Multi-level Recommendation Reasoning over Knowledge Graphs with Reinforcement Learning

Xiting Wang
Microsoft Research Asia
Beijing, China
xitwan@microsoft.com

Kunpeng Liu
University of Central Florida
Florida, United States
kunpengliu@knights.ucf.edu

Dongjie Wang
University of Central Florida
Florida, United States
wangdongjie@knights.ucf.edu

Le Wu
HeFei University of Technology
Hefei, China
lewu@hfut.edu.cn

Yanjie Fu*
University of Central Florida
Florida, United States
yanjie.fu@ucf.edu

Xing Xie
Microsoft Research Asia
Beijing, China
xing.xie@microsoft.com

ABSTRACT

Knowledge graphs (KGs) have been widely used to improve recommendation accuracy. The multi-hop paths on KGs also enable recommendation reasoning, which is considered a crystal type of explainability. In this paper, we propose a reinforcement learning framework for multi-level recommendation reasoning over KGs, which leverages both ontology-view and instance-view KGs to model multi-level user interests. This framework ensures convergence to a more satisfying solution by effectively transferring high-level knowledge to lower levels. Based on the framework, we propose a multi-level reasoning path extraction method, which automatically selects between high-level concepts and low-level ones to form reasoning paths that better reveal user interests. Experiments on three datasets demonstrate the effectiveness of our method.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → **Reinforcement learning**.

KEYWORDS

Recommendation Reasoning; Knowledge Graphs; Explainability

ACM Reference Format:
Xiting Wang, Kunpeng Liu, Dongjie Wang, Le Wu, Yanjie Fu, and Xing Xie. 2022. Multi-level Recommendation Reasoning over Knowledge Graphs with Reinforcement Learning. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3485447.3512083

1 INTRODUCTION

Personalized recommendation has become one of the key mechanisms for handling the explosive growth of online content. Recently, increasing attention has been paid to equipping recommender systems with the ability to leverage knowledge graphs (KGs). A core benefit of KGs is that they enable the modeling of explicit multi-hop relations between users and items. For example, KGs may reveal that user *Alice* and item *Mania Fragrance* has a multi-hop relation *Alice* $\xrightarrow{Purchase}$ *Luminous Silk Foundation* $\xrightarrow{Produced\_by}$ *Armani* $\xrightarrow{Produce}$ *Mania Fragrance*. Such multi-hop relational paths provide auxiliary information about user-item relations and help improve recommendation **accuracy**. Moreover, they also enable a crystal type of **explainability** that is known as reasoning [13]. Compared with traditional recommendation models, recommendation reasoning methods not only output recommended items, but also provide multi-hop paths that connect users with their recommended items to help understand the model and increase user trust [27, 30].

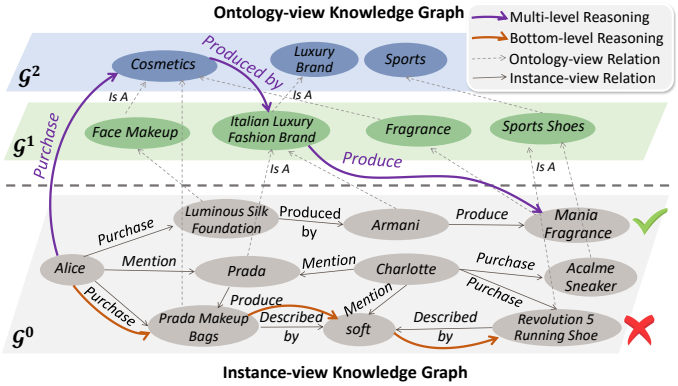


Figure 1: Multi-level reasoning over two-view KGs.

Existing works on recommendation reasoning focus on **instance-view** KGs [11, 13, 27, 30, 36]. As shown in Fig. 1, an instance-view KG includes relations between *specific entities*, e.g., *Armani*, *Produce*, *Mania Fragrance*. It lacks information about high-level connections between entities, e.g., both *Armani* and *Prada* are *Italian Luxury Fashion Brands*. Based on instance-view KGs, existing methods can perform detailed analysis about user-item relations and identify **bottom-level reasoning** paths. However, bottom-level reasoning is fragmented in the sense that it cannot directly relate different user behaviors. An example is given in Fig. 1. If we relate different behaviors of *Alice* by considering the high-level connections between entities ($\mathcal{G}^1$ and $\mathcal{G}^2$), we can see that *Alice* tends to purchase *Cosmetics* produced by *Italian Luxury Fashion Brands*. However,

*Contact 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 ACM 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.

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France

© 2022 Association for Computing Machinery.

ACM ISBN 978-1-4503-9096-5/22/04…$15.00
https://doi.org/10.1145/3485447.3512083

it is difficult for existing methods to identify this overall pattern, since they only model bottom-level relations ($\mathcal{G}^0$). This leads to two issues. First, the recommendation accuracy may be hampered. Without a correct overall understanding of the user behaviors, the fragmented bottom-level reasoning process has a larger probability to converge to a local optimum, e.g., recommending an item that belongs to the category *Sports* to *Alice* (Fig. 1, red curve). Second, the explainability of existing methods is limited, as a bottom-level reasoning path may fail to reveal the true interest of a user (e.g., *Italian Luxury Fashion Brand*) because they can only focus on a few specific, bottom-level entities (e.g., *Armani*).

To address the issues, we take a more complete view of KGs. In addition to instance-view KGs, the other major type is **ontology-view** KGs [8], which stores relations that reveal whether an entity or high-level *concept* is a child of another concept (e.g., *Prada*, *Is A*, *Italian Luxury Fashion Brand* in Fig. 1). Combining ontology-view KGs with instance-view KGs allow us to achieve **multi-level reasoning** over KGs. We refer to a recommendation model as achieving multi-level reasoning if it outputs the recommended items and their reasoning paths by considering multi-level knowledge about entities. According to psychopathology, human reasoning is by nature a multi-level information processing procedure [9]. We wish to operationalize this insight into machine learning by identifying the desirable properties (P1 and P2) for a multi-level reasoning model through reflecting on the human reasoning process.

In particular, humans usually perform reasoning by obtaining an overview first and then drilling into the details on demand [20]. For example, humans may first consider *Alice*'s interest with respect to high-level categories, e.g., whether she likes more about *Cosmetics* or *Sports*. After identifying that *Alice* likes more about *Cosmetics*, humans may then consider her interest with respect to detailed categories in *Cosmetics*. In recommendation reasoning, adopting such a top-down strategy can help prune the large search space, avoid local minimum, and converge to a more satisfying solution by considering the overall user behaviors (P1). Moreover, human reasoning and intention inherently have multiple levels of granularities. Some people may like most *Luxury Brand*s while some may have a specific preference for *Armani*. Thus, extracting multi-level reasoning paths that contain concepts from multiple levels (Fig. 1, purple curve) provides the flexibility that is essential for correctly representing user preference, which enhances both recommendation accuracy and explainability (P2).

Designing a multi-level reasoning model with desirable properties P1 and P2 is challenging. It has been pointed out that the traditional supervised learning schema is not suitable for finding reasoning paths [30, 36]. This is because 1) there lack ground-truth labels for reasoning paths; and 2) simply enumerating all possible reasoning paths to find the best one is infeasible for large real-world KGs. To address the problem, it is essential to incorporate reinforcement learning (RL), which enables policy-guided path searching in the large action space [30, 36]. For single-level reasoning, it is clear now how we can define the Markov Decision Process (MDP) in RL. However, for multi-level reasoning, how to formulate the problem in the RL setting while simultaneously satisfying P1 and P2 remains unclear. In this paper, we study this research problem and make the following technical contributions.

- We propose a Reinforcement learning framework for Multi-level recommendation Reasoning (ReMR). We show that multi-level reasoning can be formulated based on the abstract MDP [12]. Although we focus on recommendation reasoning, our framework can generalize to many other reasoning tasks over graphs.
- We design a Cascading Actor-Critic, which adopts a top-down strategy to prune the search space and ensures that knowledge from the high-level KGs can help guide the low-level reasoning policies to converge to a more satisfying solution (P1).
- We propose a multi-level reasoning path extraction algorithm that automatically selects which level of concepts should be used in each hop of the reasoning paths. This helps reveal the true level of user interest and improves accuracy and explainability (P2).

2 PRELIMINARY

In this section, we clarify some important terminologies and present our problem. A notation table is given in the supplement.

2.1 Multi-level Knowledge Graph

We are interested in leveraging both instance-view and ontology-view KGs for recommendation reasoning. Combining these two types of KGs naturally forms a multi-level KG $\mathcal{G}^{0\sim L} = \{\mathcal{G}^0, ...\mathcal{G}^L\}$ with L+1 levels, as shown in Fig. 1. More specifically:

$\mathcal{G}^0$ at the bottom is the **instance-view KG**, which contains specific entities and their in-between relations. We treat entities as level 0 concepts, and denote the entity set as C^0. The relation set is Υ. Each triplet in $\mathcal{G}^0$ is represented by $c_i^0 \xrightarrow{r_j} c_k^0$, where $c_i^0 \in C^0$ is a head entity (e.g., *Armani*), $r_j \in \Upsilon$ is a relation (e.g., *Produce*), and $c_k^0 \in C^0$ is the tail entity (e.g., *Mania Fragrance*). The items to be recommended are part of C^0. To facilitate extracting multi-hop reasoning paths between users and items, we follow existing works [30, 36] to add users and their interactions with the entities into $\mathcal{G}^0$.

$\mathcal{G}^l$ with $l > 0$ is a part of the **ontology-view KG**. To build $\mathcal{G}^l$, we leverage the widely-used Microsoft Concept Graph[1] [28, 29], which contains over 85 million *Is A* relations that illustrate whether a low-level concept is a child of the high-level concept (e.g., *Prada*, *Is A*, *Italian Luxury Fashion Brand*). We use $r_{\text{IsA}} \in \Upsilon$ to denote the *Is A* relation. Low-level concepts C^{l-1} are searched in Microsoft Concept Graph and their parents are extracted, which form triplets in the form of $c_i^{l-1} \xrightarrow{r_{\text{IsA}}} c_j^l$. To better understand the items, the item categories in the recommendation datasets are added as parents of the items. For simplicity, we only consider the top-1 parent of each concept in this paper. How to reason over ontology-view KGs with multiple parents and other relations is discussed in the supplement.

2.2 Reinforcement Learning and MDP

While supervised learning requires ground-truth labels, reinforcement learning (RL) provides an additional learning paradigm, in which an intelligent agent optimizes its behaviors by interacting with the environment. The interaction between the agent and the environment is as follows: 1) the environment informs the agent about the current **state**; 2) the agent takes an **action** based on the state by using a function called the **policy**; 3) the environment outputs a **reward** based on the action, together with the updated

[1] https://concept.research.microsoft.com/

state, which help the agent to optimize the policy. The goal of the agent is to learn a policy that results in the maximum accumulated reward. In deep RL, the policy is modeled by using a neural network. The aforementioned process repeats until the policy converges or a maximum number of iterations has been reached.

The environment in RL is typically formulated in the form of Markov Decision Processes (MDPs). An MDP is defined by $\mathbf{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}\}$. Here, $\mathcal{S}$ and $\mathcal{A}$ are state space and action space. The reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ maps a state-action pair to a scalar, and $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is the probability to transit from a state-action pair to the next state. Given $\mathbf{M}$, RL methods can be used to find a balance between *exploration* of unknown search space and *exploitation* of the current policy.

2.3 Single-level Reasoning with MDP

Our multi-level reasoning framework is designed by extending the existing RL framework for single-level reasoning [30, 36], which reasons over $\mathcal{G}^0$ by defining an MDP $\mathbf{M}^0 = \{\mathcal{S}^0, \mathcal{A}, \mathcal{R}^0, \mathcal{P}^0\}$:

State $s_t^0 \in \mathcal{S}^0$ reflects the status of the path reasoning process at time t. As all paths start from a given user, the initial state s_0^0 is the user u. At time t, the state is defined by the user as well as the latest found entities and relations in the reasoning path: $s_t^0 = (u, c_{t-K}^0, r_{t-K+1} \ldots, c_{t-1}^0, r_t, c_t^0)$, where K is a predefined number.

Action a_t at time t denotes the selected next hop in the reasoning path given the current state s_t^0. Formally, if in the selected next hop, the outgoing relation is r_{t+1} and the corresponding entity is c_{t+1}^0, then $a_t = (r_{t+1}, c_{t+1}^0)$. The action space $\mathcal{A}_t$ only includes the outgoing edges and entities that are connected to c_t^0 in the KG. Entities that are already on the path are removed from $\mathcal{A}_t$, to prevent the reasoning path from going backward. The policy is terminated after it takes T actions. By adding self-loop relation r_{Self} that connects each entity to itself and removing it from the final reasoning paths, we can search all paths with $\leq T$ hops by taking T actions.

Reward function $\mathcal{R}^0$ measures whether the reasoning path ends with an item that the user likes. A terminal reward is usually used, which is only non-zero when the policy has terminated (i.e., $t = T$). Specifically, the terminal reward of a path is 1 if it ends with an item that u interacts with and is 0 otherwise.

Transition function $\mathcal{P}^0$ describes the probability of the next state s_{t+1}^0based on s_t^0 and a_t. A deterministic MDP is usually used, which means that $\mathcal{P}^0(s_t^0, a_t, s_{t+1}^0) \equiv 1$.

After defining the MDP, the expected reward can be maximized by using RL methods like policy gradient [30] or actor-critic [36].

2.4 Multi-level Reasoning Problem Statement

Given a multi-level KG $\mathcal{G}^{0\sim L}$ and a user u, multi-level recommendation reasoning outputs:

- A set of recommended items $V_u \subset V$, where V is the item set.
- A multi-level reasoning path $\tau_u^{0\sim L}$ for each recommended item $v_u \in V_u$. Here, $\tau_u^{0\sim L} = (u, r_1, c_1^{0\sim L} \cdots, r_T, c_T^{0\sim L} = v_u)$, where each $c_t^{0\sim L} \in \cup_l C^l$ is a concept that may belong to any level of the KG.

3 METHODOLOGY

In this section, we propose our Reinforcement learning framework for Multi-level recommendation Reasoning over KGs (ReMR), which is formulated with **abstract MDPs** (Sec. 3.1). Based on the framework, we design a **Cascading Actor-Critic** (Sec. 3.2), which employs a top-down strategy to prune the search space and ensures that knowledge from the high levels can help guide the low-level reasoning policy to converge to a more accurate solution (P1). Finally, we propose our **multi-level reasoning path extraction** algorithm (Sec. 3.3), which outputs recommended items V_u and multi-level reasoning paths $\tau_u^{0\sim L}$ by automatically selecting which level of concepts should be used in each hop of reasoning in order to better represent user interest (P2).

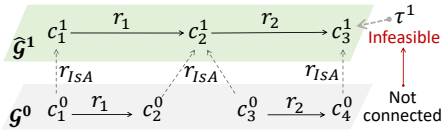


Figure 2: Illustration of infeasible reasoning paths. A path is infeasible if there exist no bottom-level paths to support it.

3.1 Formulation with Abstract MDPs

3.1.1 Issues of Straightforward Formulations. The most straightforward way to reason over a multi-level KG $\mathcal{G}^{0\sim L}$ is to collapse the multi-level graph into a single-level one by treating the high-level relation r_{IsA} in the same way as other bottom-level relations. Then, we can directly use existing single-level reasoning methods to solve the problem. However, this method can no longer distinguish high-level concepts from low-level ones, which makes it impossible to satisfy P1 or P2. Moreover, the extracted paths may be difficult to understand, as it can contain much redundant information, e.g., both the child, parent, and ancestors exist in the same path.

Another straightforward formulation is to define an MDP for each level of the KG. Specifically, we can propagate the bottom-level relations in $\mathcal{G}^0$ to higher level $\mathcal{G}^l (l > 0)$ through r_{IsA} as shown in Fig. 2. In this way, we obtain $\hat{\mathcal{G}}^l$, in which high-level concepts are connected based on their bottom-level relations. We can then extract high-level reasoning paths (e.g., $\tau^1 = c_1^1 \xrightarrow{r_1} c_2^1 \xrightarrow{r_2} c_3^1$) over $\hat{\mathcal{G}}^l$ by defining an MDP for level l with single-level reasoning methods.

While this method can well distinguish KGs at different levels, it is unclear how we can transfer knowledge between levels to fulfill P1 and how to select the correct level in each reasoning hop to satisfy P2. Moreover, this method may extract **infeasible reasoning paths**. As shown in Fig. 2, we say that a high-level path is infeasible if there does not exist a bottom-level path that can be mapped to the high-level path through r_{IsA}. Since there are no bottom-level paths to support it, an infeasible path usually does not have a clear and coherent meaning, and it may even mislead people into believing something that is not true. For example, τ^1 in Fig. 2 could be *Italian Luxury Fashion Brand* $\xrightarrow{Produce}$ *Face Makeup* $\xrightarrow{Described\_by}$ *Destructive Substance*. This path may make people think that an Italian luxury brand produces some bad face makeup, but in fact, τ^1 may be extracted only because *Armani* (c_1^0) produces a face makeup (c_2^0), and another unrelated face makeup (c_3^0) is described by *Poison* (c_4^0).

3.1.2 Our Formulation. To avoid extracting infeasible paths, it is essential that we reason over high-level KGs with the bottom-level one in mind. Based on this observation, we propose to formulate

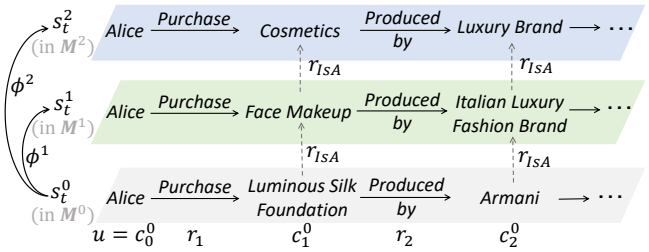


Figure 3: State abstraction function $\phi^l(l > 0)$ maps a bottom-level reasoning path represented by s_t^0 to its high-level path.

the high-level reasoning processes with abstract MDPs [12], which is defined based on a *ground MDP* (in our case the MDP of $\mathcal{G}^0$). Abstract MDPs also provide a theoretical foundation that enables us to 1) prove no infeasible paths will be extracted; and 2) show that the reasoning policies learned at higher levels can effectively guide the policy at low levels (Sec. 3.1.3).

In our framework, we define the MDP of $\mathcal{G}^0$ as $\mathbf{M}^0$ by using the method introduced in Sec. 2.3. We then formulate the reasoning process at each higher level l ($l > 0$) as an abstract MDP $\mathbf{M}^l = \{\mathcal{S}^l, \mathcal{A}, \mathcal{R}^l, \mathcal{P}^l\}$ of the ground MDP $\mathbf{M}^0 = \{\mathcal{S}^0, \mathcal{A}, \mathcal{R}^0, \mathcal{P}^0\}$. An abstract MDP $\mathbf{M}^l$ is an MDP created by aggregating multiple states in $\mathbf{M}^0$ into one state. It is defined by the ground MDP $\mathbf{M}^0$, the state abstraction function ϕ^l, and the weighting function ω^l.

The **state abstraction function** $\phi^l : \mathcal{S}^0 \to \mathcal{S}^l$ maps a ground state in $\mathbf{M}^0$ to an abstract state in $\mathbf{M}^l$. In our scenario, a nature choice of ϕ^l is the one that maps a bottom-level path represented by s_t^0 to its corresponding high-level path, as shown in Fig. 3. In this way, selecting the most valuable abstract states is equivalent to extracting high-level reasoning paths. Formally,

$$\phi^l(s_t^0) = (u, \varphi^l(c_{t-K}^0), r_{t-K+1} \ldots, \varphi^l(c_{t-1}^0), r_t, \varphi^l(c_t^0)) \quad (1)$$

where φ^l is a function that seeks the level l ancestor of a concept by jumping through r_{IsA} l times. If the concept does not have a parent (e.g., c_t^0 is a user), then φ^l maps the concept to itself ($\varphi^l(c_t^0) = c_t^0$).

Different from the method in Fig. 2, which maps each hop (e.g., $c_1^0 \xrightarrow{r_1} c_2^0$) to a higher level independently, ϕ^l maps a multi-hop path (e.g., $c_0^0 \xrightarrow{r_1} c_1^0 \xrightarrow{r_2} c_2^0$) to a higher level. Defining ϕ^l in this way ensures that the extracted high-level paths are all feasible (Theorem 1). Another difference between the two methods is the action space. Although the states of $\mathbf{M}^l$ are abstract, its action space $\mathcal{A}$ is the same with $\mathbf{M}^0$. This means that its policy $\pi^l : \mathcal{S}^l \mapsto \mathcal{A}$ can select entities instead of just concepts. This fine-grained action space of π^l allows it to build a clear connection with low-level policies, which enables effective transfer of knowledge (Lemma 1).

The **weighting function** $\omega^l : S^0 \to [0, 1]$ decides the relative weights of the ground states when they aggregate into one abstract state in $\mathbf{M}^l$. Given ω^l, ϕ^l, and $\mathbf{M}^0$, the abstract MDP $\mathbf{M}^l$ can be fully determined, and its reward and transition functions $\mathcal{R}^l$ and $\mathcal{P}^l$ are:

$$\mathcal{R}^l(s_t^l, a_t) = \sum_{s_t^0 \in \phi^{-l}(s_t^l)} \omega^l(s_t^0)\, \mathcal{R}^0(s_t^0, a_t), \quad (2)$$

$$\mathcal{P}^l(s_t^l, a_t, s_{t+1}^l) = \sum_{s_t^0 \in \phi^{-l}(s_t^l)} \sum_{s_{t+1}^0 \in \phi^{-l}(s_{t+1}^l)} \omega^l(s_t^0)\, \mathcal{P}^0(s_t^0, a_t, s_{t+1}^0) \quad (3)$$

where $\phi^{-l}(s_t^l)$ denotes the set of ground states corresponding to s_t^l.

In theory, we may use any fixed ω^l (Lemma 1). However, in practice, we need to define ω^l so that we can easily obtain s_{t+1}^l according to $\mathcal{P}^l$ in Eq. (3). $\mathcal{P}^l$ is very difficult to explicitly compute and store, due to the large state and action space caused by large real-world KGs. Thus, we propose to sample s_{t+1}^l without explicitly computing $\mathcal{P}^l$. This is achieved by defining ω^l based on a sampling policy. Specifically, given any bottom-level policy $\tilde{\pi}^0 : \mathcal{S}^0 \mapsto \mathcal{A}$, we sample states according to $\mathcal{P}^0$ and $\tilde{\pi}^0$, and determine the weight of each state s_t^0 based on the frequency it appears. Mathematically,

$$\omega^l(s_t^0) = E_{\mathcal{P}^0, \tilde{\pi}^0}\, N(s_t^0) / E_{\mathcal{P}^0, \tilde{\pi}^0}\, N(\phi^l(s_t^0)) \quad (4)$$

where $E_{\mathcal{P}^0, \tilde{\pi}^0}\, N(\cdot)$ denotes the expected number of times some state or abstract state appears when we sample states iteratively according to $s_{t+1}^0 \sim \mathcal{P}^0(s_t^0, a_t, s_{t+1}^0)$ and $a_t \sim \tilde{\pi}^0(a_t|s_t^0)$. We say that an abstract state appears if any of its bottom-level state appears.

Defining ω^l with Eq. (4) allows us to optimize the high-level policy without having to explicitly calculate $\mathcal{P}^l$ or $\mathcal{R}^l$. Specifically, we can sample high-level states s_{t+1}^l with the following two steps, and then the transition from s_t^l to s_{t+1}^l naturally satisfies Eq. (3): 1) sampling s_{t+1}^0 according to $\mathcal{P}^0$ and $\tilde{\pi}^0$; 2) mapping s_{t+1}^0 to s_{t+1}^l with ϕ^l. Moreover, the expectation of $\mathcal{R}^0(s_t^0, a_t)$ is equal to $\mathcal{R}^l(s_t^l, a_t)$ in Eq. (2). This means that we can obtain sample $(s_t^0, a_t, s_{t+1}, r_t^0 = \mathcal{R}^0(s_t^0, a_t))$ at the bottom level and use $(\phi^l(s_t^0), a_t, \phi^l(s_{t+1}), r_t^0)$ to update the policy at level l. Compared with explicitly computing and storing $\mathcal{P}^l$ and $\mathcal{R}^l$, which is infeasible for large real-world KGs, this method is both feasible and easy to implement.

Eq. (2) shows that $\mathbf{M}^l$ aggregates the rewards of bottom-level paths into their high-level path. This enables explicitly relating different bottom-level paths to achieve an overall understanding.

3.1.3 Theoretical Properties. Abstract MDPs enable transferring knowledge between levels and the extraction of feasible paths.

Transferring knowledge. We can naturally translate a high-level policy π^l into a ground policy π^0 with $\pi^0(a_t|s_t^0) = \pi^l(a_t|\phi^l(s_0^t))$, and there is an inherent relation between the optimal policy π^l in $\mathbf{M}^l$ and the optimal policy in $\mathbf{M}^0$ [12]:

Lemma 1. Given any fixed ω^l, the optimal abstract policy π^l of $\mathbf{M}^l$ is optimal in the ground MDP $\mathbf{M}^0$ when ϕ^l is properly designed.

The lemma shows that the optimal policy derived by solving a simpler abstract MDP with fewer number of states enables us to learn about the optimal policy in the more complicated ground MDP. This provides the basis for developing a top-down procedure that effectively prunes the search space and transfers knowledge.

Feasibility of extracted paths. The following theorem illustrates that abstract MDP can naturally guarantee path feasibility when ϕ^l and ω^l are defined according to Eqs. (1)(4).

Theorem 1. Suppose that $\tau_{u,t}^l = (u, r_1, c_1^l, \cdots r_t, c_t^l)$ is a reasoning path and $s_t^l = (u, c_{t-K}^l, r_{t-K+1} \ldots, c_{t-1}^l, r_t, c_t^l)$ is its corresponding state. If s_t^l can be sampled from $\mathbf{M}^l$, i.e., there exist s_{t-1}^l and a_{t-1} such that $\mathcal{P}^l(s_{t-1}^l, a_{t-1}, s_t^l) > 0$, then $\tau_{u,t}^l$ must be feasible.

The proof for Theorem 1 is given in the supplementary material.

Handling inaccurate high-level knowledge. Summarizing that *Alice* likes *Cosmetics* based on several purchase behaviors may not be accurate. This issue of inaccurate high-level knowledge can be naturally solved by our method. Interested readers are referred to the supplement for more details.

3.2 Cascading Actor-Critic

We design a Cascading Actor-Critic to learn the policies of $\mathbf{M}^L, \mathbf{M}^{L-1}$ ...,$\mathbf{M}^0$. The basic idea is to first obtain a more global understanding of the user by learning high-level policies, and then gradually fine-tune the high-level policies at lower levels to consider more detailed information. Accordingly, we adopt a top-down strategy as shown in Fig. 4. In this design, we first learn the top-level policy π^L by solving $\mathbf{M}^L$. The learned knowledge (e.g., π^L) is then passed to $L-1$, so that we can effectively prune the search space when learning π^{L-1} and ensure that it can converge to a more satisfying solution. This process repeats and we obtain $\pi^{L-1}, \pi^{L-2}..., \pi^0$ sequentially.

3.2.1 Actor-critic learning at the top level ($l = L$). Learning at the top level is simpler than other levels as it does not require leveraging the knowledge from the upper levels. The key question is: how does one learn a policy for an *abstract* MDP? Recall that by defining ω^l with Eq. (4), we can directly obtain samples of an abstract MDP. Thus, we can use traditional RL methods to solve abstract MDPs, and the only difference is that, during optimization, we need to obtain a high-level sample $(s_t^l, a_t, s_{t+1}^l, r_{t+1}^l)$ by first sampling at the bottom level, and then mapping the bottom-level sample to level l with ϕ^l. We utilize the RL method actor-critic [6] because it is effective in path reasoning [36] and facilitates transferring high-level knowledge (Sec. 3.2.2). Next, we introduce the network structures of the actor and the critic, as well as the optimization method.

Actor π^L. The actor (or policy network) outputs the probability of taking each action based on the state. Following [30], we design the actor so that it contains three-layer feed-forward networks, uses Exponential Linear Unit (ELU) as the activation function, and leverages dropout to prevent overfitting. Specifically:

$$\mathbf{x} = dropout(ELU(dropout(ELU(\mathbf{s}_t^L \mathbf{W}_1))\mathbf{W}_2)) \tag{5}$$

$$\pi^L(\cdot|s_t^L) = softmax(\mathbf{A}_t \odot (\mathbf{x}\mathbf{W}_a)) \tag{6}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d_1 \times d_2}$, $\mathbf{W}_2 \in \mathbb{R}^{d_2 \times d_3}$, and $\mathbf{W}_a \in \mathbb{R}^{d_3 \times d_a}$ are parameters to be be learned. $\mathbf{s}_t^L$ is the embedding vector of s_t^L and is computed by concatenating the embeddings of all concepts, entities, and relations in s_t^L. The embeddings are learned by using JOIE [8], which is a state-of-the-art KG embedding method that jointly encodes the ontology and instance views. $\odot$ denotes element-wise product, and $\mathbf{A}_t$ is a binary vector leveraged to select only the actions in the action space $\mathcal{A}_t$. Specifically, the i-th entry in $\mathbf{A}_t$ is 1 if the i-th action is in $\mathcal{A}^t$ and is 0 otherwise. We follow Xian et al. [30] to prune the action space and keep at most N_m candidate actions.

Critic V^L. The critic (or value network) outputs the expected reward that π^L will obtain given s_t^L. The critic shares the first two layers with the actor, and differs in the output layer:

$$V^L(s_t^L) = \mathbf{x}\mathbf{W}_c \tag{7}$$

where $\mathbf{W}_c \in \mathbb{R}^{d_3 \times 1}$ denotes parameters to be learned.

Optimization. At each iteration, we obtain sample $(s_t^0, a_t, s_{t+1}^0, \mathcal{R}^0(s_t^0, a_t))$ from $\mathbf{M}^0$ according to $\mathcal{P}^0$ and $\tilde{\pi}^0$, where $\tilde{\pi}^0(a_t|s_t^0) = \pi^L(a_t|\phi^L(s_t^0))$. By mapping the sample to level L, we get a high-level sample $(\phi^L(s_t^0), a_t, \phi^L(s_{t+1}^0), \mathcal{R}^0(s_t^0, a_t))$, which is then used to update the actor and critic. Specifically, we learn the actor and critic by minimizing $J_{\pi^L} + J_{V^L}$:

$$J_{\pi^L} = -log\pi^L(a_t|\phi^L(s_t^0))\ V^L(\phi^L(s_t^0)) \tag{8}$$

$$J_{V^L} = ||\mathcal{R}^0(s_t^0, a_t) + V^L(\phi^L(s_{t+1}^0)) - V^L(\phi^L(s_t^0))||^2 \tag{9}$$

Interpretation. By minimizing J_{V^L}, the critic V^L learns to estimate the value (expected reward) of the high-level states. By minimizing J_{π^L}, the actor increases the probability of actions that lead to more valuable states. We can see from Eq. (9) that the reward of a bottom-level state will impact the value of its ancestors at higher levels. For example, if we find that *Alice* likes a certain makeup at the bottom level, the high-level states with *Cosmetics* will be rewarded, and π^L will explore more on other *Cosmetics*.

3.2.2 Actor-critic learning at lower levels ($l < L$). The actors and critics at lower levels adopt the same network structures with that of π^L and V^L. The key question is how we can learn π^l and V^l so that the knowledge of level l+1 is effectively leveraged. Our solution is to incorporate behavior cloning and reward shaping (Fig. 4).

Behavior cloning. According to Lemma 1, the optimal policy in the abstract MDP is closely related to the optimal policy in the ground MDP. Let us consider $\mathbf{M}^l$ as the ground MDP, and $\mathbf{M}^{l+1}$ as its abstract MDP with the abstraction state function ϕ^1. We can then see that initializing the ground policy π^l with π^{l+1} is beneficial according to Lemma 1. Thus, we initialize π^l by cloning the behavior [18] of π^{l+1}, i.e., we generate paths by using π^{l+1}, and treat these paths as ground-truth labels for training the initial π^l. π^l is then refined based on the critic in the same way of π^L (Eq. (8)).

Reward shaping. To better guide V^l, we use reward shaping [15], which is robust and has good theoretical guarantees. Reward shaping is the procedure in which we add additional rewards to guide the learning process, beyond the sparse rewards supplied by the original MDP. Specifically, to solve $\mathbf{M}^l = \{\mathcal{S}^l, \mathcal{A}, \mathcal{R}^l, \mathcal{P}^l\}$, we create a shaped MDP $\hat{\mathbf{M}}^l = \{\mathcal{S}^l, \mathcal{A}, \mathcal{R}^l + \mathcal{F}^l, \mathcal{P}^l\}$, in which the shaping function $\mathcal{F}^l$ is defined based on V^{l+1}:

$$\mathcal{F}^l(s_t^l, s_{t+1}^l) = V^{l+1}(\phi^{l+1}(s_{t+1}^0)) - V^{l+1}(\phi^{l+1}(s_t^0)) \tag{10}$$

where $s_t^l = \phi^l(s_t^0)$ and $s_{t+1}^l = \phi^l(s_{t+1}^0)$. It has been proved that, such value-function-based reward shaping enables faster convergence, while a near optimal policy learned by solving $\hat{\mathbf{M}}^l$ is still near optimal in $\mathbf{M}^l$ [15]. Thus, we solve $\hat{\mathbf{M}}^l$ instead of $\mathbf{M}^l$. This is achieved

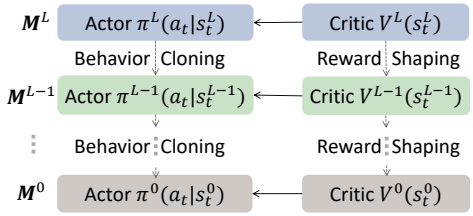


Figure 4: Architecture of the Cascading Actor-Critic.

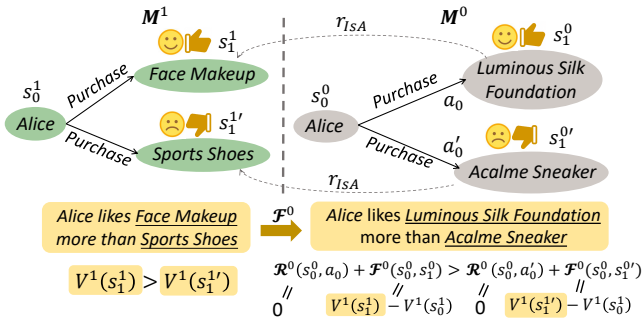


Figure 5: Illustration of how reward shaping ($\mathcal{F}^0$) helps transfer high-level knowledge ($\mathbf{M}^1$) to a lower level ($\mathbf{M}^0$).

by changing the loss in Eq. (9) to

$$J_{V^l} = ||\mathcal{R}^0(s_t^0, a_t) + \mathcal{F}^l(s_t^l, s_{t+1}^l) + V^l(\phi^l(s_{t+1}^0)) - V^l(\phi^l(s_t^0))||^2 \quad (11)$$

Fig. 5 illustrates how reward shaping helps transfer the knowledge from a higher level ($\mathbf{M}^1$) to a lower level ($\mathbf{M}^0$). After solving $\mathbf{M}^1$, V^1 learns that *Alice* likes *Face Makeup* more than *Sports Shoes* (i.e., $V^1(s_1^1) > V^1(s_1^{1\prime})$). This knowledge is transferred to level 0 by the shaping function $\mathcal{F}^0$, which immediately reveals that *Alice* likes *Luminous Silk Foundation* more than *Acalme Sneaker* by assigning a higher reward to the former (i.e., $\mathcal{R}^0(s_0^0, a_0) + \mathcal{F}^0(s_0^0, s_1^0) > \mathcal{R}^0(s_0^0, a_0{}') + \mathcal{F}^0(s_0^0, s_1^{0\prime})$). Compared with the sparse reward $\mathcal{R}^0$ that is only non-zero at the terminal state ($t = T$), $\mathcal{F}^0$ provides useful information for each $t \leq T$, which prunes the search space and guides the learning at level 0 to converge to a more satisfying solution.

3.3 Multi-level Reasoning Path Extraction

Based on the outputs of the Cascading Actor-Critic, we extract multi-level reasoning paths. In a multi-level path, each concept $c_t^{0\sim L}(t < T)$ may belong to any level in the KG. Compared with single-level paths, multi-level paths provide the flexibility that is essential for correctly presenting user interests. We extract the multi-level path $\tau_u^{0\sim L}$ with two steps: multi-level value function learning and value-based path extraction.

3.3.1 Multi-level Value Function Learning. We learn a multi-level value function $V^{0\sim L}$ to estimate the expected reward for a multi-level state $s_t^{0\sim L} = (u, c_{t-K}^{0\sim L}, r_{t-K+1} \ldots, c_{t-1}^{0\sim L}, r_t, c_t^{0\sim L})$. This is achieved by extending the value function learning method at a single level l. Specifically, in each iteration, we: 1) sample $(s_t^0, a_t, s_{t+1}^0, \mathcal{R}^0(s_t^0, a_t))$ from $\mathbf{M}^0$ according to $\mathcal{P}^0$ and π^0; 2) generate multi-level sample $(s_t^{0\sim L}, a_t, s_{t+1}^{0\sim L}, \mathcal{R}^0(s_t^0, a_t))$ by randomly mapping concepts in s_{t+1}^0 to one of the L levels; and 3) minimize $J_{V^{0\sim L}} + J'_{V^{0\sim L}}$, where

$$J_{V^{0\sim L}} = ||\mathcal{R}^0(s_t^0, a_t) + \mathcal{F}^0(s_t^0, s_{t+1}^0) + V^{0\sim L}(s_{t+1}^{0\sim L}) - V^{0\sim L}(s_t^{0\sim L})||^2$$
$$J'_{V^{0\sim L}} = ||V^{0\sim L}(s_{t+1}^{0\sim L}) - V^0(s_{t+1}^0)||^2 \quad (12)$$

$J_{V^{0\sim L}}$ and $J'_{V^{0\sim L}}$ optimize the model parameters of $V^{0\sim L}$ by using the reward $\mathcal{R}^0$ and the learned value function V^0, respectively.

Interpretation. The state space $\mathcal{S}^{0\sim L}$ of $V^{0\sim L}$ is a super set of $\mathcal{S}^0$. For bottom-level states in $\mathcal{S}^0$, $V^{0\sim L}$ is equal to V^0, while at higher levels, $V^{0\sim L}$ provides additional opportunities to ensemble different outputs of π^0, so that we may still achieve a good result when π^0 fails to capture the true user interest at the bottom level.

3.3.2 Value-based Path Extraction. Given state $s_{t-1}^{0\sim L}$, we select the next state $s_t^{0\sim L}$ by maximizing the estimated reward from $s_{t-1}^{0\sim L}$:

$$\arg\max_{s_t^{0\sim L}} \hat{\mathcal{R}}(s_{t-1}^{0\sim L}, s_t^{0\sim L}) + V^{0\sim L}(s_t^{0\sim L}) \quad (13)$$

Here, $\hat{\mathcal{R}}(s_{t-1}^{0\sim L}, s_t^{0\sim L})$ is the predicted reward for selecting $s_t^{0\sim L}$ at state $s_{t-1}^{0\sim L}$. The true reward is not used here because it is unavailable during testing. In path reasoning, there is only terminal reward. Thus, $\hat{\mathcal{R}}(s_{t-1}^{0\sim L}, s_t^{0\sim L})$ is 0 when $t \neq T$. At the terminal state, $\hat{\mathcal{R}}(s_{T-1}^{0\sim L}, s_T^{0\sim L})$ estimates how much the user likes the item $v_u = c_T^{0\sim L}$ at the end of the path, which can be any recommendation model. In this paper, we set $\hat{\mathcal{R}}(s_{t-1}^{0\sim L}, s_t^{0\sim L})$ to the scoring function in [30].

By sequentially selecting state $s_t^{0\sim L}$, we obtain a reasoning path $\tau_u^{0\sim L}$, in which the t-th hop consists of $(r_t, c_t^{0\sim L})$ in $s_t^{0\sim L}$. To avoid local minimum, we keep the top N_t states at time t. This allows us to obtain $\prod_{t=1}^{T} N_t$ candidate paths. Among these candidates, the top paths are selected according to Eq. (13). The set of items V_u at the end of the extracted paths will be recommended to user u.

4 EVALUATION

4.1 Experimental Setup

4.1.1 Datasets. Three recommendation datasets are leveraged in the evaluation: **Beauty**, **Clothing** and **Cell _Phones**. They are from three product categories ("Beauty", "Clothing, Shoes, and Jewelry", and "Cell Phones and Accessories") of Amazon 5-core[2]. We randomly select 70% of the user purchase (interaction) history as the training data and consider the remaining 30% as the test data. Based on the product meta information and reviews in each dataset, we build an **instance-view KG** by following Xian et al. [30]. Each dataset contains 5 types of entities and 8 types of relations. The **ontology-view KG** is constructed by using the Microsoft Concept Graph as introduced in Sec. 2.1. In this paper, we build a three-level KG (i.e., L=2), which we find to well balance efficiency and recommendation accuracy empirically. The statistics of the datasets are given in the supplement.

4.1.2 Baselines. We compare our methods with 7 recommendation models. **BPR** [19] is learned by considering only user-item interactions, while **Ripple Net** [22], **DKN** [23], **RuleRec** [14], **CKE** [34], **KGAT** [25], and **PGPR** [30] recommend items by incorporating KG information. Among them, PGPR conducts recommendation path reasoning on the KG, and is the most similar with ours. To eliminate the bias introduced by data, we test each KG-based baseline on three different KGs and report the best result. The three KGs are instance-view KG $\mathcal{G}^0$, the multi-level KG $\mathcal{G}^{0\sim L}$, and the multi-level KG with multiple parents. Since the baselines lack a mechanism to model parent concepts, we apply them to multi-level KG by treating concepts and r_{IsA} in the same way as entities and bottom-level relations.

4.1.3 Evaluation Criteria. We adopt four widely-used evaluation criteria: Precision (**Precision**), Recall (**Recall**), Normalized Discounted Cumulative Gain (**NDCG**) and Hit Rate (**HR**) [22, 25, 30].

[2] http://jmcauley.ucsd.edu/data/amazon

Table 1: Comparison of recommendation accuracy. The best results are highlighted in bold. Results are reported in percentages.

Datasets & Metrics	Beauty				Clothing				Cell_Phones			
	Precision	Recall	NDCG	HR	Precision	Recall	NDCG	HR	Precision	Recall	NDCG	HR
BPR	1.173	5.032	2.805	8.933	0.323	1.219	0.665	1.932	0.623	3.534	1.995	5.424
Ripple Net	1.203	5.121	2.342	9.337	0.392	1.137	0.602	1.889	0.720	3.434	2.901	5.660
DKN	1.135	2.591	1.923	8.812	0.119	0.724	0.375	1.492	0.349	3.313	1.672	4.580
RuleRec	1.239	6.032	3.072	10.731	0.302	1.173	0.652	2.166	0.738	3.648	2.023	5.772
CKE	1.422	6.241	3.824	11.132	0.390	2.604	1.656	4.329	1.073	6.981	3.849	10.633
KGAT	1.535	7.794	5.020	12.496	0.603	4.674	2.824	6.993	1.134	7.982	4.803	11.241
PGPR	1.692	8.324	5.489	14.347	0.719	4.797	2.863	7.024	1.280	8.383	4.921	11.832
ReMR (Ours)	**1.906**	**8.982**	**5.878**	**15.606**	**0.766**	**5.110**	**2.977**	**7.426**	**1.337**	**8.724**	**5.294**	**12.498**
Improvement (%)	+12.6	+7.9	+7.1	+8.8	+6.5	+4.7	+4.0	+5.7	+4.5	+4.1	+7.6	+5.6

Following Xian et al. [30], we compute the criteria based on the top-10 recommended items for each user in the test set.

4.1.4 Implementation Details. We set most hyperparameters of our method by following that of PGPR [30]. Please refer to the supplement for more details about the implementation.

4.2 Overall Performance

4.2.1 Recommendation Accuracy. Table 1 shows that our method consistently outperforms all baselines in terms of recommendation accuracy. On average, our model improves Precision, Recall, NDCG, and HR by 7.9%, 5.6%, 6.2%, and 6.7%. This demonstrates that multi-level reasoning can help better infer user interests and improve recommendation accuracy.

4.2.2 Convergence. We compare the convergence of our method ReMR with the most competitive baseline (PGPR) on Beauty and Clothing. The result shows that ReMR converges with much fewer epochs. The detailed results and a further discussion about efficiency are given in the supplement.

4.3 Influence of Components and Parameters

4.3.1 Effectiveness of Multi-level Path Extraction and Cascading Actor-Critic. Table 2 shows that ReMR outperforms ReMR-M, which is a variant of our method that eliminates multi-level path extraction and generates bottom-level paths by using π^0 of the Cascading Actor-Critic. The fact that ReMR outperforms ReMR-M reveals that compared with bottom-level paths, multi-level paths are more effective for inferring user interest. Moreover, ReMR-M consistently outperforms PGPR, although they both generate bottom-level paths. This indicates that by effectively transferring knowledge from high-level reasoning processes to low-level ones, our Cascading Actor-Critic enables the convergence to a more accurate solution.

4.3.2 Effectiveness of Multi-level KG. To understand how different parts of the multi-level KG influence performance, we implement two variants of our method, ReMR-L1 and ReMR-L2. The two methods are learned without the first and second level of the ontology-view KG, respectively. As shown in Table 2, ReMR outperforms ReMR-L1 and ReMR-L2, which demonstrates that both levels of the ontology-view KG provide valuable information for improving recommendation accuracy. Moreover, ReMR-L2 consistently achieves better performance compared with ReMR-L1, which indicates that level 2 is less useful. This is reasonable since higher levels are more abstract and may contain more noises.

Table 2: Ablation study. ReMR-L1 (or ReMR-L2) is learned without the 1st (or 2nd) level of the ontology-view KG. ReMR-M eliminates multi-level path extraction.

	Beauty				Clothing			
	Prec.	Recall	NDCG	HR	Prec.	Recall	NDCG	HR
PGPR	1.692	8.324	5.489	14.327	0.719	4.797	2.863	7.024
ReMR-L1	1.703	8.575	5.527	14.823	0.729	4.882	2.901	7.254
ReMR-L2	1.724	8.625	5.617	15.038	0.743	5.019	2.916	7.303
ReMR-M	1.759	8.637	5.706	14.723	0.730	4.846	2.906	7.211
ReMR	**1.906**	**8.982**	**5.878**	**15.606**	**0.766**	**5.110**	**2.977**	**7.426**

Table 3: Influence of state history length K on performance.

	Beauty				Clothing			
Hist.	Prec.	Recall	NDCG	HR	Prec.	Recall	NDCG	HR
K=0	1.139	5.594	3.692	10.846	0.511	3.598	2.150	5.151
K=1	1.906	8.982	5.878	15.606	0.766	5.110	2.977	7.426
K=2	**2.005**	**9.122**	**5.903**	**16.214**	**0.782**	**5.176**	**3.079**	**7.638**

Table 4: Influence of path search size $N_1 \times N_2 \times N_3$.

	Beauty				Clothing			
Size	Prec.	Recall	NDCG	HR	Prec.	Recall	NDCG	HR
30×5×1	1.885	8.723	5.625	15.132	0.765	5.078	2.826	7.114
25×6×1	1.926	9.043	5.893	15.531	0.770	5.115	2.985	7.477
15×10×1	**1.967**	**9.257**	**5.963**	**15.971**	**0.782**	**5.123**	**3.105**	**7.587**
15×5×2	1.864	8.562	5.162	14.935	0.766	4.830	2.743	6.824
10×5×3	1.522	7.489	4.565	14.661	0.693	4.526	2.582	6.599

4.3.3 Sensitivity of State History Length K. Table 3 shows that the accuracy increases with increasing K, meaning that longer state history enables more comprehensive representation of the environment and leads to a better policy. We also observe that by changing $K = 1$ to $K = 2$, the improvement in terms of accuracy is not significant, while the convergence of the policy becomes much slower (see the supplement). This indicates that setting K to 1 well balances accuracy and efficiency and is a good choice in practice.

4.3.4 Influence of Path Search Size $N_1 \times N_2 \times N_3$. N_t is the maximum number of states (actions) we select at time t when searching the reasoning paths. Table 4 shows how performance changes with different values of $N_1 \times N_2 \times N_3$. The model generally achieves better performance when N_3 is 1. This indicates that the actions in earlier steps are more important, and improving the search sizes at the beginning enables the model to better explore reasoning paths.

Table 5: Impact of incorporating demonstration paths.

	Beauty				Clothing			
Hist.	Prec.	Recall	NDCG	HR	Prec.	Recall	NDCG	HR
ADAC	1.991	9.424	6.080	16.036	0.763	5.027	3.048	7.502
ReMR+D	**2.131**	**9.601**	**6.323**	**16.806**	**0.794**	**5.172**	**3.079**	**7.831**
Imprv.(%)	+7.0	+1.9	+4.0	+4.8	+4.1	+2.9	+1.0	+4.4

4.3.5 Integrating Additional Guidance. To understand the impact of integrating additional guidance for convergence, e.g., expert demonstrations [36], we implement ReMR+D, which integrates ADAC [36] into our framework and leverages both demonstrations and the multi-level KG for recommendation reasoning. As Table 5 shows, ReMR+D consistently performs better than ADAC. This reveals that the multi-level reasoning strategy is still effective even when other information is provided for guiding the reasoning process.

4.4 Case Study

The case study shown in Fig. 6 illustrates how multi-level reasoning helps better infer and represent user interests.

First, multi-level reasoning improves recommendation accuracy by clarifying the meaning of an entity in the context of the overall user behavior. For example, the word *Tub* in the figure can be interpreted in multiple ways, e.g., as a *Cosmetic Container* or related with the *Rasdder Bath Brush* (item ❶). ReMR correctly understands that the user mentions *Tub* in the context of *Cosmetic Container* by jointly considering the word *Jar* mentioned by the user. Accordingly, our method recommends the correct items. In comparison, PGPR incorrectly connects *Tub* with item ❶ and recommends a wrong product *Minera Bath Salt* (item ❷).

Second, multi-level paths provide better explainability compared with bottom-level paths. Bottom-level paths explain about the recommendation by referring to only one user behavior, which limits their persuasiveness. For example, the upper red curve in Fig. 6 explains that an item is recommended because the user mentions *Jar*, and that one item described by *Jar* is often bought together with the recommended item. This path is not very persuasive: mentioning *Jar* does not seem to be a user behavior that is important enough for confidently recommending items, since a user can mention many other words. In comparison, the multi-level paths explain about the recommendation according to a lot more user behaviors (user mentions multiple words related to both *Container* and *Lotion*) and it becomes much more understandable why the item is recommended and what the true interest of the user is.

5 RELATED WORK

Existing KG-based recommendation models can be classified into two major categories based on whether they explicitly model the sequential connectivity patterns (paths) between users and items.

KG embedding approaches learn entity and/or relation representations from the KG and incorporate the learned representations into the recommendation model to improve recommendation accuracy [1, 2, 10, 16, 17, 21, 35]. For example, Zhang et al. propose a unified framework that jointly models latent representations in collaborative filtering and item semantic representations learned from knowledge bases [33]. Wang et al. develop a deep knowledge-aware network for news recommendation [23]. Recently, researchers have

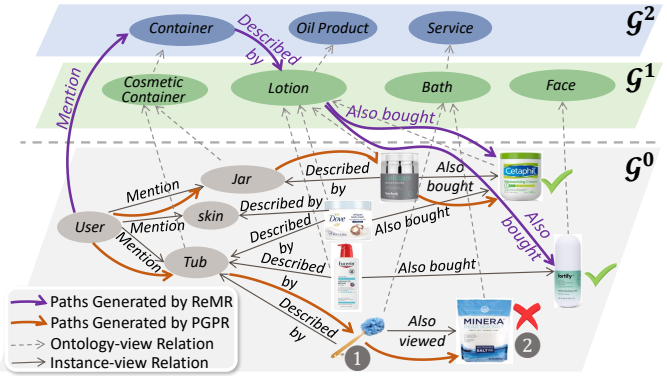


Figure 6: Case study on Beauty. We show top-2 paths generated by using our method (ReMR) and PGPR. Correct and wrong recommendations are marked with ✓ and ✗.

drawn inspiration from developments in graph neural networks [7] to better model the high-order relations in the KG [24, 25, 38]. These methods enable flexible incorporation of KG into recommendation and can successfully improve recommendation accuracy. However, they do not directly model the connectivity patterns between users and items. As a consequence, their accuracy and reasoning capabilities are hampered. For example, these methods cannot provide explanations by extracting multi-hop reasoning paths that connect users with the recommended items.

Path reasoning approaches explicitly model the paths between users and items over KGs for recommendation. It has been shown that learning user interests by path-aware aggregation improves recommendation accuracy [4, 26, 31, 40]. Moreover, explicitly modeling the paths enables *reasoning*, which is a crystal type of explainability. Pioneer works in reasoning enumerate all possible candidate paths on the KG and score each candidate path with a deep model to select the best one [5, 14, 27, 39]. These methods improve recommendation accuracy and explainability, but the exhaustive path search strategy is not applicable to large-scale KGs [30]. To solve these issues, RL has been utilized to learn a path search policy [3, 13, 30, 36, 37]. However, these methods either have difficulty converging to a good solution due to the sparse reward signals [30], or rely on heuristics and/or additional user inputs [11, 36]. Moreover, they focus only on instance-view KG, which prevents them from better inferring and representing user interests. We solve these issues by proposing a multi-level reasoning framework that effectively integrates ontology-view KG with instance-view KG. Our method enables better modeling of user interests and improves the recommendation accuracy without additional user inputs or biased heuristics.

6 CONCLUSION

We propose an RL framework for multi-level recommendation reasoning over KGs, which formulates the low-level and high-level reasoning process as MDP and abstract MDPs. We then propose a Cascading Actor-Critic to solve the multi-level MDP. Finally, a multi-level reasoning path extraction method is introduced, which selects between high-level and low-level concepts to better represent user interests. Experiments demonstrate that our method improves recommendation accuracy and explainability.

REFERENCES

[1] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019. Unifying Knowledge Graph Learning and Recommendation: Towards a Better Understanding of User Preferences. In *The Web Conference*. 151–161.

[2] Chong Chen, Min Zhang, Weizhi Ma, Yiqun Liu, and Shaoping Ma. 2020. Jointly non-sampling learning for knowledge graph enhanced recommendation. In *SIGIR*. 189–198.

[3] Zhihong Cui, Hongxu Chen, Lizhen Cui, Shijun Liu, Xueyan Liu, Guandong Xu, and Hongzhi Yin. 2021. Reinforced KGs reasoning for explainable sequential recommendation. *World Wide Web* (2021), 1–24.

[4] Chao Feng, Defu Lian, Xiting Wang, Zheng liu, Xing Xie, and Enhong Chen. 2022. Reinforcement Routing on Proximity Graph for Efficient Recommendation. arXiv:2201.09290 [cs.IR]

[5] Zuohui Fu, Yikun Xian, Ruoyuan Gao, Jieyu Zhao, Qiaoying Huang, Yingqiang Ge, Shuyuan Xu, Shijie Geng, Chirag Shah, Yongfeng Zhang, et al. 2020. Fairness-aware explainable recommendation over knowledge graphs. In *SIGIR*. 69–78.

[6] Ivo Grondman, Lucian Busoniu, Gabriel AD Lopes, and Robert Babuska. 2012. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42, 6 (2012), 1291–1307.

[7] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*. 1025–1035.

[8] Junheng Hao, Muhao Chen, Wenchao Yu, Yizhou Sun, and Wei Wang. 2019. Universal representation learning of knowledge bases by jointly embedding instances and ontological concepts. In *KDD*. 1709–1719.

[9] Charles Heriot-Maitland. 2012. Multi-level models of information processing, and their application to psychosis. *Journal of Experimental Psychopathology* 3, 4 (2012), 552–571.

[10] Jin Huang, Wayne Xin Zhao, Hong-Jian Dou, Ji-Rong Wen, and Edward Y. Chang. 2018. Improving Sequential Recommendation with Knowledge-Enhanced Memory Networks. In *SIGIR*. 505–514.

[11] Wenqiang Lei, Gangyi Zhang, Xiangnan He, Yisong Miao, Xiang Wang, Liang Chen, and Tat-Seng Chua. 2020. Interactive path reasoning on graph for conversational recommendation. In *KDD*. 2073–2083.

[12] Lihong Li, Thomas J Walsh, and Michael L Littman. 2006. Towards a Unified Theory of State Abstraction for MDPs. In *ISAIM*, Vol. 4. 5.

[13] Danyang Liu, Jianxun Lian, Zheng Liu, Xiting Wang, Guangzhong Sun, and Xing Xie. 2021. Reinforced Anchor Knowledge Graph Generation for News Recommendation Reasoning. In *KDD*. 1055–1065.

[14] Weizhi Ma, Min Zhang, Yue Cao, Woojeong Jin, Chenyang Wang, Yiqun Liu, Shaoping Ma, and Xiang Ren. 2019. Jointly Learning Explainable Rules for Recommendation with Knowledge Graph. In *The Web Conference*. 1210–1221.

[15] Andrew Y Ng, Daishi Harada, and Stuart Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, Vol. 99. 278–287.

[16] Chien-Chun Ni, Kin Sum Liu, and Nicolas Torzec. 2020. Layered graph embedding for entity recommendation using wikipedia in the yahoo! knowledge graph. In *Companion Proceedings of the Web Conference*. 811–818.

[17] Enrico Palumbo, Diego Monti, Giuseppe Rizzo, Raphaël Troncy, and Elena Baralis. 2020. entity2rec: Property-specific knowledge graph embeddings for item recommendation. *Expert Systems with Applications* 151 (2020), 113235.

[18] Dean Pomerleau. 1991. Efficient Training of Artificial Neural Networks for Autonomous Navigation. *Neural Computation* 3, 1 (1991), 88–97.

[19] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*. 452–461.

[20] Ben Shneiderman. 2003. The eyes have it: A task by data type taxonomy for information visualizations. In *The craft of information visualization*. Elsevier, 364–371.

[21] Xiaoli Tang, Tengyun Wang, Haizhi Yang, and Hengjie Song. 2019. AKUPM: Attention-enhanced knowledge-aware user preference model for recommendation. In *KDD*. 1891–1899.

[22] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. RippleNet: Propagating User Preferences on the Knowledge Graph for Recommender Systems. In *CIKM*. 417–426.

[23] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. DKN: Deep Knowledge-Aware Network for News Recommendation. In *The Web Conference*. 1835–1844.

[24] Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. 2019. Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In *KDD*. 968–977.

[25] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. Kgat: Knowledge graph attention network for recommendation. In *KDD*. 950–958.

[26] Xiang Wang, Tinglin Huang, Dingxian Wang, Yancheng Yuan, Zhenguang Liu, Xiangnan He, and Tat-Seng Chua. 2021. Learning Intents behind Interactions with Knowledge Graph for Recommendation. In *The Web Conference*. 878–887.

[27] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable Reasoning over Knowledge Graphs for Recommendation. In *AAAI*. 5329–5336.

[28] Zhongyuan Wang, Haixun Wang, Ji-Rong Wen, and Yanghua Xiao. 2015. An inference approach to basic level of categorization. In *CIKM*. ACM, 653–662.

[29] Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q Zhu. 2012. Probase: A probabilistic taxonomy for text understanding. In *SIGMOD*. ACM, 481–492.

[30] Yikun Xian, Zuohui Fu, S. Muthukrishnan, Gerard de Melo, and Yongfeng Zhang. 2019. Reinforcement Knowledge Graph Reasoning for Explainable Recommendation. In *SIGIR*. 285–294.

[31] Yikun Xian, Zuohui Fu, Handong Zhao, Yingqiang Ge, Xu Chen, Qiaoying Huang, Shijie Geng, Zhou Qin, Gerard De Melo, Shan Muthukrishnan, et al. 2020. CAFE: Coarse-to-fine neural symbolic reasoning for explainable recommendation. In *CIKM*. 1645–1654.

[32] Weikai Yang, Xiting Wang, Jie Lu, Wenwen Dou, and Shixia Liu. 2020. Interactive steering of hierarchical clustering. *IEEE Transactions on Visualization and Computer Graphics* (2020).

[33] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative Knowledge Base Embedding for Recommender Systems. In *KDD*. 353–362.

[34] Wei Zhang, Quan Yuan, Jiawei Han, and Jianyong Wang. 2016. Collaborative multi-Level embedding learning from reviews for rating prediction.. In *IJCAI*. 2986–2992.

[35] Jun Zhao, Zhou Zhou, Ziyu Guan, Wei Zhao, Wei Ning, Guang Qiu, and Xiaofei He. 2019. Intentgc: a scalable graph convolution framework fusing heterogeneous information for recommendation. In *KDD*. 2347–2357.

[36] Kangzhi Zhao, Xiting Wang, Yuren Zhang, Li Zhao, Zheng Liu, Chunxiao Xing, and Xing Xie. 2020. Leveraging Demonstrations for Reinforcement Recommendation Reasoning over Knowledge Graphs. In *SIGIR*. 239–248.

[37] Yuyue Zhao, Xiang Wang, Jiawei Chen, Wei Tang, Yashen Wang, Xiangnan He, and Haiyong Xie. 2021. Time-aware Path Reasoning on Knowledge Graph for Recommendation. *arXiv preprint arXiv:2108.02634* (2021).

[38] Kun Zhou, Wayne Xin Zhao, Shuqing Bian, Yuanhang Zhou, Ji-Rong Wen, and Jingsong Yu. 2020. Improving conversational recommender systems via knowledge graph based semantic fusion. In *KDD*. 1006–1014.

[39] Qiannan Zhu, Xiaofei Zhou, Jia Wu, Jianlong Tan, and Li Guo. 2020. A knowledge-aware attentional reasoning network for recommendation. In *AAAI*. 6999–7006.

[40] Yaxin Zhu, Yikun Xian, Zuohui Fu, Gerard de Melo, and Yongfeng Zhang. 2021. Faithfully Explainable Recommendation via Neural Logic Reasoning. In *NAACL*.

SUPPLEMENTARY MATERIAL

In this supplementary material, we present the notation table (Sec. 1), discuss more about the theoretical properties of our method (Sec. 1), provide more details for the experiments (Sec. 3), and discuss the efficiency of our method as well as how it can be extended to handle other types of KGs (Sec. 4).

1 NOTATION TABLE

Table 1 summarizes the important notations used in this paper.

Table 1: Summary of important notations.

Notations	Descriptions
$\mathcal{G}^l$	The KG at level l
$\mathcal{G}^{0\sim L} = \{\mathcal{G}^0, ...\mathcal{G}^L\}$	A multi-level KG with L+1 levels
c_i^l and C^l	A concept and the concept set of $\mathcal{G}^l$
u	A user, which is in C^0
r_i and Υ	A relation and the relation set of $\mathcal{G}^{0\sim L}$
$\mathbf{M}^l$	The (abstract) MDP at level l
s_t^l and $\mathcal{S}^l$	A state and the state space of $\mathbf{M}^l$
a_t and $\mathcal{A}$	An action and the action space
$\mathcal{R}^l$	The reward function of $\mathbf{M}^l$
$\mathcal{P}^l$	The state transition function of $\mathbf{M}^l$
ϕ^l	The state abstraction function of $\mathbf{M}^l$
ω^l	The weighting function of $\mathbf{M}^l$
π^l	The actor or policy network at level l
V^l	The critic or value network at level l
$V^{0\sim L}$, $s_t^{0\sim L}$, $c_i^{0\sim L}$	A multi-level critic, state, and concept

2 THEORETICAL PROPERTIES

Theorem 1. Suppose that $\tau_{u,t}^l = (u, r_1, c_1^l, \cdots r_t, c_t^l)$ is a reasoning path and $s_t^l = (u, c_{t-K}^l, r_{t-K+1} \ldots, c_{t-1}^l, r_t, c_t^l)$ is its corresponding state. if s_t^l can be sampled from $\mathbf{M}^l$, i.e., there exist s_{t-1}^l and a_{t-1} such that $\mathcal{P}^l(s_{t-1}^l, a_{t-1}, s_t^l) > 0$, then $\tau_{u,t}^l$ must be feasible.

Proof. Proving the feasibility of $\tau_{u,t}^l$ is equivalent to proving that there exist $\tau_{u,t}^0 = (u = c_0^0, r_1, c_1^0 \cdots r_t, c_t^0)$ that satisfies 1) $\tau_{u,t}^0$ is a path on $\mathcal{G}^0$, i.e., for $\forall t' \in [0, t-1]$, we have $(c_{t'}^0, r_{t'+1}, c_{t'+1}^0) \in \mathcal{G}^0$; and 2) the high-level path of $\tau_{u,t}^0$ is $\tau_{u,t}^l$, i.e., $\varphi^l(c_{t'}^0) = c_{t'}^l$ for $\forall t' \in [1, t]$. We prove this by finding a $\tau_{u,t}^0$ that satisfies these conditions.

According to Eq. (3) and $\mathcal{P}^l(s_{t-1}^l, a_{t-1}, s_t^l) > 0$, we can prove that there exist s_{t-1}^0, $a_{t-1} = (r_t, c_t^0)$, and s_t^0 such that: $\phi^l(s_{t-1}^0) = s_{t-1}^l$, $\phi^l(s_t^0) = s_t^l$, $\omega^l(s_{t-1}^0) > 0$, and $\mathcal{P}(s_{t-1}^0, a_{t-1}, s_t^0) > 0$. Otherwise, $\mathcal{P}^l(s_{t-1}^l, a_{t-1}, s_t^l)$ will be 0. By combining $\omega^l(s_{t-1}^0) > 0$ and Eq. (4), we can derive $E_{\mathcal{P}^0, \tilde{\pi}^0} N(s_{t-1}^0) > 0$, which means that the probability for sampling s_{t-1}^0 from $\mathbf{M}^0$ is larger than 0. Thus, there must exist a reasoning path $\tau_{u,t-1}^0$ on $\mathcal{G}^0$ whose state is s_{t-1}^0. We then obtain path $\tau_{u,t}^0$ by taking the action a_{t-1} after generating $\tau_{u,t-1}^0$. We will prove that $\tau_{u,t}^0$ is the bottom-level path that supports $\tau_{u,t}^l$.

Let us denote $\tau_{u,t-1}^0$ as $(u = c_0^0, r_1, c_1^0 \cdots, r_{t-1}, c_{t-1}^0)$. Because $\tau_{u,t-1}^0$ is a path on $\mathcal{G}^0$, we have $(c_{t'}^0, r_{t'+1}, c_{t'+1}^0) \in \mathcal{G}_0$ for $\forall t' \in [0, t-2]$. Since $\phi^l(s_{t-1}^0) = s_{t-1}^l$ and $\tau_{u,t-1}^0$ corresponds to s_{t-1}^0, we have $\varphi^l(c_{t'}^0) = c_{t'}^l$ for $\forall t' \in [1, t-1]$. According to $\mathcal{P}(s_{t-1}^0, a_{t-1}, s_t^0) > 0$, $a_{t-1} = (r_t, c_t^0)$ is a feasible action for state s_{t-1}^0, i.e., $(c_{t-1}^0, r_t, c_t^0) \in \mathcal{G}_0$. Since $\phi^l(s_t^0) = s_t^l$, we know that $\varphi^l(c_t^0) = c_t^l$. Thus, $\tau_{u,t}^0$ generated by taking action a_{t-1} after generating $\tau_{u,t-1}^0$ satisfies 1) $(c_{t'}^0, r_{t'+1}, c_{t'+1}^0) \in \mathcal{G}_0$ for $\forall t' \in [0, t-1]$ and 2) $\varphi^l(c_{t'}^0) = c_{t'}^l$ for $\forall t' \in [1, t]$. □

Handling inaccurate high-level knowledge. When utilizing high-level knowledge, an key question is whether we can ensure that the learned high-level knowledge is correct, especially when the data is sparse. Take Fig. 1 as an example. Summarizing that *Alice* likes *Cosmetics* based on two purchase behaviors of Alice may not be accurate due to the sparse data (a limited number of observed user behaviors). An advantage of our method is that it naturally solve this issue. We will explain how we eliminate the negative impact of inaccurate high-level knowledge when 1) learning, 2) passing, and 3) selecting high-level knowledge.

1. Learning high-level knowledge. We learn the high-level knowledge by computing a value function V^l, which automatically decides the importance of high-level knowledge. The importance is determined based on how helpful the high-level knowledge is in improving recommendation accuracy. Take the top level in Fig. 1 as an example. The value (V^l) for choosing *Cosmetics* or *Sports* is determined based on the expected recommendation accuracy (reward) of their bottom-level paths (last paragraph of Secs. 3.1.2 and 3.2.1). If the number of examples is too small to clearly distinguish *Cosmetics* from *Sports* (no significant difference between their recommendation accuracy), then the value for choosing *Cosmetics* and *Sports* will be similar. In this case, low-level policies will treat *Cosmetics* and *Sports* in a similar way (Eq. (10)), just like when there is no high-level knowledge. In other words, the high-level knowledge has a low importance (small impact on the result). Note that the value function is learned by considering all users, thus it can also use collective information from other samples to further handle data sparsity like recommendation models (similar users have similar results).

2. Passing high-level knowledge to lower levels. Even if part of the value function (high-level knowledge) is not accurate, we can eliminate bias with optimal policy consistency when passing knowledge to lower levels. Optimal policy consistency means that the optimal low-level policy remains the same after integrating high-level knowledge (a formal discussion in the paragraph after Eq. (10)). Thus, even if the high-level knowledge is incorrect (*Alice* does not like *Cosmetics*), the low-level policy will not make a biased decision (e.g., selecting paths related to *Cosmetics* even if they lead to a low accuracy). Instead, the algorithm will only first explore the more promising direction (*Cosmetics*) judging by the high-level knowledge, which typically enables more effective exploration of the action space and helps avoid local minimum. If the algorithm finds that the accuracy for *Cosmetics* is not good, it will finally choose other paths to optimize the reward (accuracy).

3. Selecting the correct level of knowledge. To further eliminate the negative impact of inaccurate high-level knowledge, we have a mechanism for choosing the best level of knowledge (Sec. 3.3). Take Fig. 1 as an example. The high-level path (*Alice* purchases *Cosmetics* produced by *Italian Luxury Fashion Brands*) will only be chosen when it leads to better expected reward (accuracy) compared

Table 2: Statistics of datasets.

	Beauty	Clothing	Cell_Phone
#Users	22,363	39,384	27,879
#Items	12,101	23,033	10,429
#Interactions	198,502	278,677	194,439
#Entities	224,074	425,528	163,249
#Triplets	7,832,720	10,671,090	6,299,494
#Concept in $\mathcal{G}^1$	11,077	16,236	9,396
#Concept in $\mathcal{G}^2$	6,705	9,441	5,024

with the bottom-level paths (Eq. (13)). This design ensures that the expected accuracy of our multi-level method is better than that of the bottom-level method (a more formal discussion in the last paragraph of Sec. 3.3.1).

3 EXPERIMENTAL DETAILS

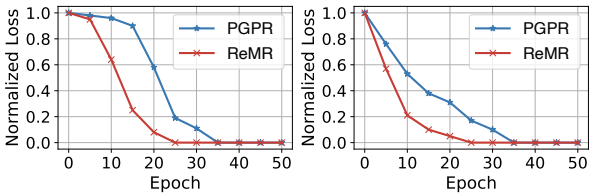
In this section, we provide more details about the experiments.

Dataset statistics are summarized in Table 2.

Implementation details. The hyperparameters for the baselines are initialized as in the corresponding papers, and tuned to achieve optimal performance. When choosing baselines, we focus on models for user-to-item recommendation. Methods for other recommendation tasks or requires additional data are omitted. We set most hyperparameters of our method by following that of PGPR [30]. In particular, the history length K is set to 1, and the embedding sizes of the relations, entities, and concepts are set to 100. Thus, the embedding of a state $s_t^l = (u, c_{t-1}^l, r_t, c_t^l)$ has a dimension of 1×400 (i.e., $d_1 = 400$). Hidden sizes d_2 and d_3 are set to 512 and 256, the action space size N_m is set to 250, the dropout rate is 0.5, and the maximum length of reasoning path T is 3. N_1, N_2, and N_3 are set to 25, 5, 1, respectively. The neural networks are trained for 50 epochs by using Adam optimizer with a learning rate of $1e^{-4}$.

Additional experimental results. We compare the convergence of our method ReMR with the most competitive baseline (PGPR) on Beauty and Clothing. We normalize the training losses of the actors to [0,1] and study how normalized losses decrease with training epochs. As shown in Fig. 1, PGPR converges in around 35 epochs, while ReMR converges in less than 25 epochs. This indicates that transferring knowledge from the higher-level reasoning processes helps prune the search space and enables faster convergence.

Fig. 2 shows that the convergence becomes much slower (Fig. 2) when $K = 1$ changes to $K = 2$. We can also see from the main paper that by changing $K = 1$ to $K = 2$, the improvement in terms of accuracy is not significant. This indicates that setting K to 1 well balances accuracy and efficiency and is a good choice in practice.



(a) Beauty (b) Clothing

Figure 1: Comparison of convergence.

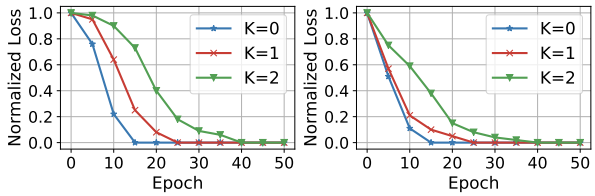
4 DISCUSSION

In this section, we discuss how our method can be extended to ontology-view KGs with multiple parents and other types of relations, as well as the efficiency of our method.

Considering multiple parents on the ontology-view KG. In this paper, we only consider the top-1 parent for each concept on the ontology-view KG. Experiments show that introducing this simple ontology-view KG already brings good improvement in recommendation performance. To further improve the performance, it may be beneficial to consider multiple parents of a concept, and allow the model to choose which parent is the best. This can be achieved by slightly modifying the sampling method used in the Cascading Actor-Critic, as what we have done in Sec. 3.3. Specifically, we can first sample $(s_t^0, a_t, s_{t+1}^0, \mathcal{R}^0(s_t^0, a_t))$ from $\mathbf{M}^0$, and then generate a high-level sample $(s_t^l, a_t, s_{t+1}^l, \mathcal{R}^0(s_t^0, a_t))$ by randomly mapping concepts to one of the parents, and finally minimize the same loss function. This allows a high-level path to aggregate the rewards of all its bottom-level paths. While the method is reasonable in theory, it may take many steps to learn a good actor-critic, especially when the number of parents is large. In this case, we can first prune the candidate parents and keep only those that are important in the specific dataset, e.g., by using the ant-colony-based algorithm proposed in [32]. We would like to experiment with these ideas in the future to see how they work for multiple parents.

Extending to ontology-view KGs with other types of relations. Some ontology-view KGs may contain other types of relations, in addition the parent-child relation like r_{IsA}. For example, in $\mathcal{G}^1$ of Fig. 6, there may exist a relation *Placed_In* between concepts *Lotion* and *Cosmetic Container*. If these high-level relations do not exist at the bottom level, it is currently difficult to extract a high-level path with such relations. However, our method provides a starting point for thinking about incorporating such information. For example, we may consider a mixture of MDPs at level l. Specifically, level l contains both the abstract MDP we currently considered, and another MDP that considers additional new relations provided by level l. When sampling the level-l states, we may use a mixture probability distribution to determine which MDP we will sample the states from.

Efficiency. Although the proposed method ReMR enables faster convergence at the bottom level (Fig. 1), overall ReMR may take more time as it requires additional higher level reasoning. Empirically, each epoch at a higher level takes approximately the same time as a training epoch at a lower level, since they both require samping from the bottom level. However, the number of epochs required at each level is significantly smaller than that without high-level knowledge. To further improve efficiency, we may integrate additional guidance (Sec. 4.3.5) or resort to more efficient methods for sampling bottom-level paths.



(a) Beauty (b) Clothing

Figure 2: Influence of state history length K on convergence.