

Exploiting Cognitive Structure for Adaptive Learning

Qi Liu¹, Shiwei Tong¹, Chuanren Liu², Hongke Zhao³, Enhong Chen^{1,*},
Haiping Ma^{4,5}, Shijin Wang^{4,5}

¹Anhui Province Key Laboratory of Big Data Analysis and Application, School of Computer Science and Technology & School of Data Science, University of Science and Technology of China, {qiliuq, cheneh}@ustc.edu.cn, tongsw@mail.ustc.edu.cn

²Business Analytics and Statistics, University of Tennessee, chuanren@xminer.org

³The College of Management and Economics, Tianjin University, hongke@tju.edu.cn

⁴iFLYTEK Research, iFLYTEK CO., LTD., ⁵State Key Laboratory of Cognitive Intelligence, {hpma,sjwang3}@iflytek.com

ABSTRACT

Adaptive learning, also known as adaptive teaching, relies on learning path recommendation, which sequentially recommends personalized learning items (e.g., lectures, exercises) to satisfy the unique needs of each learner. Although it is well known that modeling the cognitive structure including *knowledge level* of learners and *knowledge structure* (e.g., the prerequisite relations) of learning items is important for learning path recommendation, existing methods for adaptive learning often separately focus on either knowledge levels of learners or knowledge structure of learning items. To fully exploit the multifaceted cognitive structure for learning path recommendation, we propose a Cognitive Structure Enhanced framework for Adaptive Learning, named **CSEAL**. By viewing path recommendation as a Markov Decision Process and applying an actor-critic algorithm, CSEAL can sequentially identify the right learning items to different learners. Specifically, we first utilize a recurrent neural network to trace the evolving knowledge levels of learners at each learning step. Then, we design a navigation algorithm on the knowledge structure to ensure the logicity of learning paths, which reduces the search space in the decision process. Finally, the actor-critic algorithm is used to determine what to learn next and whose parameters are dynamically updated along the learning path. Extensive experiments on real-world data demonstrate the effectiveness and robustness of CSEAL.

KEYWORDS

Adaptive Learning; Knowledge Graph; Reinforcement Learning

ACM Reference Format:

Qi Liu, Shiwei Tong, Chuanren Liu, Hongke Zhao, Enhong Chen, Haiping Ma, & Shijin Wang. 2019. Exploiting Cognitive Structure for Adaptive Learning. In *The 25th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD'19), August 4–8, 2019, Anchorage, AK, USA*. ACM, NY, NY, USA, 9 pages. <https://doi.org/10.1145/3292500.3330922>

*Corresponding Author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](https://permissions.acm.org).

KDD '19, August 4–8, 2019, Anchorage, AK, USA

© 2019 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-6201-6/19/08...\$15.00

<https://doi.org/10.1145/3292500.3330922>

1 INTRODUCTION

Learning is the ladder of the progress of mankind, by which people can acquire knowledge and skills. Different from traditional learning (e.g. courses in classrooms) that presents the same material to all learners, adaptive learning aims at providing personalized learning items and paths tailored to individual learners [3]. As shown in Figure 1, adaptive learning recommends a learning path $C \rightarrow D \rightarrow B \rightarrow \dots$ for the learner who wants to learn *multiplication* by considering his or her current level of knowledge and the prerequisite relation of learning items (e.g., *two digit addition* is a prerequisite of *multiplication*). Usually, examinations, e.g., two tests in Figure 1 on D, are used to retrieve the learning effects. The personalized learning path helps the learner understand the new learning items efficiently [19, 33]. Recently, adaptive learning has become a crucial component for many applications such as on-line education systems (e.g., *KhanAcademy.org*, *junyiacademy.org*).

Research on education has shown that the cognitive structure has great impacts on adaptive learning [25, 26, 28]. The cognitive structure describes the qualitative development of knowledge and contains two parts: knowledge level of learners and knowledge structure (e.g., the prerequisite relations) of learning items. The knowledge level reflects the masteries on learning items which keeps evolving and can not be observed directly (e.g. genetic epistemology [26]), meanwhile the knowledge structure captures the cognitive relations among the learning items. However, existing methods for adaptive learning only utilize either knowledge level [33, 40] or knowledge structure [38, 41]. Although these methods have made a great success in adaptive learning, there are some limitations of them. To be specific, methods based on knowledge level without the knowledge structure may fail to resolve learning items dependency, e.g., prerequisite. The methods based on knowledge structure which ignore learners' knowledge level can not reflect the learning abilities of different learners, so that they can not precisely determine the customized learning tempo [6]. Therefore, the recommended learning paths to each learner may be less suitable and inefficient. Thus, how to systematically exploit cognitive structure including both knowledge level and knowledge structure for adaptive learning is still a challenging problem.

We summarize three challenges along this line. First, the knowledge level of learner cannot be observed directly and keeps evolving. As shown in the radar graphs of Figure 1, the learner's masteries of each learning items, i.e. the knowledge level, are continuously changing during learning but can not be observed directly. The knowledge level of learners influences the learning effectiveness

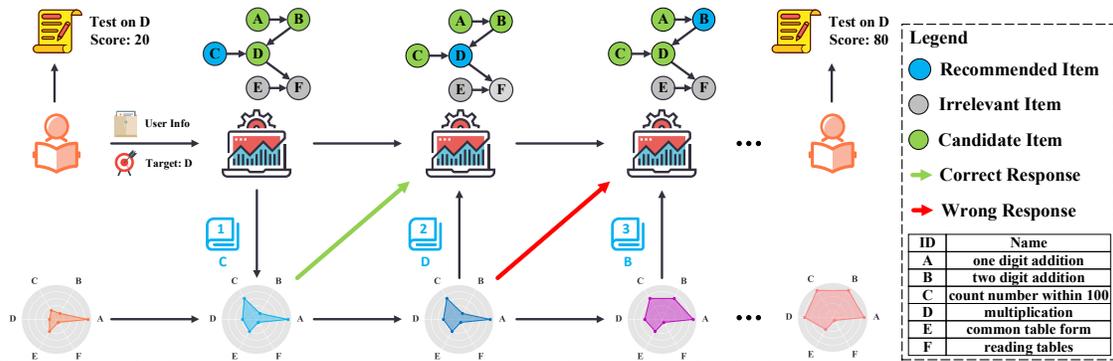


Figure 1: Illustration of Adaptive Learning. $C \rightarrow D \rightarrow B \rightarrow \dots$ is a learning path which promotes the learner’s mastery on D from 20 pt to 80 pt. User Info contains the basic information of learners such as grade, historical learning records. Target includes one or more learning items that the learner wants to master. Radar graphs in the bottom show the evolving knowledge level during learning which can not be directly observed, and the top directed acyclic graphs represent the knowledge structure.

of learning items. For example, with the poor mastery of *two digit addition*, it is difficult to learn *multiplication*. Thus, it is necessary to model the implicit evolving knowledge level. Second, the knowledge structure of learning items should be incorporated to determine logical learning paths [17]. As shown in the directed acyclic graphs of Figure 1, a learner who wants to learn the senior item D (*multiplication*) should firstly learn the prerequisites B (*two digit addition*) and C (*count number within 100*). Therefore, the learning path should be in accordance with the logicity determined by the knowledge structure of learning items. Third, a good learning path recommendation should maximize the overall gain along the whole learning path instead of only focusing on the gain in one step. As shown in Figure 1, the learning effectiveness is the promotion in examinations rather than simply the correction of one item [6].

To address the challenges, we propose a general Cognitive Structure Enhanced framework for Adaptive Learning (CSEAL). We model the sequential learning path recommendation as a Markov Decision Process (MDP). We apply reinforcement learning to gradually optimize the recommendation strategy based on cognitive structure of adaptive learning. Specifically, the Knowledge Tracing model based on Long Short-Term Memory (LSTM) network is first applied to retrieve the evolving knowledge level. Second, to prevent learning path from violating the sequential logicity being recommended, the Cognitive Navigation algorithm is designed to select a certain number of learning items, i.e., candidates, based on knowledge structure, which can also reduce the large searching space. Finally, the Actor-Critic Recommender will determine what to learn next, whose parameters are updated to improve the effectiveness of the whole recommended learning path rather than that of only one item. Extensive experiments show that CSEAL not only significantly outperforms several baselines, but also provides interpretable insights in the learning path recommendations.

2 RELATED WORK

Generally, the related work of this study can be grouped into the following three categories.

Learning Path Recommendation. The simplest way to generate learning paths is to introduce those methods aiming to solve sequence recommendation problem, e.g., collaborative filtering methods (e.g., KNN [9], MPR [37]) and deep learning methods (e.g. GRU4Rec [11]). For example, Zhou et al. [40] introduced Recurrent

Neural Network (RNN) to predict the expectation of the whole path for learner groups. Some researches proposed to enhance the recommendation strategy by explicitly using cognitive structure. One branch is to model the evolution of knowledge level. Chen et al. [6] and Tang et al. [33] used the transition matrix in MDP to model the evolution of knowledge level and used reinforcement learning algorithm to evaluate the impact of learning items on knowledge level. Meanwhile, works of the other branch focused on employing the knowledge structure to make a recommendation, for example, Zhu et al. [41] made several path generation rules on knowledge structure by expertise and Yu et al. [38] put up a method using semantic inference on ontology to generate learning paths. Previous methods only consider either the importance of knowledge level or knowledge structure without the combination of these two parts. To our best, few of existing works has well established the cognitive structure to make learning path recommendation.

Cognitive Structure. Cognitive structure contains two parts: knowledge level of learners and knowledge structure of items. Two kinds of techniques can be applied to describe these two components, i.e., knowledge tracing for knowledge level and knowledge graph for knowledge structure. Knowledge tracing models learners’ knowledge level over time so that how learners will perform on future interactions can be accurately predicted [7, 8, 31]. Deep Knowledge Tracing (DKT) [27] used RNN to model such states in a high-dimensional and continuous representation. However, the mentioned-above knowledge tracing approaches are hard to fully reveal the relations among the learning items.

Knowledge Graph, with entities (e.g. learning items) as nodes, relations (e.g. prerequisite) as edges, stores a large amount of information containing domain knowledge by graph structure [23]. Meanwhile, the education knowledge graph is able to represent the multi-dimension relationships [5, 13, 18]. Though abundant of knowledge can be thus embedded in Knowledge Graph, the personalized evolution of knowledge level’s characteristic, especially the complexities and the dynamic specialty with infinite time horizons, makes it difficult to be described in the graph structure.

Reinforcement Learning. Deep reinforcement learning, as one of state-of-the-art techniques [1], has shown superior abilities in many fields [36]. The main idea is to learn and refine model

parameters according to task-specific reward signals. For example, Tang et al. [32] introduced reinforcement learning to train an efficient dialogue agent on existing transcripts from clinical trials, which improves mild cognitive impairment prediction; Wang et al. [34] utilized the actor-critic algorithm for treatment recommendation, helping to handle complex relations among multiple medications, diseases and individual characteristics. However, due to three key challenges, the traditional reinforcement learning is difficult to be applied in learning path recommendation: (1) how to represent state; (2) how to avoid the recommendation violating the sequence logicity during exploring; (3) how to reduce the large searching space of learning item paths.

3 PRELIMINARIES

This section discusses the definition of the terminologies and the formulation of learning path recommendation.

3.1 Terminologies

3.1.1 Learning Session. Learning is a long procedure composed of many learning sessions, and each learning session has individual learning targets which can be set by tutors or learners. As shown in Figure 2, each learning session includes two main components: learning path and examinations. Learning path consisting of many learning items is a learning track of a learner. Examinations are used to retrieve the learning effectiveness of the learning path, i.e., the promotion on the learning target. Some typical learning sessions are homework, chapters and semesters, which differ in gratitude. Without loss of generality, the effectiveness of a learning session $E_{\mathcal{P}}$ can be calculated by the following equation:

$$E_{\mathcal{P}} = \frac{E_e - E_s}{E_{sup} - E_s}, \quad (1)$$

where E_s is the score of the beginning examination in a session, E_e is the score of the end, and E_{sup} is the full score of the examination. For example, let $E_s = 80$, $E_e = 90$, $E_{sup} = 100$ and then $E_{\mathcal{P}} = 0.5$, or let $E_s = 0.2$, $E_e = 0.6$, $E_{sup} = 1.0$, then $E_{\mathcal{P}} = 0.5$.

3.1.2 Prerequisite Graph. An educational knowledge graph has some special properties such as prerequisite and similarity, which can represent the knowledge structure. As learners often start from basic items before accessing to those senior ones which are more complicated and hard [5], experts hence summarize a relation of learning items, named prerequisite. A prerequisite graph is a sub-graph of knowledge graph, which indicates the hierarchical structure existing among learning items. As shown in directed acyclic graphs of Figure 1, the nodes of the graph represent learning items while the arrows from one to another mean that the former is a prerequisite for the latter, e.g., *two digit addition* is a prerequisite for *multiplication*.

3.2 Problem Formulation

As mentioned above, learning is composed of many learning sessions. In each session, a learner will try to master a specific learning target $\mathcal{T} = \{t_0, t_1, \dots\}$, which contains one or more learning items. The learning items are the nodes on a prerequisite graph G where the edge represents the prerequisite relation, i.e., (i, j) . The tuple (i, j) indicates that the learning item i is a prerequisite of j . For a learner who is going to begin a new session, the historical learning records generated in previous learning sessions are denoted as $\mathcal{H} = \{h_0, h_1, \dots, h_m\}$. Each record $h_i = \{k, score\}$ contains a learning item k and the corresponding performance *score*. Without loss

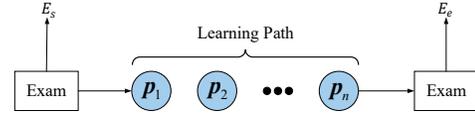


Figure 2: Illustration of a Learning Session.

of generality, we assume *score* as a discrete number 0 or 1, where 1 indicates the corresponding item is correctly answered, while 0 stands in the opposite. Our goal is to recommend an optimized learning path $\mathcal{P} = \{p_0, p_1, \dots, p_N\}$ containing N items to the learner sequentially, by which can the learner achieve a greater promotion. Specifically, at step i , a learning item p_i is recommended and the interaction learning record $\mathcal{F}_i = \{p_i, score_i\}$ can be observed. At the end of the learning session, we can calculate the learning effectiveness $E_{\mathcal{P}}$. After all, the problem is defined as:

DEFINITION 1. (Learning Path Recommendation Problem) Given historical learning records \mathcal{H} , a certain learning target \mathcal{T} of a learner and a prerequisite graph G , our task is to recommend a N -length learning path \mathcal{P} step by step that can maximize the effectiveness $E_{\mathcal{P}}$ of the whole learning path. During recommendation, we can observe a new interaction learning record \mathcal{F}_i of each recommended learning item p_i instantly.

4 CSEAL

This section begins with a brief overview of our framework with the definition of the MDP. The details of CSEAL is then introduced.

4.1 Overview

The target is to learn a policy to recommend tailored learning paths based on the cognitive structure. We model such sequential path generation as a decision making problem and treat it as a MDP [2]. The state, action and reward of the MDP are defined as follows:

State. Generating the probability of learning items at each step is based on the learning target and previous learning records. The state at step i is represented as the combination of the learning target \mathcal{T} and current knowledge level S_i , which are combined and denoted as *state_i*. Specifically, we use one-hot encoding to signify the learning target as $\mathcal{T} = \{0, 1\}^M$:

$$\mathcal{T}^j = \begin{cases} 1 & \text{if } j \text{ in the learning target} \\ 0 & \text{otherwise,} \end{cases} \quad (2)$$

where M is the number of nodes in the prerequisite graph G . However, the current knowledge level can not be observed directly, thus we need to find a way to retrieve it from the previous learning records \mathcal{L}_{i-1} including historical learning records \mathcal{H} and previous interaction learning records $\mathcal{F}_{0, \dots, i-1}$.

Action. Taking action a_i refers to generating the recommended learning item p_i at step i . With the probability of each item as output, the CSEAL can be viewed as a stochastic policy that generates actions by sampling from the distribution $\pi(a|state_i; \theta) = P(a|\mathcal{H}, \mathcal{F}_{0, \dots, i-1}, \mathcal{T}; \theta)$, where θ is the set of model parameters.

Reward. After taking the action, a reward signal r is received. We determine the reward r_i at step i keeps to be 0 during learning session. Once the learning session is completed, the reward is calculated by Equation (1). Our goal is to maximize the sum of the discounted rewards from each step i . That is, the return:

$$R_i = \sum_{j=0}^{N-i} \gamma^j r_{i+j}, \quad (3)$$

where γ is the discount factor, which is usually set to 0.99.

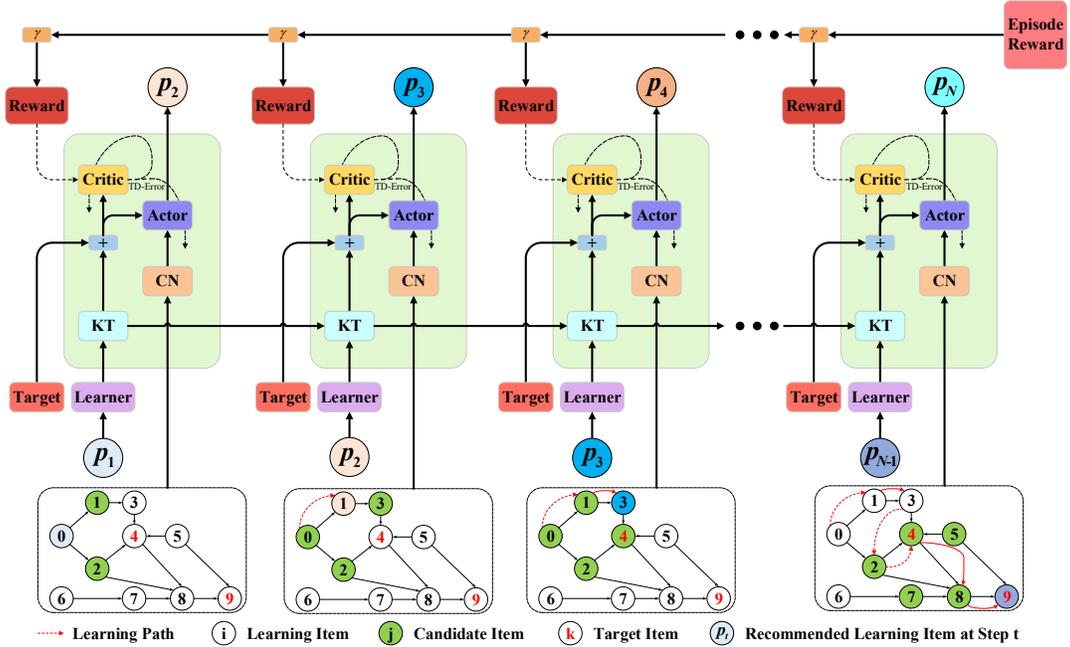


Figure 3: The overview of our framework.

Generally, our CSEAL model contains three submodules, i.e., Knowledge Tracing (KT), Cognitive Navigation (CN), Actor-Critic Recommender (ACR). As shown in Figure 3, KT retrieves the knowledge level S from previous learning records at each step and CN selects several candidates based on the prerequisite graph G . With the learning target and knowledge level composing the state, ACR determines what to learn next by maximizing the overall gain along the whole learning path. At the end of the episode, i.e., learning session, an episode reward will be passed to CSEAL and used in reinforcement learning stage.

4.2 Knowledge Tracing

Knowledge level of learners does greatly influence the strategy of recommending learning path. Also it is a key component of state in MDP, which should be well established. However, learner's knowledge level is unobservable and evolving. To this end, Knowledge Tracing is applied to retrieve the implicit knowledge level S_i from previous learning records $\mathcal{L}_{i-1} = \mathcal{H} \oplus \mathcal{F}_{0, \dots, i-1}$. We follow the structure proposed by the original work of Deep Knowledge Tracing (DKT) [27] and extend an embedding layer which reduces the large feature space. Figure 4 is a cartoon illustration of the architecture of this module. Without the loss of generality, we assume $score_t$ in each record of previous learning records $\mathcal{L}_t = \{p_t, score_t\}$ is either 0 or 1. We use one-hot representation to stand for \mathcal{L}_t as described in DKT.

Firstly, an embedding operation is utilized to convert the one-hot representation of the record into a low-dimensional one. Formally, for a record \mathcal{L}_t , the converted vector x_t is expressed as:

$$x_t = \mathcal{L}_t W_u. \quad (4)$$

Here, $W_u \in \mathbb{R}^{2 \cdot M \times d}$ indicates the parameters of the embedding layer and $x_t \in \mathbb{R}^d$, where d is the output dimension.

After obtaining the feature representation of a learning record, KT aims at tracing the knowledge level along the learning records.

An LSTM architecture as described in DKT is then used to map the input sequence of vectors x_1, x_2, \dots, x_N to the output sequence of hidden knowledge level vectors o_1, o_2, \dots, o_N . The hidden state h_t at the t -th input step is updated as following formulas:

$$\begin{aligned}
 i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i), \\
 f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f), \\
 o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o), \\
 c_t &= f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c), \\
 h_t &= o_t \tanh(c_t).
 \end{aligned} \quad (5)$$

Meanwhile, by using a fully connected layer to retrieve the knowledge level from o_1, o_2, \dots, o_N , vector representation of knowledge level $S_t \in \mathbb{R}^M$ can be expressed as:

$$S_t = \sigma(W_{of}o_t + b_f), \quad (6)$$

where $i_\bullet, f_\bullet, c_\bullet, o_\bullet$ are the input gate, forget gate, memory cell, output gate of LSTM respectively. W_\bullet and b_\bullet are learned weight matrices and biases.

4.3 Cognitive Navigation

The learning items have some inherent semantic structure characteristics, i.e., knowledge structure, which should be maintained to make sure the paths are logical sequences. Intuitively, dependency resolution methods [14, 39] can be applied. However, such methods require clear conditions for the resolution, which is hard to be satisfied in learning path recommendation due to the complexity of combination of knowledge level and knowledge structure. Thus, we focus on quickly selecting the potential candidates. These potential candidates would not only prevent recommended learning items from violating the logicity of the path, especially during exploring (e.g., avoid exploring the effect of recommending *calculus* to junior students) but also reduce the large searching space.

With the prerequisite graph under p_3 in Figure 3 as an example, when a learner finishes the learning item 3, he or she is arranged to review the prerequisites (e.g., item 0, item 2) or continue to

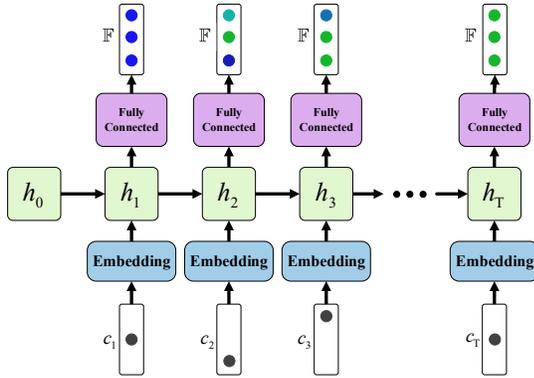


Figure 4: Illustration of an embedding DKT.

preview the the following item 4. We call the just learned item as “central focus” (i.e., item 3) and those items which can be then possibly selected to learn as “candidates” (e.g., item 0, item 1, item 2 and item 4). Simply, these candidates can be chosen from the neighbors of “central focus” which are potential to resolve the dependency of learning targets, more precisely, reach the learning targets. The algorithm runs following the steps shown in Algorithm 1.

Algorithm 1 Cognitive Navigation.

Input: central focus C , a prerequisite graph G , learning target \mathcal{T}
Output: $\forall d \in D$ has a path to \mathcal{T} in G and is one of the k -hop neighbors of C

- 1: Initialize candidates $\mathcal{D} = \emptyset, Q = \emptyset$;
- 2: add C to \mathcal{D} ;
- 3: add successors within $k - 1$ hop of C to \mathcal{D} ;
- 4: add predecessors within k hop of C to Q ;
- 5: **while** $Q \neq \emptyset$ **do**
- 6: $q \leftarrow Q.pop()$;
- 7: add q to \mathcal{D} ;
- 8: add successors within 1 hop of q to D ;
- 9: **end while**
- 10: **for** d in D **do**
- 11: **if** d can not reach T **then**
- 12: del d from D ;
- 13: **end if**
- 14: **end for**
- 15: **return** \mathcal{D}

By setting k as 2, the bottom part of Figure 3 illustrates how central focus and candidates change with path generating where the green ones are the candidates of next step. The central focus C at the first step can be assigned as the last item of \mathcal{H} , which can also be specified manually or chosen from any nodes without predecessors. The Cognitive Navigation is well capable to maintain the logicity of the path during generation. We set $k = 2$ in following analysis.

4.4 Actor-Critic Recommender

Although we achieve the two parts’ information of the cognitive structure, there still exists a problem: which candidate item should be chosen based on the current knowledge level of the learner. To solve this problem, we use a policy network as the actor to generate actions among \mathcal{D} given by Section 4.3 through sampling from the distribution $\pi(a|state_i; \theta)$ and a value network as the critic to evaluate the state. The value network $\mathcal{V}(\cdot; \theta_V)$ is used to estimate the expected return from each state. It is a feed-forward network whose input is $state_i = S_i \oplus \mathcal{T}$, where S_i is the current

knowledge level given by Section 4.2 and \mathcal{T} is the learning target. The estimated expected return \mathcal{V}_i at step i is computed by:

$$v_i = \mathcal{V}(state_i; \theta_V) = \mathcal{V}(S_i \oplus \mathcal{T}; \theta_V). \tag{7}$$

With a stochastic policy together with a value network, we apply the actor-critic algorithm [2, 15] to our sequential generation problem, with the policy network trained using policy gradient at each step i as:

$$\nabla_{\theta} = \log \pi(a|state_i; \theta)(R_i - v_i), \tag{8}$$

and the value network trained by optimizing the distance between the estimated value and actual return:

$$\mathcal{Loss}_{value} = \|v_i - R_i\|_2^2. \tag{9}$$

A too fast convergence of the value network may result in the slow convergence or even no-convergence of policy network. We therefore raise a policy enhanced loss item to address this issue, and the loss function is thus formulated as:

$$\begin{aligned} \mathcal{Loss} = & \|\mathcal{V}(state_i; \theta_V) - R_i\|_2^2 + \alpha \cdot -\log \pi(a|state_i; \theta)(R_i - v_i) \\ & + \beta \cdot -\log \pi(a|state_i; \theta)R_i, \end{aligned} \tag{10}$$

where α and β are the hyper-parameters.

5 EXPERIMENTS

In this section, we first introduce the dataset. Then, we train agents of reinforcement learning models and evaluate recommended learning paths in two kinds of environments. At last, the performance of our framework is compared with several baselines.

5.1 Dataset Description

The experiment dataset is from *junyiacademy.org* and collected by Chang et al. [4]. The dataset includes a knowledge graph and more than 39 million learners’ logs. Each record in the learners’ log contains the information of a learner for one exercise, i.e., user id, concept name, session id, correction and time stamp. The exercise can be mapped to a node in the knowledge graph based on the concept name. And each exercise and concept is one-to-one correspondence¹. Those records with the same session id represent that the practiced data contributed by the same learner in one session. Grouped by session id and sorted by time stamp, session learning records can be extracted (e.g., $\{(representing_numbers, correct), (division_4, wrong), (conditional_statements_2, wrong), (conditional_statements_2, wrong)\}$). We further extract a prerequisite graph from the knowledge graph. The prerequisite graph contains several edges, e.g., $(one_digit_addition, two_digit_addition)$ stands for the linkage between the node *one_digit_addition* and the node *two_digit_addition* where the former is the prerequisite of the latter. We delete some loop in order to keep the graph to be a Directed Acyclic Graph (DAG), which indeed is the knowledge structure. The preprocessed dataset are detailed in Table 1 and Figure 5. Observed from the data, three key notes should be emphasized: (1) The length of more than 75% sessions are longer than 6; (2) more than 75% sessions contain one more concepts; (3) the median of practice frequency on one concept in one session is 8. It conclusively infers that a concept in one session may be practiced repeatedly

¹In some works the exercise and concept may be one-to-many correspondence [7, 27], while others has the same correspondence relationship as ours [5, 6].

Table 1: The statistics of the dataset.

| Statistics | Value |
|--|------------|
| number of learners | 247,548 |
| number of sessions | 525,062 |
| number of learner logs | 39,462,202 |
| number of exercises in learner logs correctly answered | 21,460,360 |
| median of exercises in one session | 21 |
| median of knowledge concepts in one session | 3 |
| median of practice frequency on a concept in one session | 8 |
| number of nodes in graph | 835 |
| number of links in graph | 978 |

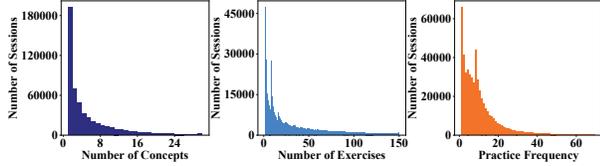


Figure 5: Distributions of Sessions.

and some relevant concepts are learned simultaneously. In other words, during a learning path of the session, multiple concepts and the related ones contribute to the final learning result.

5.2 System Simulators

A key problem is that existing realistic data only contains static information, i.e., several exercise sequences and whether a certain exercise is answered correctly. This information can not be directly employed to analyze whether an exercise not included in a certain exercise sequence can be answered correctly. Thus the realistic data can not be directly used as the environment to evaluate the learning paths (e.g. calculating the promotion) or to train the agents (e.g. CSEAL and other baselines) in reinforcement learning models. It is therefore important to construct a simulator as the environment which can model qualitative development of knowledge and the performance on a certain learning item. More precisely, at each episode, the environment can simulate a learner whose knowledge level can be changed by the recommended path. The knowledge levels of the learner will be measured by the proposed environment at the beginning and the end of the learning session. Hence, E_s , E_e and E_{sup} are obtained by the simulators. The promotion of the level (i.e. E_p) therefore can be calculated. We refer the ideas constructing simulators in state-of-art methods of not only education [6, 33] but also other areas, like transportation [16, 35], e-commerce [12]. Following these works, we raise two ways to build the simulators.

Knowledge Structure based Simulator(KSS). According to previous works [6, 27, 33], we design a simulator where the pattern of qualitative knowledge development fits perfectly for knowledge structure. More specifically, in this simulator, the masteries on prerequisites do affect the successors, e.g., poor mastery of *two digit addition* impairs the learning effectiveness of *multiplication*. To scale the relation of mastery and learner performance, we use a widely applied method in education, Item Response Theory (IRT) [20, 21]. The 3-parameter logistic model of IRT is formulated as:

$$P(\theta) = c + \frac{1 - c}{1 + e^{-Da(\theta - b)}}, \quad (11)$$

where $D = 1.7$ is a constant; c is a pseudo-guessing parameter; θ is the mastery of the learner on a certain learning item; a is the item discrimination; b is the item difficulty; and $P(\theta)$ is the probability

Table 2: Characteristics of the comparison methods.

| | Knowledge Level | Knowledge Structure |
|-----------|-----------------|---------------------|
| KNN | × | × |
| GRU4REC | × | × |
| MCS-10 | ✓ | × |
| MCS-50 | ✓ | × |
| DQN | ✓ | × |
| CSEAL-NCN | ✓ | × |
| CN-Random | × | ✓ |
| Cog | ✓ | ✓ |
| CSEAL | ✓ | ✓ |

of correctly answering the corresponding exercise which is used in computing $score_i$ for each p_i and calculating the reward. The relevant parameters and evolving rules of θ are designed by experts in education which makes KSS a rule-based expert system.

Knowledge Evolution based Simulator(KES). We propose a data-driven method to construct a simulator which can better approximate learners’ knowledge level and be abbreviated as knowledge evolution. We train a DKT model based on the existing data. The input of DKT is a record data, and the output, S_i , is the current knowledge level of the learner. Then whether the next exercise can be answered correctly is determined by this knowledge level. More precisely, the probability of an exercise answered correctly is treated as its mastery value. For example, the probability of an exercise belonged to i being answered correctly is S_i . The probability will be used to compute $score_i$ for each p_i and calculate the reward. To be noticed, KES requires a learning record to initialize the learner’s original knowledge level.

It’s should be noted that neither of these two simulators is perfect. Each of them has its own limitation, i.e., the rule-based knowledge evolution in KSS possibly differs with the real-world and KES has the problem in describing the relations of knowledge structure. As these two simulators are complementary, the proposed approach should outperform others in these diverse environments at the same time to be proved robust. During simulation, the learners mastering all target items at the beginning will be skipped.

5.3 Experimental Setup

5.3.1 Data Partition and Preprocessing. The mentioned-above two simulators are applied as the environment for evaluating and on-line training. Due to their different characteristics, different data partition and preprocessing for them are applied.

KSS. As a rule-based simulator, KSS does not require any extra data to initialize it. Inspired by previous works [6, 27, 33], we employ KSS to generate the off-line dataset, i.e., *dataOff*, which is then used to train baseline methods or Knowledge Tracing model in the agents. The dataset *dataOff* has 4,000 records with a max-length of 50.

Without loss of generality, we randomly divide the dataset *dataOff* into training, validation, and testing sets by the proportion of 80/10/10. The prerequisite graph in KSS contains 10 nodes and 12 links. The learning targets are randomly selected from the nodes.

KES. As a data-driven simulator, KES requires a certain amount of data for training DKT model so that we introduce the learning records mentioned in Section 5.1 as *dataSim*. Furthermore, we randomly divide *dataSim* into two parts: *dataOff* and *dataRec* by the proportion 50/50. The details of three datasets are listed as follows:

- (1) *dataSim* is utilized to train the DKT model in the environment;

Table 3: Overall results of $E\mathcal{P}$.

| | KSS | KES |
|-----------|-----------------|-----------------|
| KNN | 0.000700 | 0.257919 |
| GRU4Rec | 0.007727 | 0.201219 |
| MC-10 | 0.110577 | 0.002236 |
| MC-50 | 0.108636 | -0.005103 |
| DQN | 0.100610 | 0.002688 |
| CSEAL-NCN | 0.222363 | 0.003354 |
| CN-Random | 0.272784 | 0.138526 |
| Cog | 0.164128 | 0.166560 |
| CSEAL | 0.346883 | 0.405823 |

(2) *dataOff* is used to train baseline methods and Knowledge Tracing model in the agents, which is the same as in KSS;

(3) *dataRec* is to retrieve the initialization records and learning targets. For each session record in *dataRec*, the first 60% of the data is applied to initialize the original learner’s knowledge level in DKT. The middle 20% is masked and the lasting 20% is reserved as the learning target of this session.

We then divide each of those above datasets (i.e., *dataDKT*, *dataSimDKT* and *dataRec*) for training, validation, and testing by the proportion of 80/10/10.

5.3.2 Framework Setting. Due to the different number of learning items in two simulators where 10 in KSS (a small number learning items scenario) and 835 in KES (a large number learning items scenario), different settings are set as follows:

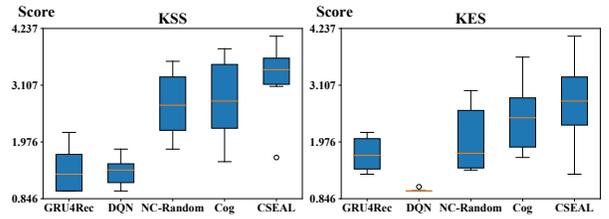
- **KSS:** The embedding dimension used in DKT is 15, hidden dimension of LSTM is 20 while the dimension of output layer is the same as the number of learning items, i.e., 10. The dimensions of two layers in value-policy network are 128 and 32 respectively.
- **KES:** The embedding dimension used in DKT is 600, hidden dimension of LSTM is 900 and the dimension of output layer is the same as the number of learning items, i.e., 835. The dimensions of two layers in value-policy network are 1,024 and 512 respectively. The DKT embedded in environment and agent shares the same value of dimension parameters but diverse training data.

5.3.3 Training Details. We initialize parameters in all networks with *Xavier* initialization [10], which is designed to keep the scale of gradients roughly the same in all layers. The initialization fills the weights with random values in the range of $[-c, c]$ where $c = \sqrt{\frac{3}{n_{in} + n_{out}}}$. n_{in} is the number of neurons feeding into weights, and n_{out} is the number of neurons the result is fed to. We set mini-batches as 16. We also use dropout [30] with the probability 0.2 for DKT embedding and each output in value-policy network and 0.5 for LSTM output to prevent overfitting and gradient clipping [24] to avoid the gradient explosion problem. Some of our codes are available in <https://github.com/bigdata-ustc>.

5.4 Baseline Approaches

In order to demonstrate the effectiveness and robustness of our framework, we compare it with following methods.

- **KNN:** KNN [9] finds a predefined number of learners nearest to the new learner by comparing the cosine distance of their learning paths, and decides what to learn next for the new learner.
- **GRU4Rec:** GRU4Rec is a classical session-based recommendation model [11]. The input of the model is the sequence of the session while the output is the probability distribution of learning items which appear in the next step.

**Figure 6: Overall results of human study.**

- **MCS:** Monte Carlo Search (MCS) [22] combined with KT is a searching method where KT predicts the promotion of each search path as the index for ranking.
- **DQN:** Some works [6, 33] have proposed to leverage reinforcement learning for this problem, but these works require abundant human domain knowledge to design the transition matrix in MDP and an exact initial state, which is not practical. Thus KT model and Deep Q Learning replace the states in MDP and simple q-learning separately.
- **CN-Random:** The recommended item is randomly picked from the candidate items selected by CN, and this baseline is treated as a simple knowledge structure based approach.
- **Cog:** The recommended item is weighted-randomly picked from the candidate items selected by CN, where the weight of the item is inversely proportional to the mastery measured by KT model.
- **CSEAL-NCN:** It is similar with our proposed model but without the Cognitive Navigation system.

For better illustration, we summarize the characteristics of these models in Table 2. All deep learning involved models are implemented by MXNet and trained on a Linux server with four 2.0GHz Intel Xeon E5-2620 CPUs and a Tesla K20m GPU.

5.5 Evaluation Metrics

Learning path recommendation focuses on the learning effectiveness rather than the selection of the learner (i.e., being practiced by the learner) or the correction of one exercise, which is essentially different from the general recommendation problem (e.g., merchandise recommendation, movie recommendation) [41]. Because of this issue, classical metrics like precision, recall and NDCG can not be applied. However, the quantification of learning effect is not very clear and still a challenge [29]. Previous works use either the logicity of sequence [41] or the promotion of knowledge level [6, 29] (i.e., $E\mathcal{P}$) as the quantitative metric to evaluate the learning path. For more comprehensive results, we use both logicity and promotion as metrics for evaluation. We validate the performance of models particularly based on the promotion $E\mathcal{P}$ given by simulators and the logicity evaluated by human experts.

5.6 Experimental Results

5.6.1 Promotion Comparison. The length of recommended learning paths is set consistently to be 20 according to the median of exercises in one session estimated in Table 1. Extensive experiments on the length of path will be conducted in Section 5.6.4.

Table 3 shows the average $E\mathcal{P}$ according to Equation (1) of all models. According to the results, obviously CSEAL performs best. Specifically, by modeling the knowledge levels, it beats CN-Random, GRU4Rec and KNN. By applying Cognitive Navigation on the knowledge structure, it achieves the better performance than DQN, MC-10, MC-50 and CSEAL-NCN. By well combining

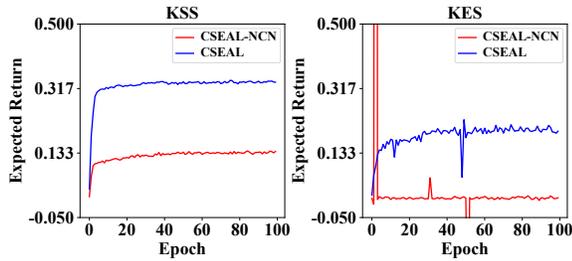


Figure 7: Expected return in different learning epochs with or without Cognitive Navigation.

the knowledge level and the knowledge structure, it beats Cog. Then, in KSS, the methods with knowledge structure have better performance than the ones without knowledge structure which indicates that the knowledge structure contributes a lot to the effectiveness of recommended learning paths. Last but not least, in KES, GRU4Rec without explicitly modeling the knowledge level and the knowledge structure outperforms other methods except CSEAL. These observations infer that exploiting cognitive structure with the comprehensive consideration of knowledge level and knowledge structure for adaptive learning is necessary but challenging.

5.6.2 Experts Comparison. Inspired by previous works [41], we invite six experts in education area who are familiar with learning path scheduling to evaluate the results of various methods based on their own logicity. Experts rate every learning path with a score from 1 to 5 and a higher score represents the higher logicity. Experts are asked to rate the 100 selected cases. Every case contains the historical learning record, the learning target and the recommended learning path. Four typical baselines, GRU4Rec, DQN, CN-Random and Cog, differing in whether having knowledge level or knowledge structure are selected as the comparison methods.

As can be seen in Figure 6, CSEAL outperforms all baselines. In other words, recommendations from CSEAL are in the most accordance with logicity of knowledge structure. Besides, it is observed that the methods with the Cognitive Navigation achieve a higher scores in experts evaluating, which indicates that the Cognitive Navigation helps to maintain the logicity of learning paths. Furthermore, two interesting phenomena draw our attention: (1) the scores of different experts for the same case sometimes are quite different; (2) compared with the former result of E_p , we notice some models with higher score of logicity may not achieve better promotion. From these observations, the exact definition of logicity is not easy to be expressed and captured but varies in different people and it is additionally not equal to the promotion.

5.6.3 Impact of Knowledge Structure. Figure 7 presents the expected return obtained in each learning epoch of CSEAL and the variant CSEAL-NCN which does not have Cognitive Navigation. Notably, CSEAL is able to utilize the knowledge structure to obtain the optimal policy in a stable manner with the help of Cognitive Navigation. We can see CSEAL-NCN obtains a better policy in KSS than in KES because of the smaller searching space (i.e., smaller graph) in KSS. These issues indicate the knowledge structure can help reduce the searching space in reinforcement learning.

5.6.4 Performance with Different Length. As shown in Section 5.1, the median of the length of a session is 21, from which we suspect that the most suitable learning length in KES should be near to it. We select the same methods in Section 5.6.2 to verify our suspicion,

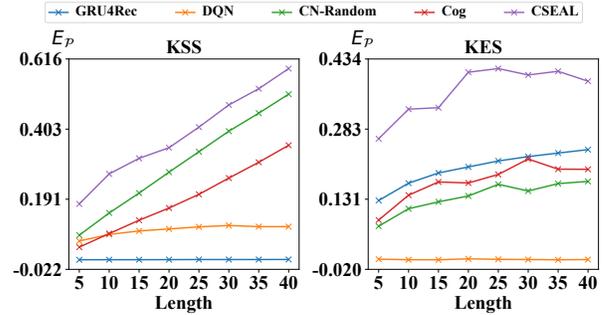


Figure 8: Influence of learning path length.

and an extensive experiment is also performed on KSS. Figure 8 shows the results. We can see that, in KSS, E_p grows with the length increasing because KSS is made by rules which have no limitation on length. While the effectiveness can hardly be promoted for all methods after 20 in KES, which verifies our suspicion.

5.6.5 Case Study. We give an example with visualization in Figure 9. In the example, a latest learning record of a learner who wants to learn the learning item 642, *completing_the_square_1*, is $\{(642, 0), (642, 0), (642, 0), (642, 0), (642, 0)\}$ which illustrates that the learner possibly has some trouble in learning the target. To help this learner, different methods give different learning paths. For better understanding, we draw the subgraph containing three-hop neighbors of the target in Figure 9. GRU4Rec recommends a path where the learner continuously directly learns the target which he or she already seems to be stuck with. Cog recommends a path, where most learning items are far away from the target. Our method, CSEAL, encourages the learner firstly to review the prerequisite learning items and then go back to learn the target with reviewing the prerequisites. This visualization hints that CSEAL provides a more efficient and logical learning path for the learner to master the learning target.

6 CONCLUSIONS

In this paper, we proposed a novel recommendation framework for adaptive learning, named CSEAL. Specifically, based on the historical learning records, learning target and prerequisite graph, we firstly used a Knowledge Tracing model to retrieve the knowledge level of each learner. Then, we applied a Cognitive Navigation system to maintain the knowledge structure of learning items. Finally, we designed the Actor-Critic Recommender to dynamically provide learning items during a learning cycle. The experimental comparisons with seven baseline methods on two Simulators (i.e., KSS and KES) with diverse scenarios and human experts clearly demonstrated both the effectiveness and robustness of our framework. As a general framework, each step of CSEAL may be further improved in the future.

Acknowledgements. This research was partially supported by grants from the National Key Research and Development Program of China (No. 2018YFC0832101), the National Natural Science Foundation of China (Grants No. 61672483, U1605251), and the Science Foundation of Ministry of Education of China & China Mobile (No. MCM20170507). Qi Liu gratefully acknowledges the support of the Young Elite Scientist Sponsorship Program of CAST and the Youth Innovation Promotion Association of CAS (No. 2014299).

