MaHRL: Multi-goals Abstraction Based Deep Hierarchical Reinforcement Learning for Recommendations

Dongyang Zhao
Peking University
zdy_macs@pku.edu.cn

Liang Zhang*
JD.com
zhangliangshuxue@gmail.com

Bo Zhang
JD.com
zhangbo35@jd.com

Lizhou Zheng
JD.com
interson.zlz@gmail.com

Yongjun Bao
JD.com
baoyongjun@jd.com

Weipeng Yan
JD.com
Paul.yan@jd.com

ABSTRACT

As huge commercial value of the recommender system, there has been growing interest to improve its performance in recent years. The majority of existing methods have achieved great improvement on the metric of click, but perform poorly on the metric of conversion possibly due to its extremely sparse feedback signal. To track this challenge, we design a novel deep hierarchical reinforcement learning based recommendation framework to model consumers' hierarchical purchase interest. Specifically, the high-level agent catches long-term sparse conversion interest, and automatically sets abstract goals for low-level agent, while the low-level agent follows the abstract goals and catches short-term click interest via interacting with real-time environment. To solve the inherent problem in hierarchical reinforcement learning, we propose a novel multi-goals abstraction based deep hierarchical reinforcement learning algorithm (MaHRL). Our proposed algorithm contains three contributions: 1) the high-level agent generates multiple goals to guide the low-level agent in different sub-periods, which reduces the difficulty of approaching high-level goals; 2) different goals share the same state encoder structure and its parameters, which increases the update frequency of the high-level agent and thus accelerates the convergence of our proposed algorithm; 3) an appreciated reward assignment mechanism is designed to allocate rewards in each goal so as to coordinate different goals in a consistent direction. We evaluate our proposed algorithm based on a real-world e-commerce dataset and validate its effectiveness.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Theory of computation** → **Reinforcement learning**.

KEYWORDS

Recommender Systems; Deep Hierarchical Reinforcement Learning; Conversion; Multi-goals

ACM Reference Format:
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 (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401170

1 INTRODUCTION

In this information era, end users/consumers usually suffer from heavy burden of content and product choices when browsing the Internet. The recommender system is an important form of intelligent application, which assists users to alleviate from such information redundancy. More specifically, the recommender agents discover users' short-term and long-term interest/ preference from their browsing histories in Internet. They build user models based on their interest/preference and automatically recommend personalized items so as to satisfy users' information needs. As a result, the recommender systems have become increasingly popular, and have been applied to a variety of domains in Internet such as e-commerce, news, movies [18, 19].

To improve the performance of recommender systems, lots of works have been proposed, evolving from the traditional models like collaborative filtering [2], to the mainstream deep models like Wide&Deep [4] and finally to the trend of deep reinforcement learning based methods [31]. The deep models have shown excellent performance, due to their powerful capabilities of extracting highly non-linear features and relationships. For instance, DIEN [33] designed an interest extractor layer to capture temporal interest from historical behavior sequence. Most of these deep methods are static, which can hardly follow dynamic changes of users' preference. The deep reinforcement learning (DRL) based methods overcome this problem via interacting with users in real time and dynamically adjust the recommendation strategies. For instance, DEERS [31] adopted a Deep Q-Network framework and integrated both positive and negative feedback simultaneously. Furthermore, the DRL based recommendations maximize the long-term cumulative expected returns, instead of just immediate (short-term) rewards as traditional deep model, which can bring more benefits in the future.

At present, the majority of works about recommender systems focus on optimizing the metric of click and have already achieved

*Corresponding author.

SIGIR '20, July 25–30, 2020, Virtual Event, China

ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401170

great improvements [4, 25, 33]. As the competition becomes fiercer, the recommender agents gradually pay more attention on the metric of conversion, especially in e-commerce recommender systems. On the one hand, the metric of conversion is more realistic as counterfeiting conversion is more difficult. On the other hand, the e-commerce recommender systems usually recommend natural items and display ads together. The advertisers care more about the direct conversions, instead of indirect clicks, so as to guarantee their revenue over investment. Few of works consider the metric of conversion [21, 27]. For instance, Yang et al. [27] combined natural language processing and dynamic transfer learning into a unified framework for conversion rate prediction. Either of these works only optimize the metric of click or the metric of conversion. The click and conversion are highly correlated, but may not have the positive correlation. An item which is more likely to be clicked, may result in lower probability of conversion, e.g., the item with relative cheap price but poor product quality.

In this paper, we adopt deep reinforcement learning based methods to optimize the metrics of click and conversion jointly. The sequential user behaviors from impression, to click and finally conversion, reflects users' hierarchical interest. More specifically, the click signals from exposed items reflect various superficial interest such as the curiosity for new items, the return clicks for previously purchased items, the initial purchase willingness, etc., while the conversion signals from clicked items show the pure and deep purchase interest. The existing DRL based methods in recommendations usually treat the sparser conversion signals just as same as the clicks, except for assigning some larger weights. For instance, Hu et al. [8] assigned the conversion weight according to the price of each product item. The large weights can partially alleviate the sparsity problem of conversion signals, but this results in more serious instability for DRL based algorithms. What's worse, such method requires DRL techniques to track the conversion signals from impressions directly, just as tracking click signals from impressions. The sparse conversion events might still be overwhelmed easily by relative non-sparse click.

To solve this sparsity problem, we propose a new hierarchical reinforcement learning (HRL) based recommender framework, which consists of two components, i.e., high-level agent and low-level agent. More specifically, the high-level agent tries to catch the long-term sparse conversion signals based on users' click and conversion histories, and automatically sets goals for the low-level agent. On the other hand, the low-level agent captures the short-term click signals based on users' impression and click histories, and interacts with the environment (i.e., users) via recommending items and receiving feedback. This framework differentiates users' hierarchical interest via hierarchical agents efficiently. There exist several problems in this framework. Firstly, how does the high-level agent automatically generates goals for the low-level agent. There exists no explicit goals for the high-level agent in recommender systems. Secondly, how does the high-level goals influence the low-level agent. The appropriate way to guide the low-level agent can reduce the difficulty of approaching the high-level goals. Thirdly, how to increase the update frequency of high-level agent so as to accelerate its convergence. The feedback frequency of high-level agent is far less than that of low-level agent.

To tackle these challenges, we further propose a novel multi-goals abstraction based deep hierarchical reinforcement learning algorithm, namely MaHRL. We demonstrate the MaHRL algorithm via the interaction between recommender agents and users. More specifically, the high-level agent first generates a set of abstract goals based on users' click and conversion histories, and conveys them to the low-level agent. Each goal has the same form as the action of the low-level agent. Furthermore, different abstract goals guide the low-level agent in different interaction sub-periods. All these designs make the high-level goals easier to approach. Then, the low-level agent generates real recommended items based on users' browsing and click histories, and collects users' feedback as external reward. The low-level agent also accepts the internal reward, which is generated from the distance between the action and its corresponding goal. Finally, the low-level agent conveys the users' feedback to the high-level agent, which improves the quality of goals in the following interactions. To enhance the cooperation of each goal, we design the shared state encoder structure for each goal. Its parameters are updated when each goal updates its own parameters, which accelerates the convergence of our proposed algorithm. In addition, we design an appreciated reward assignment mechanism which allocates rewards in each goal so as to coordinate the goals in a consistent direction.

In summary, this paper has the following contributions:

- We propose a new HRL based recommender framework. The high-level agent catches long-term sparse conversion signals, while the low-level agent captures short-term click signals.
- We propose a novel multi-goals abstraction based deep hierarchical reinforcement learning algorithm (MaHRL). The multiple high-level goals reduce the difficulty for the low-level agent to approach the high-level goals.
- We design a shared state encoder for each goal which accelerates the update frequency, and a reward assignment mechanism to coordinate different goals correctly.
- We carry out the offline and online evaluation based on the real-world e-commerce dataset. The experimental results demonstrate the effectiveness of our proposed algorithm.

In this paper, we first introduce the details of our proposed framework in Section 2. Then, we present our training procedure in Section 3. After that, we demonstrate our experiments in Section 4. The related work is discussed in Section 5. At last, we conclude this paper in Section 6.

2 THE PROPOSED FRAMEWORK

This section begins with an overview of our proposed HRL based recommender framework. Then we introduce the technical details of the high-level agent and the low-level agent in this framework.

2.1 Framework Overview

In this work, we model the recommendation task as Markov Decision Process (MDP). Users are regarded as the environment, while the recommender system is regarded as the agent. We utilize users' behaviors to represent the environment state. According to current state, the agent selects an action (a recommended item), and then

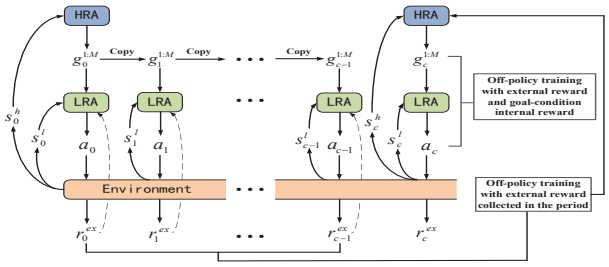


Figure 1: The interaction procedure.

the environment gives feedback: browse (skip), click, order (convert), or leave, etc. The agent obtains corresponding rewards, and the state of the environment is updated.

Based on the above settings, we propose a novel HRL based recommender framework, including a high-level recommendation agent (HRA) and a low-level recommendation agent (LRA). In order to express our ideas clearly, firstly we define the notations required.

- **High-level state space** S^H: A high-level state $s^h \in S^H$ is a hidden vector representing user's current long-term preference, which is extracted from user's click histories $\{e_1^c, e_2^c, \cdots, e_N^c\}$ and order histories $\{e_1^o, e_2^o, \cdots, e_N^o\}$. e_i^c or e_i^o is a vector representing real recommended item.
- **Low-level state space** S^L: A low-level state $s^l \in S^L$ is a hidden vector representing user's current short-term preference, which is extracted from user's browsing histories $\{e_1^b, e_2^b, \cdots, e_N^b\}$ and click histories $\{e_1^c, e_2^c, \cdots, e_N^c\}$.
- **Action space** A: An action $a \in A$ is a vector representing a recommended item.
- **Goal space** G: A goal $g \in G$ is a vector generated from s^h, which is used to guide the generation of a.
- **Internal Reward** R^{IN}: An internal reward $r^{in}(g, a)$ is a scalar utilized to evaluate the degree of guidance.
- **External Reward** R^{EX}: An external reward $r^{ex}(s^l, a)$ is a scalar used to evaluate users' feedback.
- **Transition** P^H / P^L: Transition $p(s^{h'}|s^h, g)$ / $p(s^{l'}|s^l, a)$ defines the state transition in high/low-level state spaces.
- **Discount factor** γ: $\gamma \in [0, 1]$ defines the discount factor to measure the current value of future rewards. In particular, when $\gamma = 0$, the agents only consider the immediate reward. When $\gamma = 1$, all future rewards are counted in without loss.

The goal of HRL is to find a recommendation policy $\pi = (\pi^h, \pi^l)$, where we define high-level policy $\pi^h : S^H \to G$, and low-level policy $\pi^l : S^L \times G \to A$, which can maximize the cumulative external rewards $E_\pi[\sum_{k=t}^{T} \gamma^{k-t} r^{ex}(s_k^l, a_k)|s_t^h, s_t^l]$.

Figure 1 demonstrates the interactions between the agents and the environment under the HRL-based recommender framework. More specifically, the environment provides a high-level state s_t^h and a low-level state s_t^l at time step t. The HRA observes the high-level state s_t^h and produces a set of goals $g_t^{1:M} = \{g_t^1, g_t^2, \cdots, g_t^M\}$ when $t \equiv 0 (mod\ c)$. This provides temporal abstraction, since HRA produces goals only every c steps, which would be utilized to guide LRA in the entire c steps. The LRA observes the low-level state s_t^l and the set of goals $g_t^{1:M}$, and produces a low-level atomic action a_t applied to the environment. Then the LRA receives an internal reward r_t^{in} and an external reward r_t^{ex}. The internal reward is sampled from the internal reward function $r_t^{in}(g_t^{1:M}, a_t)$, which indicates how the LRA follows the goals. The external reward is provided by the environment which represents users' actual feedback. As the consequence of action a_t, the environment updates the high-level state to s_{t+1}^h with high-level transition $p(s_{t+1}^h|s_t^h, a_t)$ and updates the low-level state to s_{t+1}^l with low-level transition $p(s_{t+1}^l|s_t^l, a_t)$. After c time steps (from t to $t+c$), the LRA collects recent c external rewards $r_{t:t+c-1}^{ex} = (r_t^{ex}, r_{t+1}^{ex}, \cdots, r_{t+c-1}^{ex})$ and conveys them to the HRA to improve its performance. In the interaction procedure mentioned above, the LRA will store the experience $(s_t^l, g_t^{1:M}, a_t, r_t^{ex}, s_{t+1}^l)$ for off-policy training, while the HRA will store the experience $(s_t^h, g_t^{1:M}, r_{t:t+c-1}^{ex}, s_{t+c}^h)$ for off-policy training.

Both HRA and LRA adopt Actor-Critic architectures. The actor of HRA aims to produce a set of abstract goals $g^{1:M}$, while the critic tries to evaluate the expected return achievable by the high-level policy as follows:

$$Q_i^h(s^h, g^i) = E_{\pi^h}\left[r_i^h + \gamma Q_i^h(s^{h'}, g^{i'})|s^h, g^i\right],\ 1 \le i \le M, \quad (1)$$

where s^h and g^i are the current high-level state and goal, and $s^{h'}$ and $g^{i'}$ are the next high-level state and goal. In addition, r_i^h represents the high-level reward obtained under the guidance of goal g^i. The actor of LRA aims to output a deterministic action a, while the critic of LRA tries to evaluate the expected return achievable by the low-level policy as follows:

$$Q^l(s^l, a) = E_{\pi^l}\left[r^l + \gamma Q^l(s^{l'}, a')|s^l, a\right], \quad (2)$$

where s^l and a are the current low-level state and real action, and $s^{l'}$ and a' are the next low-level state and real action. Furthermore, r^l represents the low-level reward received by LRA after taking action a. We elaborate the details of HRA and LRA architecture in the next subsections.

2.2 Architecture of High-Level Agent

In this subsection, we first introduce the state encoder commonly used by the actor and critic of HRA, and then describe their architectures in details.

2.2.1 State Encoder for HRA. We use state encoder to extract the high-level hidden state from user's behaviors, which includes user's latest clicked items $\{e_1^c, e_2^c, \cdots, e_N^c\}$ and latest ordered items $\{e_1^o, e_2^o, \cdots, e_N^o\}$ (sorted in chronological order) before the current time step. Here, each clicked item e_i^c and each ordered item e_i^o can be represented by dense and low-dimensional vector. The actor and the critic of HRA have the same encoder structure but with different parameters. The output of the encoder is further processed either in the actor or in the critic.

We introduce an RNN model with Gated Recurrent Units (GRU) in the state encoder to capture users' long-term preference from their sequential behaviors. We leverage GRU rather than Long Short-Term Memory (LSTM) because GRU has similar performance, but smaller number of parameters [7]. We use the final hidden state h_N as the output of RNN layer. In our framework, two such RNN models with GRU are used seperately. The one receives user's last clicked items $\{e_1^c, e_2^c, \cdots, e_N^c\}$ as input and outputs the final

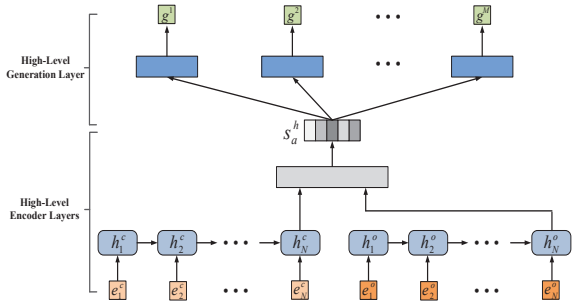


Figure 2: The architecture of high-level actor.

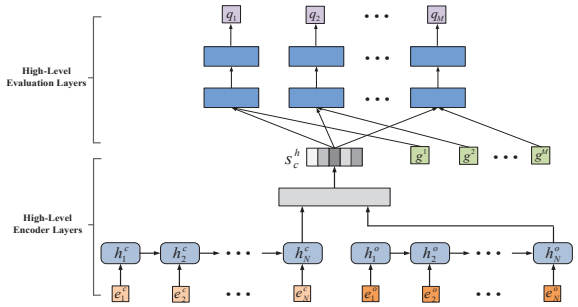


Figure 3: The architecture of high-level critic.

hidden state h^c_N, while the other receives user's last ordered items $\{e^o_1, e^o_2, \cdots, e^o_N\}$ as input and outputs the final hidden state h^o_N. Finally, a linear layer is utilized to merge the two hidden states and produce the user's long-term preference:

$$s^h = w^h_c h^c_N + w^h_o h^o_N + b^h_s. \tag{3}$$

2.2.2 Actor Architecture of HRA. The actor of HRA aims to generate multiple abstract goals, which are used to guide the low-level agent. To obtain such goals, the actor of HRA first abstracts the high-level state s^h_a based on the state encoder mentioned previously from users' click/order behaviors. After that, M parallel fully connected layers are used behind to generate M abstract goals. We omit the sub-index which differentiates the parameters of different goals in both the actor and the critic of HRA, e.g., g can represent any goal $g_i, 1 < i < M$. Thus, we have:

$$g = B \tanh(w^h_a s^h_a + b^h_a), \tag{4}$$

where parameter B represents the bound of goals and "tanh" activate function is utilized to ensure $g \in (-B, B)$.

We illustrate the details of the actor's architecture in Figure 2. Note that all M goals share the same encoder structure, but their generation layers are different. Due to the sharing mechanism, when each goal updates its related parameters in the learning procedure, both the generation layer and the encoder layer will be updated. The update frequency of parameters in the state encoder is M times than that in the generation layer. It has two advantages: 1) the update frequency of complex state encoder is greatly improved; 2) the state encoder can obtain information from multiple goals, which improves its stability.

2.2.3 Critic Architecture of HRA. The critic of HRA leverages an goal-value estimator to learn expected high-level return under the goal g shown in Eq. (1), which is a also judgement of whether the goal generated by the high-level policy match the current high-level state s^h_c. Based on expected returns, the high-level policy network can update its' corresponding parameters in a direction of improving performance. To estimate the expected returns for each goal, we abstract high-level state s^h_c via the state encoder of HRA from users' click/order behaviors, similar with the actor of HRA, but with different parameters. After that, we combine the high-level state s^h_c and each goal g via a fully connected layer with "ReLU" as its activation function to capture the non-linear relationship. Based on the hidden output embedding v_h, we further add one fully connected layer to extract deeper long-term preference and obtain the estimated expected return q^h. We also use "ReLU" as its activation function since $q^h \in (0, +\infty)$. More formally, we have:

$$v^h = Relu(w^h_1 s^h_c + w^h_2 g + b^h_v),\ q^h = Relu(w^h_v v^h_h + b^h_q). \tag{5}$$

We illustrate the details of the actor's architecture in Figure 3. In this architecture, M parallel evaluation layers are placed behind the shared encoder layers. Due to the sharing mechanism, the update speed and convergence stability of high-level critic are also improved. Note that each output of high-level critic estimates the expected return of each goal generated by high-level policy network. To coordinate each goal in a consistent direction of guiding LRA, rewards for the high-level critic should be designed carefully. In this paper, we propose a reward assignment mechanism to deal with this problem, which would be discussed in Section 3.2.

2.3 Architecture of Low-Level Agent

In this subsection, we first introduce the state encoder which is commonly used by the actor and critic of LRA, and then describe the actor and critic in details.

2.3.1 State Encoder for LRA. In our state encoder for LRA, two RNN models with GRU, similar to HRA in Section 2.2.1 are utilized. The one receives user's latest browsed items $\{e^b_1, e^b_2, \cdots, e^b_N\}$ as input and outputs the final hidden state h^b_N, while the other receives user's latest clicked items $\{e^c_1, e^c_2, \cdots, e^c_N\}$ as input and outputs the final hidden state h^c_N. Finally, a linear layer is utilized to merge the two states and produce the low-level hidden state:

$$s^l = w^l_b h^b_N + w^l_c h^c_N + b^l_s. \tag{6}$$

2.3.2 Actor Architecture of LRA. The actor of LRA aims to generate real recommended items based on users' behaviors. To obtain such action, the actor of LRA first abstracts the low-level state s^l_a based on the previous state encoder from users' browsed/clicked behaviors. After that, a fully connected layer is utilized behind as the action generation layer, i.e.,

$$\hat{a} = B \tanh(w^l_a s^l_a + b^l_a), \tag{7}$$

where parameter B represents the bound of the action and we utilize "tanh" as the activate function.

We illustrate the details of the actor's architecture in Figure 4. Note that the generated item embedding $\hat{a}$ may be not in the real

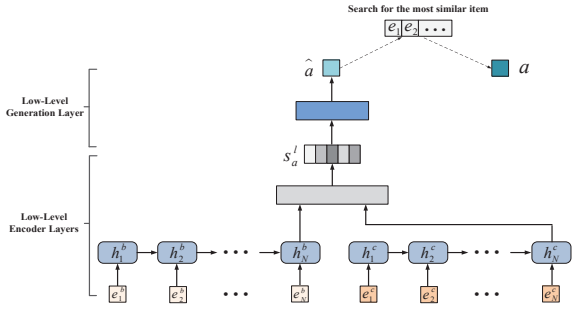


Figure 4: The architecture of low-level actor.

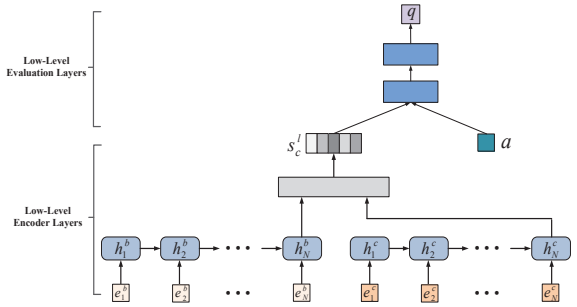


Figure 5: The architecture of low-level critic.

item embedding set. Thus, we need to map it to real item embedding, which will be provided in Section 3.1.

2.3.3 Critic Architecture of LRA. The critic of LRA leverages an action-value estimator to learn expected low-level return under the real action a, shown in Eq. (2). Similar to the high-level critic, the critic of LRA can assist the low-level policy in updating its' corresponding parameters in a direction of generating better actions. To estimate the expected return, we abstract the low-level hidden state s_c^l via state encoder of LRA from users' browsed/clicked behaviors, similar to the low-level actor, but with different parameters. After that, we combine the low-level state s_c^l and real action a via a fully connected layer with "ReLU" as its activation function. Based on the hidden embedding v_l, we further add one fully connected layer to extract deeper short-term preference and obtain the estimated expected return q^l. We also utilize "ReLU" as its activation function since $q^l \in (0, +\infty)$. More formally, we have:

$$v^l = Relu(w_1^l s_c^l + w_2^l a + b_v^l),\ q^l = Relu(w_q^l v_l + b_q^l). \tag{8}$$

The detail of the whole architecture for low-level critic is illustrated in Figure 5.

3 TRAINING PROCEDURE

With the above HRL based recommender framework, we can discuss the training procedure in this section. We first introduce the action mapping method and reward assignment mechanism. Then we demonstrate our training algorithm in details.

3.1 Action Mapping

As mentioned in Section 2.3.2, we generate a recommended item embedding $\hat{a}$ based on the user's short-term preference s_a^l. But $\hat{a}$ is a virtual-action because it may not be in the real item set I. Note that I may be the static whole candidate set or dynamic subset with most relative candidate items. So we have to map this virtual-action $\hat{a}$ into a real action a (a real item embedding). Under this setting, for each $\hat{a}$, we choose the most similar $a \in I$ as the real item embedding. In this work, we use cosine similarity as the metric:

$$a = \arg\max_{a_i \in I} \frac{\hat{a}^T \cdot a_i}{\|\hat{a}\| \|a_i\|} = \arg\max_{a_i \in I} \hat{a}^T \cdot \frac{a_i}{\|a_i\|}. \tag{9}$$

To reduce the amount of computation, we pre-compute $\frac{a_i}{\|a_i\|}$ for all $a_i \in I$ and use the item recall mechanism to eliminate irrelevant and redundant items. Thus, with the action mapping function, denoted as $f(\cdot)$, and the low-level actor network, denoted as $\hat{\pi}^l(\cdot)$, we can obtain the low-level policy as $\pi^l(\cdot) = f(\hat{\pi}^l(\cdot))$.

3.2 Reward Assignment Mechanism

3.2.1 Low-level Reward. As shown in Eq. (2), the low-level reward r^l affects the expected returns for the low-level critic, and finally determines the update direction of the low-level actor. The low-level reward consists of two parts, i.e., the external reward r^{ex} and the internal reward r^{in}. The external reward accepts the feedback from users while the internal reward accepts the guidance from HRA. More formally, we have:

$$r^l = r^{ex}(s^l, a) + \alpha r^{in}(g^{1:M}, a), \tag{10}$$

where the hyper-parameter α regulates the influence of the internal reward. Since the internal reward connects HRA and LRA, a reasonable internal reward function $r^{in}(\cdot)$ is important for HRA to guide the real action of LRA a via different goals $g^{1:M}$. In this work, we cut a period of c steps into equal M sub-periods, and utilize each goal to guide LRA in its corresponding sub-period with $\lfloor c/M \rfloor$ steps. Only one goal would be utilized in each time step. To measure the degree of guidance, we introduce the cosine similarity to measure the distance between each action and its corresponding goal, and thus produce the internal reward:

$$r_t^{in}(g_t^{1:M}, a_t) = \frac{a_t^T \cdot g_t^i}{\|a_t\| \left\|g_t^i\right\|}, \lfloor c/M \rfloor (i-1) < t \leq \lfloor c/M \rfloor i. \tag{11}$$

3.2.2 High-level Reward. As shown in Eq. (1), the high-level reward r^h affects the expected return for the high-level critic, and finally determines the update of the high-level actor. Since there exist M goals, a reasonable reward assignment mechanism, which decides the rewards of each goal, is very important to coordinate them in a consistent direction of guiding LRA. As mentioned previously, each goal g^i in HRA guides the LRA in continuous $\lfloor c/M \rfloor$ steps in sub-period i, thus it would collect all external feedback from users during this sub-period. A naive method of allocating rewards for goal g^i is to accumulate all external rewards in sub-period i, i.e.,

$$\phi_i(r_{t:t+c-1}^{ex}) = \sum_{j=\lfloor c/M \rfloor (i-1)}^{\lfloor c/M \rfloor i-1} r_{t+j}^{ex}. \tag{12}$$

However, such assignment method does not consider the coordination among different goals. To deal with this problem, we

propose an extended high-level reward assignment function, denoted as $\phi_i'(r^{ex}_{t:t+c-1})$, based on Eq. (12) as follows:

$$\phi_i'(r^{ex}_{t:t+c-1}) = \sum_{k=1}^{i} \beta^{i-k} \phi_k(r^{ex}_{t:t+c-1}), \tag{13}$$

where parameter β is the high-level reward discount factor. In Eq. (13), each goal is assigned with the cumulative discounted external rewards from the beginning of period to the current sub-period, forcing each goal to improve the performance of whole period. When $\beta = 0$, it is equivalent to Eq. (12); when $\beta = 1$, all related rewards should be considered equally. The rewards are then assigned to HRA as the high-level reward for goal g^i, i.e., $r^h_i = \phi_i'(r^{ex}_{t:t+c-1})$.

3.3 Training Algorithm

In the proposed HRL based recommender framework, both HRA and LRA adopt the Actor-Critic architecture. Since both the high-level policy and the low-level policy in this framework are deterministic, we utilize DDPG [11] based algorithm to train the parameters.

In HRA, there exist M parallel high-level critic networks $Q^h_i, 1 < i < M$, which share the same encoder. We denote the parameters of critic network i as θ^h_i and the whole parameters for high-level critic network as Θ^h. Similarly, there also exist M parallel high-level policy networks $\pi^h_i, 1 < i < M$, which share the same state encoder. We denote the parameters of policy network i as ψ^h_i and the whole parameters of the high-level policy as Ψ^h. Then, the high-level critic networks can be trained by minimizing a series of loss functions as follows:

$$L(\theta^h_i) = E\Big[\Big(y^h_i - Q^h_i(s^h, g^i; \theta^h_i)\Big)^2\Big],\ 1 \le i \le M, \tag{14}$$

where $y^h_i = r^h_i + \gamma Q^h_i\big(s^{h\prime}, \pi^h(s^{h\prime}, \Psi^h_-); \Theta^h_-\big)$ is the high-level target value and Ψ^h_- (Θ^h_-) are the parameters of high-level target policy (critic) network. The parameters Ψ^h_- and Θ^h_- are fixed when optimizing the loss function $L(\theta^h_i)$. In this work, we adopt stochastic gradient descent to optimize loss functions. The derivatives of loss function $L(\theta^h_i)$ are represented as follows:

$$\nabla L(\theta^h_i) = E\big[\big(y^h_i - Q^h_i(s^h, g^i; \theta^h_i)\big)\ \nabla_{\theta^h_i} Q^h_i(s^h, g^i; \theta^h_i)\big]. \tag{15}$$

The high-level policy network is updated by applying the chain rule to the expected high-level return J^h from the start distribution with respect to the policy parameters [11], i.e.,

$$\nabla_{\psi^h_i} J^h = E\big[\ \nabla_{g^i} Q^h_i(s^h, g^i)\ \nabla_{\psi^h_i} \pi^h_i(s^h; \psi^h_i)\big], 1 \le i \le M, \tag{16}$$

where $g^i = \pi^h_i(s^h; \psi^h_i)$.

In LRA, we denote the parameters of low-level critic as Θ^l and the parameters of low-level policy network as Ψ^l. Similarly, the low-level critic can be trained by minimizing the loss function $L(\Theta^l)$ as:

$$L(\Theta^l) = E\big[\big(y^l - Q^l(s^l, a; \Theta^l)\big)^2\big], \tag{17}$$

where $y^l = r^l + \gamma Q^l\big(s^{l\prime}, \pi^l(s^{l\prime}, \Psi^l_-); \Theta^l_-\big)$ is the low-level target value and Ψ^l_- (Θ^l_-) represent parameters of the low-level target policy (critic) network. The derivatives of loss function $L(\Theta^l)$ with respect to parameters Θ^l are represented as follows:

$$\nabla\ L(\Theta^l) = E\big[\big(y^l - Q^l(s^l, a; \Theta^l)\big)\ \nabla_{\Theta^l} Q^l(s^l, a; \Theta^l)\big]. \tag{18}$$

Similarly, the low-level policy network is also updated by applying the chain rule to the expected low-level return J^l from the start distribution with respect to the low-policy parameters, i.e.,

$$\nabla_{\Psi^l} J^l = E\big[\ \nabla_{\hat{a}} Q^l(s^l, \hat{a})\ \nabla_{\Psi^l} \hat{\pi}^l(s^l; \Psi^l)\big], \tag{19}$$

where $\hat{a} = \hat{\pi}^l(s^l)$. Note that the action $\hat{a}$ is virtual action, instead of real action. This guarantees that policy gradient is taken at the actor network $\hat{\pi}^l$ [6].

Algorithm 1 Online Training Algorithm.

Initialize $\{\Theta^h, \Theta^l, \Psi^h, \Psi^l\}$ with random weights
Initialize target networks $\Theta^h_- \leftarrow \Theta^h$, $\Theta^l_- \leftarrow \Theta^l$, $\Psi^h_- \leftarrow \Psi^h$, $\Psi^l_- \leftarrow \Psi^l$
Initialize high-level buffer D^h and low-level buffer D^l
for *session* $\in [1, G]$ **do**
  Initialize clock $t \leftarrow 0$
  Receive initial high-level and low-level state s^h_t, s^l_t
  while $t < T$ **do**
    //Stage 1. Transition Generating Stage
    if $t \equiv 0 (mod\ c)$ **then**
      Generate a set of goals $g^{1:M}_t$ according to Eq. (4)
    else
      $g^{1:M}_t \leftarrow g^{1:M}_{t-1}$
    end if
    Select an action a_t according to the mapping in Eq. (9)
    Execute action a_t and observe external reward r^{ex}_t
    Update new high-level and low-level state s^h_{t+1}, s^l_{t+1}
    Store transition $(s^l_t, g^{1:M}_t, a_t, r^{ex}_t, s^l_{t+1})$ in D^l
    $t \leftarrow t + 1$
    if $t \equiv 0 (mod\ c)$ **then**
      Collect the recent c external rewards $r^{ex}_{t-c:t-1}$
      Store transition $(s^h_{t-c}, g^{1:M}_{t-c}, r^{ex}_{t-c:t-1}, s^h_t)$ in D^h
    end if
    //Stage 2. Parameter updating stage
    Sample mini-batch of N^h transitions from D^h
    Update high-level actor and critic based on Eq. (15)(16)
    Sample mini-batch of N^l transitions from D^l
    Update low-level actor and critic based on Eq. (18)(19)
    Update the target networks:
    $\Theta^h_- \leftarrow \tau\Theta^h + (1-\tau)\Theta^h_-,\ \Psi^h_- \leftarrow \tau\Psi^h + (1-\tau)\Psi^h_-$
    $\Theta^l_- \leftarrow \tau\Theta^l + (1-\tau)\Theta^l_-,\ \Psi^l_- \leftarrow \tau\Psi^l + (1-\tau)\Psi^l_-$
  end while
end for

The online training algorithm for the proposed HRL based recommender framework is presented in Algorithm 1. In the algorithm, several widely used techniques are introduced to train the framework, such as experience replay [12], separated evaluation and target networks [14], which can help smooth the learning and avoid the divergence of parameters.

4 EXPERIMENTS

In this session, we conduct a series of experiments with the datasets from a real e-commerce company so as to evaluate the effectiveness

of our proposed framework. We mainly focus on two questions: 1) how the proposed framework performs compared to representative baselines; and 2) how the components in the framework contribute to the performance.

4.1 Experiment Settings

4.1.1 Dataset. We evaluate our proposed algorithm on the dataset with users' behaviors from the front page of a real e-commerce App. Specifically, the training data consists of four-days users' behaviors log collected from Aug. 8th to Aug. 11th in 2018. The testing data consists of one-day data collected on Aug. 12th in the same year. Note that we filter the long tail sessions in both the training set and the testing set, which contain no click behaviors. The statistics about the datasets are shown in Table 1.

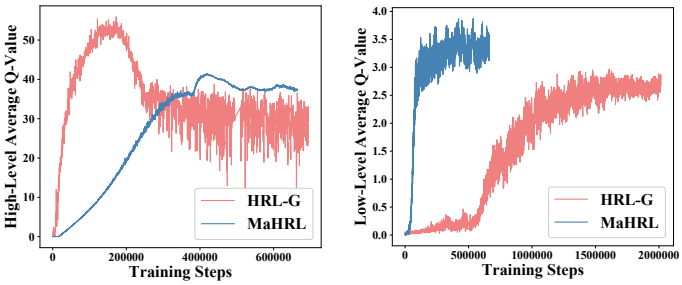
Table 1: Statistics on the dataset

Dataset	#Samples	#SKU	#Clicks	#Orders	#Sessions	Avg.Length
Train	8,596,852	553,156	843,249	46,022	184,838	46.51
Test	2,231,651	287,689	218,053	10,552	46,492	48.00

4.1.2 Baselines. To verify the effectiveness of our proposed MaHRL algorithm, we choose two types of baselines for comparison. The first is the classical methods in recommendation to show its overall performance and the second is the variants of MaHRL to show its ablation performance.

- **CF** [2]: Collaborative filtering is a classical method which recommends items to each user according to other users' behaviors with similar interest.
- **FM** [17]: Factorization Machines enhances the two-order interactions among features and can also deal with side information of users and items.
- **Wide&Deep** [4]: This baseline is the classical work of applying deep learning models to YouTube recommendations and widely deployed in many other companies.
- **GRU** [7]: This baseline treats users' behaviors as sequences and adopts the one-layer GRU model to abstract click and purchase interest in sequences.
- **DEERS** [31]: This baseline is the recent work applying deep reinforcement learning framework in industrial recommendation systems, which adopts the classical DQN framework.
- **DDPG+**: We apply the classical DDPG algorithm [11] in our recommendation system with suitable adjustment. This is in fact just the low-level component of our proposed method.
- **HRL-X**: This reflects a serious ablation baselines based on our MaHRL model. Here, X in $\{I, S, G\}$ stands for the ablation of different components, i.e., only click history as Input, guiding low-level agent via State instead of internal rewards, one Goal in high-level agent.

4.1.3 Parameter Settings. We adopt the pre-trained Telepath [24] model to generate the dense item embedding from image data. The dimension of the dense item embedding from Telepath is set as $n = 50$. In this work, we use $N = 10$ items for skipped/clicked/ordered history sequence to generate high-level and low-level states. The bound parameter B is set as $B = 1.94$ to match the bound of real action embedding. The external rewards r^{ex} of skipped/clicked/ordered actions are empirically set as 0, 1, and 5, respectively. And we set the discounted factor $\gamma = 0.95$. For the soft updates of target networks, we set $\tau = 0.01$. We use the Adam optimizer to minimize the losses, and the default learning rate is set to 10^{-6}. We set the influence factor as $\alpha = 0.5$, the number of goals for our MaHRL algorithm as $M = 2$, the high-level reward discount factor as $\beta = 1$ if without other specified. The default guiding step for all HRL methods is set as $c = 10$. The capacities of buffers for both high-level and low-level agent are set as 400, and we randomly extract 10% samples in buffers for parameter update at each time step. We design candidate set I in action mapping procedure as the subset of whole candidates with most relative items, the size of which is set $|I| = 1000$. For the parameters of the proposed framework, we select them via cross-validation. To construct a fair comparison, we also do parameter-tuning for baselines.



(a) High-level agent (b) Low-level agent

Figure 6: Training procedure.

4.2 Performance Comparison

We validate the performance of our proposed algorithm based on both online and offline evaluation. For the offline test, we select **MAP** [22] and **NDCG@20(40)** [9] as the metrics to measure the performance. For the online test, we leverage the average summations of different types of rewards in each session as the metrics, i.e., accumulated total rewards, click rewards and order rewards. The difference of ours from traditional Learn-to-Rank methods is that we rank both clicked and ordered items together, and set them by different rewards, rather than only rank clicked items as that in the Learn-to-Rank setting.

4.2.1 Training Convergence. We do training on the simulated online environment, which is trained on training set. It has the similar architecture with low-level critic, while the output layer is a softmax layer. We generate immediate feedback according to predicted click/order probability. We test the simulator on users' logs, and experimental results demonstrate that the simulated online environments has overall 90% precision for immediate feedback prediction task. This result suggests that the simulator can accurately simulate the real online environment, which enables us to train and test our model on it. During the training with the simulator, we evaluate the performance of training convergence via monitoring the average

Q-values in each batch. After training, we freeze the parameters of the model, and then do offline and online tests respectively.

Figure 6 illustrates the training convergence of the high-level and low-level agents in HRL-G and our proposed MaHRL algorithm. In Figure 6 (a), the high-level agent in HRL-G has an initially much too fast growth, and eventually falls back to the convergence position. The convergence of high-level agent in HRL-G is unstable and the average Q-value fluctuate dramatically. The main reason is that the low-level agent in HRL-G converges too slow, which can be verified in 6 (b). More specifically, the low-level agent in HRL-G converges extremely slow in the initial stage due to the difficulties of following the goals in high-level agent. After the convergence of high-level agent, the low-level agent gradually converge in a faster way. On the other hand, both the low-level agent and the high-level agent in our MaHRL algorithm converges fast and steadily. Our MaHRL converges around 85,000 steps, which is almost six times faster than HRL-G algorithm. This is because multiple goals greatly reduce the difficulty for the low-level agent to achieve the goal. In addition, the sharing mechanism improves the update speed and stability of the high-level agent in MaHRL. Figure 6 also demonstrate that our MaHRL has much higher average Q-value than HRL-G in both low-level agent and high-level agent. More specifically, the average Q-value of high-level agent in MaHRL on convergence state is around 38, much larger than 30 for that in HRL-G. Similar result happens in the low-level agent. The main reason is that the MaHRL has a larger accumulated rewards and thus better performance, which can be verified in the next subsection.

4.2.2 Offline Evaluation. When we do offline testing, we re-rank the items in each session from users' log on testing set. If the proposed framework works well, the clicked/ordered items in this session will be ranked at the top of the new list. For the well-trained Actor-Critic based reinforcement learning algorithms, e.g., DDPG+, we use critic network to re-rank the items in each session. For the hierarchical reinforcement learning algorithms, we adopt the low-level critic network to re-rank these items. For a new session, the initial high-level and low-level state are collected from the previous sessions of the user. To demonstrate the performance more clearly, we set the benchmark as the DEERS algorithm and calculate the gain of each algorithm based on it.

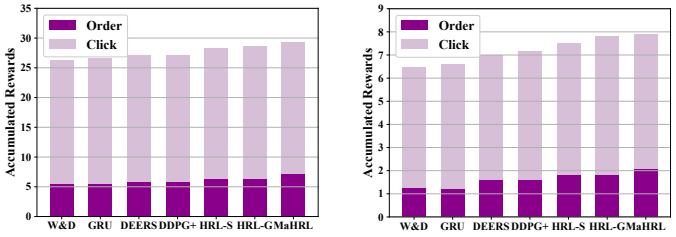
Table 2: Performance comparison for offline test.

Metrics	MAP	Gain	NDCG@20	Gain	NDCG@40	Gain
CF	0.2953	-9.78%	0.2156	-15.58%	0.2152	-15.74%
FM	0.2818	-13.90%	0.2011	-21.26%	0.2003	-21.57%
Wide&Deep	0.3110	-4.98%	0.2417	-5.36%	0.2426	-5.01%
GRU	0.3125	-4.52%	0.2425	-5.05%	0.2433	-4.74%
DEERS*	0.3273	–	0.2554	–	0.2554	–
DDPG+	0.3262	-0.34%	0.2497	-2.23%	0.2548	-0.23%
HRL-I	0.3087	-5.68%	0.2396	-6.19%	0.2407	-5.76%
HRL-S	0.3438	+5.04%	0.2587	+1.29%	0.2595	+1.61%
HRL-G	0.3477	+6.23%	0.2611	+2.23%	0.2624	+2.74%
MaHRL	**0.3534**	**+7.97%**	**0.2691**	**+5.36%**	**0.2715**	**+6.30%**

Table 2 demonstrates the overall offline performances of our proposed MaHRL and all baselines. We can have the following observations. First, our proposed hierarchical reinforcement learning based method MaHRL always outperforms the-state-of-art baselines on all of metrics. More specifically, our MaHRL have 7.97% improvement over DEERS on the metric of MAP, 5.36% improvement on the metric of NDCG@20, and 6.30% improvement on the metric of NDCG@40. Secondly, the RL-based methods usually have better performance than supervised methods. For instance, the performance of widely adopted supervised method Wide&Deep is 4.98%, 5.35%, and 5.01% lower than DEERS on the metrics of MAP, NDCG@20 and NDCG@40 accordingly. This is because supervised algorithms only consider immediate reward, while other RL-based baselines take long-term cumulative returns into account and thus achieving higher performance.

Table 2 also demonstrates the ablation experiment over different components of our proposed method. First, the elimination of impressions/orders input results in the most serious performance reduction. Specifically, compared with DEERS, HRL-I has 5.68% performance reduction on the metric of MAP while MaHRL has 6.23% improvement on that metric. One reason is that this results in information loss. What's more, the function of high-level agent and low-level in hierarchical reinforcement learning framework can not differentiate well, resulting in the invalidation of high-level agent. Secondly, guiding low-level agent directly via internal rewards have better performance than indirect guiding via merging the goal from high-level agent into input of the low-level agent as the state. For instance, HRL-S has a 4.17% performance reduction on the metric of NDCG@20 under the benchmark of DEERS. At last, multi-goals abstraction indeed bring performance improvement. Specifically, HRL-G has a 1.74%, 3.13%, and 3.56% performance reduction on the metric of MAP, NDCG@20 and NDCG@40 accordingly.

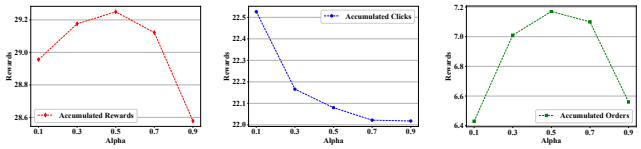
4.2.3 Online Evaluation. Due to the difficulty and huge cost of A/B test, we do online evaluation on the simulated online environment as [30, 31]. We build a similar simulator for online testing based on the testing data set. The noise can be also added to the simulator so as to imitate the uncertainty of users' preference. As the online test is based on the simulator, we can control the length of recommendation sessions to study the performance in short and long sessions. Here, we define short sessions with 50 recommendation items, while long sessions with 200 recommendation items. Figure



(a) Performance in long sessions. (b) Performance in short sessions.

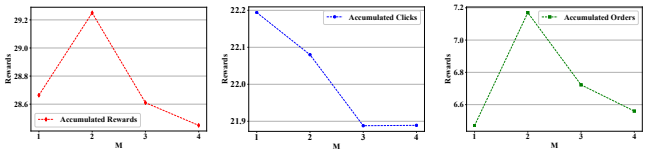
Figure 7: Performance comparison for online test.

7 shows the overall comparisons on the online test. We can have the following observations. First, the cumulative total rewards of MaHRL are significantly higher than all the-state-of-art baselines on online test. For instance, in the short sessions, the MaHRL have a



(a) Rewards (b) Clicks (c) Orders

Figure 8: Parameter sensitiveness of α.



(a) Rewards (b) Clicks (c) Orders

Figure 9: Parameter sensitiveness of M.

13.56% improvement over RL-based DEERS and 20.0% improvement over supervised based GRU model. Moreover, MaHRL algorithm has excellent performance especially on improving the performance of order action. Specifically, MaHRL has almost 30.62% improvement over DEERS on the order reward in the short sessions. Secondly, our MaHRL also has much better performance than the ablation baselines. For instance, HRL-S has a 2.96% reduction on the total rewards and 10.88% reduction on the ordered rewards. Thirdly, our proposed MaHRL has much better improvement on order rewards when the session is shorter. For instance, MaHRL improves 30.62% over DEERS on the order reward in the short session while has 22.88% improvement in the long session. This may be due to the reason that our MaHRL makes the order happen more in the front of recommendations. And this can be partially verified by the offline test on the metrics of NDCG. Fourthly, the RL-based methods also have better performance than supervised methods in both short sessions and long sessions. For instance, DEERS improves 3.10% over Wide&Deep on the cumulative total rewards and 5.52% on the order rewards.

4.3 Parameter Sensitivity

Our method has two key parameters: α controls the influence of internal reward and M controls the number of goals. To study the impact of these parameters, we investigate how the proposed framework works with the changes of one parameter, while fixing other parameters. Figure 8 shows the sensitivity of α in online recommendation task in long sessions. We have the following observations. Firstly, the performance for the recommendation achieves peak value when $\alpha = 0.5$, shown in Figure 8 (a). This indicates that both the high-level agent and the low-level agent in MaHRL contribute to its excellent performance. Secondly, as the increase of α, the click reward decreases. For instance, in the long session shown in Figure 8 (b), the click reward decreases from 22.5 to 22 as α increases from 0.1 to 0.9. This is consistent with our intuition that the larger α is, the less important the click signal is. Thirdly, the larger α may result in less order rewards. The optimal value for order rewards is also $\alpha = 0.5$. This indicates that suitable guidance of high-level agent benefits both the cumulative orders and total rewards.

Figure 9 shows the sensitivity of M in online recommendation task. First, the performance for the recommendation achieves peak when $M = 2$. This indicates that multi-goals indeed benefit the performance, but too many goals may cause the cumulative clicks to decrease. The intuition is that the larger number of goals increases the difficulties of model training and goals coordinating. Secondly, as the increase of M, the click reward decreases. For instance, in Figure 9 (b), the click reward decreases from almost 22.2 to around 21.9 as M increases from 1 to 4. Thirdly, the larger M may result in less order rewards. The optimal value for order rewards is also $M = 2$. This indicates that suitable number of goals in high-level agent benefits both the cumulative orders and total rewards.

5 RELATED WORK

The recommendation algorithms can be roughly divided into three categories: traditional methods, deep learning based methods and reinforcement learning based methods.

Firstly, traditional recommendation algorithms mainly include collaborative filtering [2], content-based filtering [15], both of which try to recommend items with similar properties to people with similar tastes [13]. Besides, hybrid methods are used more frequently, which are combined by several single techniques [3].

Secondly, deep learning based recommendation algorithms have become the current mainstream recommendation methods. Deep learning methods can help to learn item embedding from sequences, images or graph information [5]. They can also extract users' potential tastes [26], or improve the traditional methods directly [29]. Recently, DIEN [33] designed an interest extractor layer to capture temporal interest from historical behavior sequences.

Thirdly, reinforcement learning based recommendation algorithms are far more different from the above two categories. They model the recommendation procedure as the interaction sequences between users and recommendation agent, and leverage reinforcement learning to learn the optimal recommendation strategies. For instance, Li et al. [10] presented a contextual-bandit approach for personalized news article recommendation, in which a part of new items are exposed to balance exploration and exploitation. Zhao et al. [30, 32] proposed a novel page-wise recommendation framework based on reinforcement learning, which can optimize a page of items with proper display based on real-time feedback from users.

Hierarchical reinforcement learning (HRL) is dedicated to expanding and combining existing reinforcement learning methods to solve more complex and difficult problems [1, 20]. Recently, a goal-based hierarchical reinforcement learning framework [16, 23] has emerged, with high-level and low-level communicating through goals. However, different from games like ATARI, recommendation system has much huge and dynamic discrete action space and multi-objective rewards. In works of Vezhnevets et al. [23] and Nachum et al. [16], the policy of lower agent usually outputs the probability of each action under DDPG framework, which is not suitable for large-scale and dynamic discrete action space. As far as we know, only one recent work [28] applying deep hierarchical reinforcement learning to recommendations, but with explicit binary set $\{0, 1\}$ as

the action space of its high-level agent. In our work, we proposed a novel multi-goals abstraction based hierarchical reinforcement learning algorithm to solve these problems.

6 CONCLUSION

In this paper, we propose a novel multi-goals abstraction based hierarchical reinforcement learning algorithm for recommendations, which consists of two components, i.e., high-level agent and low-level agent. The high-level agent tries to catch long-term sparse conversion signals, and automatically sets abstract multi-goals for the low-level agent, while the low-level agent follows different goals in different sub-periods and interacts with real-time environment. The multiple high-level goals reduce the difficulty for the low-level agent to approach the high-level goals and accelerate the convergent rate of our proposed algorithm. The experimental results based on a real-world e-commerce dataset demonstrate the effectiveness of the proposed framework. There are several interesting research directions in the future. Firstly, the low-level agent can be guided in other ways, such as a hidden state representing the long-term preference. Secondly, the framework is general, and more specific information can be used to improve the performance in specific tasks, such as category information of items, user profiles, etc.

ACKNOWLEDGMENTS

This material is based upon work supported by, or in part by, the National Key R&D Program of China (No. 2019YFA0706401) and the National Natural Science Foundation of China (No. 61672264, No. 61632002, No. 61872166, No. 61872399, and No. 61902005).

REFERENCES

[1] Andrew G. Barto and Sridhar Mahadevan. 2003. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems* 13, 1-2 (2003), 41–77.
[2] John S Breese, David Heckerman, and Carl Kadie. 1998. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th. conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 43–52.
[3] Robin Burke. 2002. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction* 12, 4 (2002), 331–370.
[4] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu Google, and Hemal Shah. 2016. Wide & Deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 7–10.
[5] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for YouTube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 191–198.
[6] Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, TheophaneWeber, Thomas Degris, and Ben Coppin. 2015. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679* (2015).
[7] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).
[8] Yujing Hu, Qing Da, Anxiang Zeng, Yang Yu, and Yinghui Xu. 2018. Reinforcement learning to rank in e-Commerce search engine: formalization. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM.
[9] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 422–446.
[10] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World Wide Web*. ACM, 661–670.
[11] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).
[12] Long-Ji Lin. 1993. *Reinforcement learning for robots using neural networks*. Technical Report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science.
[13] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing* 7, 1 (2003), 76–80.
[14] 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).
[15] Raymond J Mooney and Loriene Roy. 2000. Content-based book recommending using learning for text categorization. In *Proceedings of the 5th ACM conference on Digital libraries*. ACM, 195–204.
[16] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. 2018. Data-efficient hierarchical reinforcement learning. In *Advances in neural information processing systems*.
[17] Steffen Rendle. 2010. Factorization machines. In *10th IEEE International Conference on Data Mining (ICDM)*. IEEE, 995–1000.
[18] Paul Resnick and Hal R Varian. 1997. Recommender systems. *Commun. ACM* 40, 3 (1997), 56–58.
[19] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2011. Introduction to recommender systems handbook. In *Recommender systems handbook*. Springer, 1–35.
[20] 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.
[21] Yumin Su, Liang Zhang, Quanyu Dai, Bo Zhang, Jinyao Yan, Dan Wang, Yongjun Bao, Sulong Xu, Yang He, and Weipeng Yan. 2020. An attention-based model for conversion rate prediction with delayed feedback via post-click calibration. In *International Joint Conference on Artificial Intelligence - Pacific Rim International Conference on Artificial Intelligence*.
[22] Andrew Turpin and Falk Scholer. 2006. User performance versus precision measures for simple search tasks. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 11–18.
[23] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. 2017. FeUdal networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1703.01161* (2017).
[24] Yu Wang, Jixing Xu, Aohan Wu, Mantian Li, Yang He, Jinghe Hu, and Weipeng P. Yan. 2018. Telepath: understanding users from a human vision perspective in large-scale recommender systems. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
[25] Yikai Wang, Liang Zhang, Quanyu Dai, Fuchun Sun, Bo Zhang, Yang He, Weipeng Yan, and Yongjun Bao. 2019. Regularized adversarial sampling and deep time-aware attention for click-through rate prediction. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. ACM, 349–358.
[26] Sai Wu, Weichao Ren, Chengchao Yu, Gang Chen, Dongxiang Zhang, and Jingbo Zhu. 2016. Personal recommendation using deep recurrent neural networks in NetEase. In *Data Engineering (ICDE), 2016 IEEE 32nd International Conference on Data Engineering*. IEEE, 1218–1229.
[27] Hongxia Yang, Quan Lu, Angus Xianen Qiu, and Chun Han. 2016. Large scale CVR prediction through dynamic transfer learning of global and local features. In *Proceedings of the 5th International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications at KDD 2016*, Vol. 53. PMLR, 103–119.
[28] Jing Zhang, Bowen Hao, Bo Chen, Cuiping Li, Hong Chen, and Jimeng Sund. 2019. Hierarchical reinforcement learning for course recommendation in MOOCs. *Psychology* 5, 4.64 (2019), 5–65.
[29] Shuai Zhang, Lina Yao, and Aixin Sun. 2017. Deep learning based recommender system: A survey and new perspectives. *arXiv preprint arXiv:1707.07435* (2017).
[30] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. 2018. Deep reinforcement learning for page-wise recommendations. *arXiv preprint arXiv:1805.02343* (2018).
[31] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. 2018. Recommendations with negative feedback via pairwise deep reinforcement learning. In *KDD'18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM.
[32] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Dawei Yin, Yihong Zhao, and Jiliang Tang. 2018. Deep reinforcement learning for list-wise recommendations. *arXiv preprint arXiv:1801.00209* (2018).
[33] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2018. Deep interest evolution network for click-through rate prediction. *arXiv preprint arXiv:1809.03672* (2018).