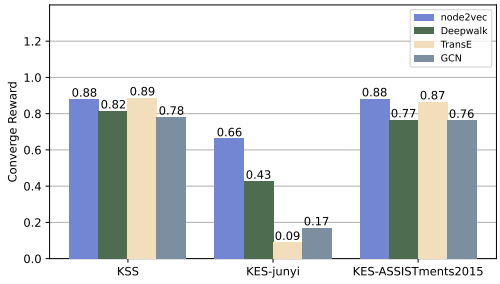
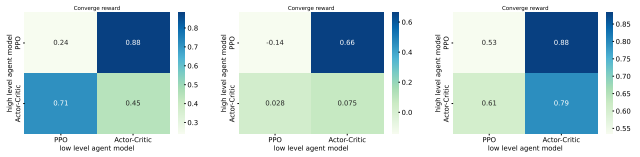


Figure 6: Compare the performances with different settings of graph embedding model.

5.7 Agent Model Study

In our setting, we employ PPO as the high-level agent model and vanilla Actor-Critic as the low-level agent. The reason is that the low-level agent can get one transition for training at each step in one session, but the high-level agent can only get one transition every sub-session (every N step). To keep the high-level and low-level agents in the same training rhythm, we set the high-level agent as a more advanced model. To show the importance of this balancing operation, we conduct experiments on different agent model combinations. The results are showed in Figure 7. The results demonstrate that an advanced high-level policy combined with a simple low-level policy could perform the best because the low-level agent could get more training samples in one session.



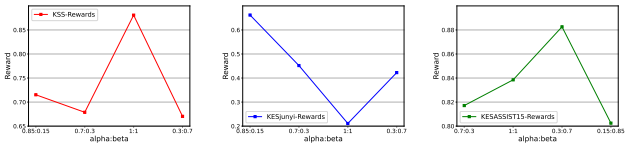
(a) KSS **(b) KES-junyi** **(c) KES-ASSIST**

Figure 7: Comparing different combinations of high-level agent and low-level agent

5.8 Graph Embedding Model Study

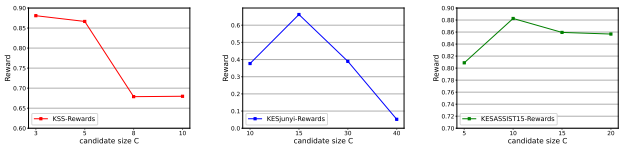
As explained in Section 4.4.2, our graph embedding-based candidate selection algorithm can be combined with any graph embedding model. We compared the effects of representative graph embedding models like node2vec [11], deepwalk [29], TransE [3], GCN [17], as shown in Figure 6.

It can be seen from Figure 6 that node2vec achieves the best overall results. In our model, the main function of graph embedding is to capture nodes close to or similar to the sub-goal nodes in the knowledge graph for use as candidate sets. A simple model like node2vec can achieve this. It considers the "homophily" and "structural equivalence" of the graph by adjusting the random walk weight, which can effectively capture the structure and distance of nodes in the graph. TransE performs poorly in KES-junyi with a large number of learning items. It is more suitable for graphs with different kinds of edges. In our graph, there is only one edge. GCN is difficult to converge to a good result because it is trained based on RL's loss and does not take the node distance information and similarity as the training target. Deepwalk works similarly to node2vec, while it doesn't consider the "homophily" and "structural equivalence" of the graph.



(a) KSS (b) KES-junyi (c) KES-ASSIST

Figure 8: Parameter sensitiveness of $\alpha : \beta$



(a) KSS (b) KES-junyi (c) KES-ASSIST

Figure 9: Parameter sensitiveness of candidate set size C.

5.9 Parameter Sensitivity

Our model has three key parameters: embedding-based candidate size C and low-level reward weights α and β. We conduct experiments with different parameter settings of them. Figure 8 shows the sensitivity of α and β. We can see that the weight distribution of the two rewards greatly impacts the performance, and different datasets have their best-fitting parameters, respectively. This is mainly due to the stability of different simulators. The more stable the simulator is, the external reward is more trustworthy and should be weighted higher. Among simulators used in the experiments, KSS is the most stable because of its rule-based simulation, and KES-junyi is the most unstable one due to the unseen items in training data which makes the reward in large variance. We can see from Table 2 that the rewards are in great variance.

Figure 9 shows how performance changes with C. We discuss a few observations here. Firstly, in KES-ASSISTments15 where the item set size is 100, the performance achieves peak when $C = 10$, while in KES-junyi where the item size is 835, the performance achieves peak when the candidates size is 15. This indicates that candidate size grows with the learning item set size, but their relationship is not linear. This is because the search space of learning paths grows exponentially with the candidate size. Second, we can see in KES-junyi and KES-ASSISTments15 that the performance does not change monotonically with candidate size, indicating that the candidate size should not be too small to exclude the best solution, nor too large to explore.

5.10 Case Study

Figure 10 shows an example from the experiments in KSS of different models recommending learning paths for the same learner with learning goals indexed 0, 3, and 7 within maximum 20 steps. Goal 0 is a basic knowledge point that most learners have acquired prior to taking the path which the agents should acquire this information during training.

Actor-Critic recommends some basic items and comes to 7 at last, which is not a reasonable path for achieving these learning goals. Therefore, it ends with goals 3 and 7 unfinished. CSEAL recommend inefficient paths containing learning items unrelated to the goals. As an example, the recommended path includes irrelevant items 2 and 8. Furthermore, the path to achieving goal 7 contains too many items with item 6, which is ineffective. The reason behind this is that CSEAL lacks a lower-level agent that can learn how to accomplish a specific goal. As a result, the path may be ineffective due to the inclusion of redundant actions. Through training, our GEHRL model acquires the understanding that goal 0 is already mastered by the majority of learners. Therefore, in the initial sub-session (which has a length of 5), the model sets the sub-goal as item 3 and accomplishes it by recommending the prerequisite items 0 and 1. Once the learner has successfully accomplished goal 3, the sub-goal is updated to the final goal 7. Our approach achieves this sub-goal through a shorter path compared to CSEAL. This is attributed to the effectiveness of our low-level agent, which possesses a better understanding of how to achieve individual goals.Finally, the learner has completed all learning goals. The effectiveness of our approach is evident in the capabilities of our high-level agent to offer sensible sub-goals and our low-level agent to suggest paths exclusively consisting of goal-related learning items.

The reason for the repetition of recommending the same item in the learning path is due to our experimental setup, where the recommended learning item is set as a knowledge concept. The purpose of this repetition is to enable the learner to engage with different resources (exercises or videos) related to that particular knowledge concept, thereby deepening their understanding and mastery of it.

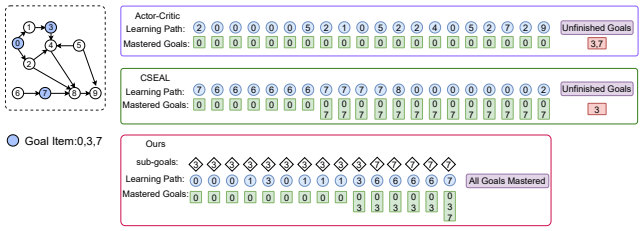


Figure 10: Different models' recommending learning paths for a learner with goal learning items 0,3,7

6 CONCLUSION

We present a graph-enhanced hierarchical reinforcement learning framework for goal-oriented learning path recommendation, including a high-level agent and a low-level agent to overcome the under-utilization of learning goals and offer efficient goal-oriented paths. To improve the framework's exploration ability, we propose a graph-based candidate selector and a test-based internal reward mechanism. Extensive experiments show the framework's usefulness in recommending learning paths to achieve learner goals efficiently.

7 ACKNOWLEDGMENTS

The work is supported by National Natural Science Foundation of China (No. 62177033). The work is also sponsored by Huawei Innovation Research Program. We gratefully acknowledge the support of MindSpore [1], CANN (Compute Architecture for Neural Networks), and Ascend AI Processor used for this research.

REFERENCES

[1] 2023. MindSpore. https://www.mindspore.cn/

[2] Cun-Ling Bian, De-Liang Wang, Shi-Yu Liu, Wei-Gang Lu, and Jun-Yu Dong. 2019. Adaptive learning path recommendation based on graph theory and an improved immune algorithm. KSII Transactions on Internet and Information Systems (TIIS) 13, 5 (2019), 2277–2298.

[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. Advances in neural information processing systems 26 (2013).

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

[5] Haw-Shiuan Chang, Hwai-Jung Hsu, and Kuan-Ta Chen. 2015. Modeling Exercise Relationships in E-Learning: A Unified Approach.. In EDM. 532–535.

[6] Haw-Shiuan Chang, Hwai-Jung Hsu, and Kuan-Ta Chen. 2015. Modeling Exercise Relationships in E-Learning: A Unified Approach. In EDM.

[7] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014).

[8] Benoit Choffin, Fabrice Popineau, Yolaine Bourda, and Jill-Jenn Vie. 2019. DAS3H: modeling student learning and forgetting for optimally scheduling distributed practice of skills. arXiv preprint arXiv:1905.06873 (2019).

[9] Pragya Dwivedi, Vibhor Kant, and Kamal K Bharadwaj. 2018. Learning path recommendation based on modified variable length genetic algorithm. Education and information technologies 23, 2 (2018), 819–836.

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

[11] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. 855–864.

[12] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In International conference on machine learning. PMLR, 1861–1870.

[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] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural computation 9, 8 (1997), 1735–1780.

[15] 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.

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

[17] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016).

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

[19] 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).

[20] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. Advances in neural information processing systems 29 (2016).

[21] Yuanguo Lin, Shibo Feng, Fan Lin, Wenhua Zeng, Yong Liu, and Pengcheng Wu. 2021. Adaptive course recommendation in MOOCs. Knowledge-Based Systems 224 (2021), 107085.

[22] 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. 627–635.

[23] Teng Long, Ryan Lowe, Jackie Chi Kit Cheung, and Doina Precup. 2016. Leveraging lexical resources for learning entity embeddings in multi-relational data. arXiv preprint arXiv:1605.05416 (2016).

[24] Frederic Lord. 1952. A theory of test scores. Psychometric monographs (1952).

[25] Frederic M Lord. 2012. Applications of item response theory to practical testing problems. Routledge.

[26] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013).

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

[28] Mehdi Niknam and Parimala Thulasiraman. 2020. LPR: A bio-inspired intelligent learning path recommendation system based on meaningful learning theory. Education and Information Technologies 25, 5 (2020), 3797–3819.

[29] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. 701–710.

[30] 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).

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

[32] Richard S Sutton, Doina Precup, and Satinder Singh. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. Artificial intelligence 112, 1-2 (1999), 181–211.

[33] Hangyu Wang, Ting Long, Liang Yin, Weinan Zhang, Wei Xia, Qichen Hong, Dingyin Xia, Ruiming Tang, and Yong Yu. 2023. GMOCAT: A Graph-Enhanced Multi-Objective Method for Computerized Adaptive Testing. In Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 2279–2289.

[34] Han Xiao, Minlie Huang, Lian Meng, and Xiaoyan Zhu. 2017. SSP: semantic space projection for knowledge graph embedding with text descriptions. In Thirty-First AAAI conference on artificial intelligence.

[35] Ruobing Xie, Shaoliang Zhang, Rui Wang, Feng Xia, and Leyu Lin. 2021. Hierarchical reinforcement learning for integrated recommendation. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35. 4521–4528.

[36] Hang Yin, Zhiyu Sun, Yanchun Sun, and Gang Huang. 2021. Automatic Learning Path Recommendation for Open Source Projects Using Deep Learning on Knowledge Graphs. In 2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC). IEEE, 824–833.

[37] 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.

[38] Qian Zhang, Jie Lu, and Guangquan Zhang. 2021. Recommender Systems in E-learning. Journal of Smart Environments and Green Computing 1, 2 (2021), 76–89.

[39] Dongyang Zhao, Liang Zhang, Bo Zhang, Lizhou Zheng, Yongjun Bao, and Weipeng Yan. 2020. Mahrl: Multi-goals abstraction based deep hierarchical reinforcement learning for recommendations. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. 871–880.

[40] Ling Zhong, Yantao Wei, Huang Yao, Wei Deng, Zhifeng Wang, and Mingwen Tong. 2020. Review of deep learning-based personalized learning recommendation. In Proceedings of the 2020 11th International conference on E-education, E-business, E-management, and E-learning. 145–149.

[41] 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.