Applied Soft Computing Journal 188 (2026) 114380

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Evolutionary reinforcement learning for explainable recommendation on knowledge graph

Yuanguo Lin [a,b,1], Hong Chen [c,1], Xiuze Zhou [c], Wei Zhang [b,*], Huanyu You [d], Fan Lin [b,*], Pengcheng Wu [e]

[a] School of Computer Engineering, Jimei University, Xiamen 361021, China
[b] School of Informatics, Xiamen University, Xiamen 361005, China
[c] Information Hub, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou 511453, China
[d] Institute of Systems Science, National University of Singapore, Singapore 119077, Singapore
[e] Nanyang Technological University, Singapore 639798, Singapore

HIGHLIGHTS

- We integrate evolutionary strategies with RL for explainable recommendations.
- A novel mutation-based mechanism dynamically optimizes the large action space.
- We propose an entropy-guided approach to improve policy optimization.
- Empirical results demonstrate that ERLER outperforms several baselines.

ARTICLE INFO

Keywords:
Recommender systems
Evolutionary reinforcement learning
Explainable recommendations
Knowledge graph reasoning
Policy optimization

ABSTRACT

In recent years, Knowledge Graphs (KGs) with Reinforcement Learning (RL) have emerged as powerful tools for the path reasoning in recommendation systems. However, explaining the reasoning behind recommendations remains a challenge, i.e., how to efficiently infer an effective path from the available options in large action spaces. To this end, we propose a novel framework, namely Evolutionary Reinforcement Learning for Explainable Recommendation (ERLER), that leverages KGs to deliver transparent and personalized recommendations. By combining evolutionary strategies with RL, the proposed framework dynamically optimizes recommendation policies in the path reasoning. Furthermore, we refine the critic network with an adaptive mechanism. This strengthens the framework's capacity for adjustment, thereby accelerating convergence and improving stability. Experimental results on benchmark datasets demonstrate that our ERLER not only achieves higher recommendation accuracy (e.g., improving Precision@10 by 2.28 % and HR@10 by 2.02 % over the state-of-the-art on the Amazon Clothing dataset) in most cases but also enhances the performance of path reasoning.

1. Introduction

In recent years, Knowledge Graphs (KGs) have been widely utilized in recommender systems, which mitigate the data sparsity problem and improve both recommendation accuracy and explainability through rich contextual information [17]. To better capture user preferences, KGs establish relationships between users and items [13], and introduce the semantic information of items into the recommendation process [49]. Besides, recommender systems equipped with KGs enhance user trust by explaining recommendations based on the structured data provided by KGs [6].

However, gathering information from various sources to construct a so-called complete KGs is a significant challenge, especially from unstructured, diverse, and inaccessible data sources. Therefore, manual data construction is necessary, but due to the high costs, the issue of incomplete KGs remains unresolved [19]. One possible solution to this issue is to introduce Reinforcement Learning (RL), which uses a

* Corresponding authors.
Email addresses: xdlyg@jmu.edu.cn (Y. Lin), hchen763@connect.hkust-gz.edu.cn (H. Chen), xzhou154@connect.hkust-gz.edu.cn (X. Zhou), vvcheung@stu.xmu.edu.cn (W. Zhang), e1221683@u.nus.edu (H. You), iamafan@xmu.edu.cn (F. Lin), pengchengwu@ntu.edu.sg (P. Wu).
[1] Co-first authors.

https://doi.org/10.1016/j.asoc.2025.114380
Received 6 December 2024; Received in revised form 14 November 2025; Accepted 24 November 2025
Available online 8 December 2025

weak supervision mechanism to design a reward function. Hence, it requires only a small amount of labeled data to achieve relatively good results, thus mitigating the incompleteness problem to some extent. DeepPath [61] learns multi-hop relational paths using a policy-based RL agent over KG embeddings. While it shows promising results on smaller KGs like Freebase, its performance degrades in terms of convergence speed and stability when facing very large, sparse KGs. The agent in RL conducts multi-hop reasoning on KGs through RL to improve explainability [47], or dynamically adjusts its path and infers missing relations and entities from partially known information [42].

Although RL brings new possibilities to KG-based recommender systems, it also introduces some new challenges, e.g., path reasoning convergence, which is computationally expensive when finding an optimal solution. While the exhaustive search method can locate the optimal path, its scalability is limited in large-scale KGs, as it requires a significant amount of time [40,50]. The REINFORCE algorithm from RL provides optimization for path finding through random search and sparse reward signals [61], but the convergence speed is relatively slow due to several factors, e.g., the large action space and sparse rewards. These approaches struggle to converge to an ideal solution when dealing with large-scale KGs.

In addition, the explainability issue in existing KG-based recommender systems with RL has received increasing attention. Since RL models aim to maximize rewards, the agent may adopt shortcuts during path reasoning [9], which might not provide a reasonable or explainable recommendation path. Moreover, in large action spaces, the quality of candidate actions varies significantly, making it even harder for RL to infer an explainable path from the available options. For instance, the QEPS model [79] proposes query-aware rewards and demonstration-guided policy to improve path explainability, but often at the cost of increased complexity and somewhat reduced generalization across diverse domains.

In this paper, we present a novel framework, namely Evolutionary Reinforcement Learning for Explainable Recommendation (ERLER), designed for efficient convergence while providing explainable recommendation paths. Compared with embedding-based methods that lack explicit reasoning paths and RL-based approaches that suffer from slow convergence in large action spaces, the framework optimizes the action space dynamically, reducing computational complexity and enhancing solution diversity through a mutation-based mechanism. This adaptive adjustment not only streamlines decision-making but also introduces variability that fosters exploration.

To further improve policy optimization, ERLER integrates entropy into the advantage calculation, enabling the policy to dynamically adjust its update direction and magnitude. This entropy-guided approach mitigates the risk of premature convergence by maintaining a balance between exploration and exploitation. By combining knowledge graphs with evolutionary strategies in this way, our framework effectively addresses the limitations of existing approaches while ensuring both efficiency and interpretability.

Our contributions can be summarized as follows:

- By integrating evolutionary strategies with RL, we introduce a novel action space optimization mechanism that enhances the accuracy and path reasoning over KGs.
- To improve the adaptive adjustment capabilities of the framework, we optimize the loss calculation of the critic network by introducing an adaptive mechanism, which leads to accelerated convergence and enhanced stability.
- Extensive experimental results demonstrate that our ERLER outperforms the state-of-the-art methods, in terms of recommendation accuracy and policy convergence.

2. Related work

2.1. Knowledge graph-based explainable recommendation

KG-based explainable recommendation models offer transparency and insight into the underlying reasoning behind recommendations, which encode relationships between entities in KG. Generally, KG-based explainable recommendations can be divided into the following approaches [17].

2.1.1. Embedding-based approach

Due to the ability to model complex relationships among users, items, and attributes in a high-dimensional space, the embedding-based approach [65] has gained traction for building explainable recommender systems. It leverages KG structure to represent entities as low-dimensional vectors (embeddings), to make personalized and explainable recommendations [4]. For instance, to achieve the explainable training course recommendations, Yang et al. [63] proposed a contextualized embedding-based approach, which learns the representations of talents and courses by the contextualized neighbor semantics. Also, the attention mechanism is usually applied to improve both recommendation accuracy and the explainability of results in the embedding-based approach [23]. Wang et al. [48] utilized the attention mechanism and multi-task learning method to predict user preferences for entities in the KGs. However, embedding-based methods often struggle to provide explicit reasoning paths, since the explanation is usually derived from vector similarity rather than human-readable connections in the KG. This limits their interpretability in real-world scenarios, despite their strong performance in accuracy.

2.1.2. Connection-based approach

By focusing on explicit paths and connections in the KGs, the connection-based approach offers explainable recommendations. Both path-based [15] and meta-structure based [70] methods allow for explanations that clearly demonstrate how recommendations are derived from relationships in KGs. Furthermore, by showing which paths were emphasized in the recommendation process, the attention mechanism naturally provides explanations [50]. For instance, Eytan et al. [12] proposed a path-based attentive recommender system, in which the self-attention is used to represent paths in the KGs and rank them based on their relevance. Besides, by combining RL, connection-based approach can learn to traverse these connections to arrive at a recommendation, which inherently provides a rationale or explanation [41,67]. Li et al. [24] designed a counterfactual path-based framework for explainable recommendations, which learns a path manipulation policy via an RL method for counterfactual reasoning. While connection-based approaches are more interpretable, they usually require searching or enumerating paths, which can be computationally expensive. Moreover, the reliance on predefined path patterns sometimes limits their generalization ability across different domains.

2.1.3. Propagation-based approach

Propagation-based approaches [45,62] aggregate multi-hop relational information in the KGs for deeper insights. The paths or weights assigned during propagation help explain the underlying reasoning behind the recommendations. For example, Liu et al. [30] proposed a socially enhanced KG-based explainable recommendation model, in which an attentive information propagation can capture the high-order semantic information.

For the propagation in the item KGs [43], the edge weights in the KGs are explicit. In this case, the salient path connecting the candidate item to the interacted item can be used to explain the recommendation results. Similarly, for the propagation in the user–item KGs [44], via the salient paths connecting the target user to the candidate item, explanations for the recommendation results can be provided. The

explanations are more intuitive, as they clearly show the contribution of each interacted item.

2.2. Evolutionary reinforcement learning

Evolutionary RL integrates evolutionary computation techniques with RL to enhance exploration, improve robustness against noisy gradients, and efficiently search for policies in complex environments [38]. Instead of relying solely on backpropagation, evolutionary RL maintains a population of candidate policies or structures that evolve through mutation, crossover, and selection. This paradigm enables evolutionary RL to explore diverse policy spaces in a parallel and gradient-free manner, providing advantages in environments with sparse feedback, non-stationarity, or non-differentiability. In general, evolutionary RL approaches can be categorized into four groups, among which evolutionary strategy–based and genetic algorithm–based methods are the most representative [27].

2.2.1. Evolutionary strategy

Evolutionary Strategy (ES)-based RL focuses on optimizing policy parameters through stochastic perturbations and fitness-driven updates [1]. ES samples candidate policies by injecting noise into parameters, evaluates their performance, and updates the search distribution toward better-performing regions. Its gradient-free nature offers high scalability and strong stability in challenging environments. For instance, Fernandez and Caarls [14] applied evolutionary computing to tune RL parameters through iterative fitness evaluation, demonstrating improved adaptability. RL-RVEA [25] further incorporated reference vector adaptation to handle multi-objective problems by combining ES principles with Q-learning. Additionally, Peng et al. [34] introduced RLHDE, which employs Q-learning to dynamically regulate mutation strategies in differential evolution, yielding superior global search capability and convergence in interplanetary trajectory design tasks.

2.2.2. Genetic algorithm

Genetic Algorithm (GA)-based methods [3,26] incorporate classical genetic operators—selection, crossover, and mutation—to explore a broad policy search space, making them effective for high-dimensional or multi-modal RL problems. By maintaining population diversity, GA-based RL enhances global exploration and mitigates the risk of premature convergence. For example, Sehgal et al. [36] adopted GA to optimize hyperparameters in deep RL models, achieving improved training stability. In multi-agent coordination, Aydeniz et al. [2] introduced a novelty-driven evolutionary strategy that encourages diverse behaviors within populations. Additionally, the DDQN-RS framework [18] used Gaussian-based random search to sample individuals and evaluate their fitness, demonstrating superior lane-keeping performance when compared to conventional Double DQN.

2.2.3. Cross-entropy method

The Cross-Entropy Method (CEM)-based approach formulates policy optimization as a probabilistic search problem [75]. CEM iteratively samples candidate solutions from a parameterized distribution, retains elite samples based on fitness, and updates the distribution's parameters to bias future sampling toward high-quality individuals. As summarized in our survey [27], CEM-based RL emphasizes elite-guided learning, where a small subset of top-performing policies drives distribution updates [8,20]. This makes CEM particularly effective for black-box optimization and sparse-reward environments. It is commonly integrated into hybrid RL frameworks to stabilize policy search and accelerate convergence in both continuous control and multi-objective scenarios.

2.2.4. Population-based training

Population-Based Training (PBT)-based RL [33] maintains multiple policies that co-evolve during training through periodic exploitation and exploration. In exploitation, weaker agents adopt the weights of stronger ones, while exploration perturbs hyperparameters to maintain diversity. This dynamic evolution enables PBT to adaptively tune learning rates, reward functions, and other hyperparameters throughout training. In multi-agent RL, population curricula have been shown to scale learning across diverse opponents and tasks [32]. Moreover, evolutionary multi-objective deep RL approaches [37] leverage population diversity to generate behaviorally rich policies. The use of PBT also appears in parameter optimization and constrained RL settings, providing robust adaptation across varying environments and reward structures [52].

2.3. Reinforcement learning-based recommender systems

As an autonomous learning algorithm, RL has attracted interest in recommender systems in recent years. In practice, RL algorithms have been applied to various recommendation scenarios [10,31,77], such as news recommendation [74], movie recommendation [72], and e-learning [28]. Depending on specific recommendation tasks, existing RL-based recommendation methods can be divided into the following four branches [29].

2.3.1. Interactive recommendation

In interactive recommender systems, RL algorithms are competent in learning the recommendation strategies from the user–item interactions [5,22]. For example, to enable user–item interactions effectively, Yu et al. [64] proposed an attribute augmented RL method to model explicit multi-modal matching. Zhang et al. [66] introduced a reward constrained recommendation method to sequentially capture the user's preferences. Thus, user feedback can be taken as a constraint, which is incorporated into the next item recommendation. Besides, a tree-structured Policy Gradient algorithm [7] was presented to alleviate the discrete action space issue in large-scale interactive recommender systems.

2.3.2. Sequential recommendation

Under sequential user–item interactions, sequential recommendation methods aim to recommend the next items that meet users' future preferences [60]. For instance, Zhao et al. [73] introduced a pairwise deep RL algorithm to learn user preferences, which simultaneously offers positive and negative feedback under sequential user–item interactions. To capture long-term user preferences, [46] adopted a knowledge-guided RL model to generate recommendations over the KGs. The agent obtained knowledge-level and sequence-level rewards when it recommended related items for users.

2.3.3. Conversational recommendation

Conversational recommender systems first interact with the specific users to learn their active feedback and then provide the corresponding recommendations for these users through multi-turn interactive conversations [21]. To grasp the user's current intention in the process of conversational recommendations, Sun et al. [39] developed a deep policy network to guide the dialogue management, in which the policy parameter is optimized using the REINFORCE algorithm. Greco et al. [16] used an HRL approach to effectively manage each phase of the dialogue module, which trains a meta-controller to understand the user utterance. In addition, Deng et al. [11] proposed a unified framework to systematically combine conversation and recommendation tasks, utilizing a graph-based RL approach to learn a unified policy for the decision-making processes in the conversational recommender system.

2.3.4. Explainable recommendation

Explainable recommendation methods focus on the explainability of recommendations or interpretable models. For example, [51] leveraged a deep RL algorithm to generate sentence explanations. An agent in the proposed model aims to generate the explanations while the

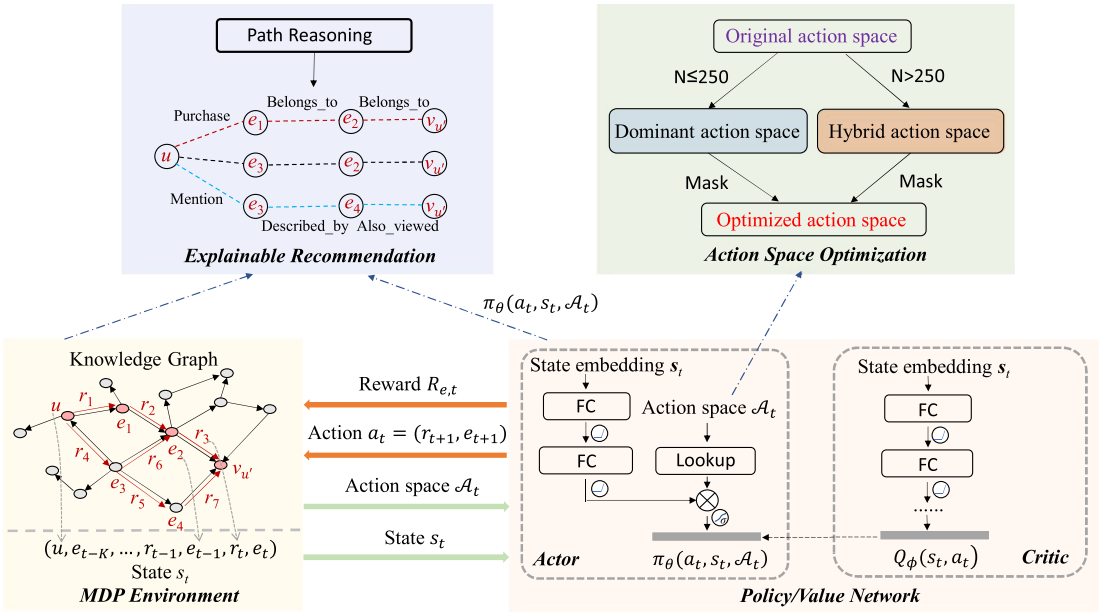


Fig. 1. The framework of ERLER.

other agent predicts the recommendation ratings based on the corresponding explanations. Besides, Xian et al. [58] proposed the PGPR framework to make recommendations and generate their interpretability. They adopted the Policy Gradient algorithm to guide the agent in seeking feasible recommendation paths over KGs. Expanding the PGPR framework, Zhang et al. [67] proposed a self-supervised RL method to conduct knowledge-aware recommendation reasoning on the KGs. Differing from the PGPR framework, Zhao et al. [71] presented an adversarial Actor-Critic method to achieve more accurate recommendations. In particular, it utilizes an adversarial imitation learning method to guide the pathfinding.

However, in large-scale KGs, existing RL-based recommendation models usually face key challenges in path reasoning convergence and the quality of candidate actions. The vast number of potential actions can lead RL agents to converge on suboptimal solutions, making it difficult to discover more effective paths, which affects recommendation relevance and interpretability. To this end, we propose a new framework, namely ERLER, that combines evolutionary RL with KGs to accelerate the convergence of path reasoning. Our ERLER incrementally adapts to evolve policies that balance recommendation accuracy and explainability by optimizing explainable paths in the KGs.

3. Methodology

3.1. Overview

As shown in Fig. 1, ERLER establishes a recommendation framework based on the Markov Decision Process (MDP) over a KG. Unlike conventional RL-based methods, ERLER leverages evolutionary mechanisms to dynamically optimize the action space, thereby accelerating convergence and improving interpretability. The MDP environment encodes user states and candidate actions derived from KG relations, while the Actor–Critic network guides policy learning with an entropy-based adjustment to avoid premature convergence. In parallel, a mutation-driven operation refines the action space, ensuring both efficiency and diversity in reasoning. Through this design, ERLER produces explainable recommendation paths that connect users to items in a transparent manner. Fig. 2 shows the flowchart of our algorithm.

3.2. Preliminaries

3.2.1. Knowledge graph

The knowledge graph in ERLER is denoted as $G = \{(e_{head}, r, e_{tail}) | e_{head}, e_{tail} \in E, r \in \mathcal{R}\}$, where E contains user entities U and item entities I. These relational structures not only represent semantic connections but also define the action space for the RL agent. Unlike a static background resource, the KG in ERLER functions as the basis for generating and optimizing recommendation paths. To effectively manage large action spaces, ERLER employs a mutation mechanism that prunes redundant actions and introduces controlled variability, resulting in a hybrid action space. This enables the system to balance exploration with efficiency, making the reasoning process both effective and explainable.

3.2.2. Markov decision process

In the realm of the decision-making process, the MDP integrates five fundamental components: state, action space, state transition probability, discount factor, and reward.

State. Given a user $u \in U$, an entity at time step t, e_t, and historical data, h_t, the state S_t is characterized as a triplet (u, e_t, h_t). An n-step history is defined as the path sequence aggregating entities and relations over the preceding n steps, presented as $\{e_{t-n}, r_{t-n}, \ldots, e_{t-1}, r_{t-1}\}$. The initialization of a user's state is rendered as $S_0 = (u, u, \varnothing)$, where $\varnothing$ denotes an empty set.

Action Space. Action is defined as the step an entity takes at a given state, denoted as s_t, to transition to another state. The action, a_t, is taken by the entity e_t to reach the next entity e_{t+1}. The set of all possible actions at this juncture is known as the action space A_t, which is formally described by the set $A_t = \{(r, e) | (e_r, r, e) \in \mathcal{G}_\mathcal{R}, e \notin \{e_0, \ldots, e_{t-1}\}\}$. Here, $\mathcal{G}_\mathcal{R}$ is the relational graph that includes all the possible transitions between entities.

Generally, the action space may contain many candidate actions, which are sorted according to some standards (e.g., score). To avoid the computational complexity caused by a large action space while maintaining diversity, we proposed an action space operation combined with "mutation". First, given the action space size N and the maximum number of actions M, if $N < M$, all candidate actions will be selected, and the corresponding action space will be considered as the dominant action space according to Fig. 1. Otherwise, the action space size needs to

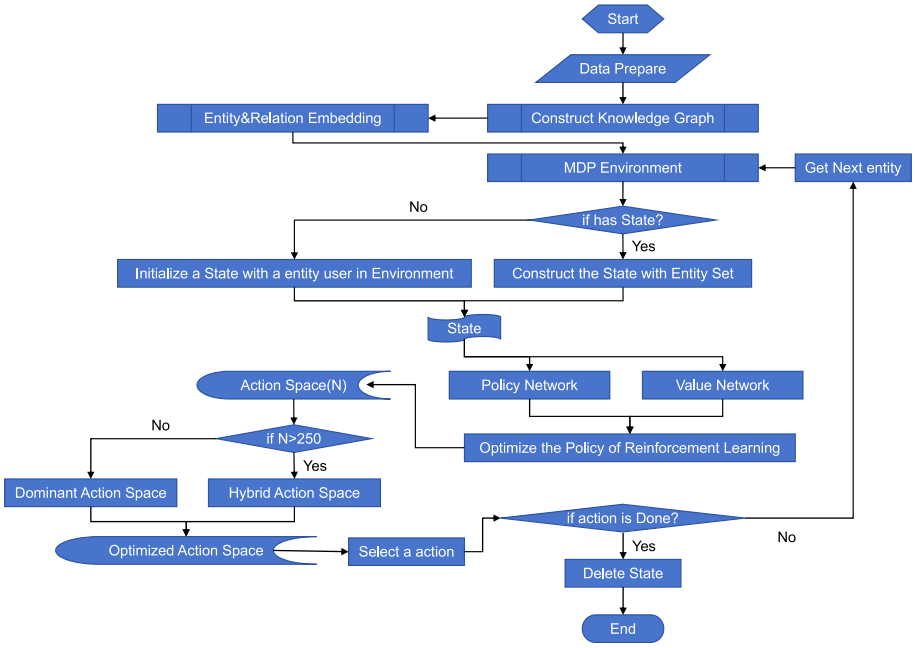


Fig. 2. The algorithm flowchart of ERLER.

be controlled. The maximum mutation amount C is defined as the number of redundant actions, calculated as $C = N - M$. The corresponding mutation rate is defined as:

$$\begin{aligned} \eta_{\max} &= \frac{C}{M} \\ \eta_{\text{actual}} &= \min(\eta_{\text{custom}}, \eta_{\max}) \\ m &= \lfloor \eta_{\text{actual}} \times M \rfloor, \end{aligned} \tag{1}$$

where η_{custom} is a custom mutation rate. The lesser of η_{custom} and $\eta_{\max}$ will be the final mutation rate η_{actual}, and m stands for the number of actions that will actually mutate. The custom mutation rate is determined by a formula that adjusts based on the population size:

$$\eta_{\text{custom}} = \frac{\beta}{N}, \tag{2}$$

where β is a constant that controls the mutation rate. In the experiments we conducted, β is set to 30. This calculation allows the mutation rate to scale inversely with the size of the action space.

In the mutation process, m actions are randomly selected and tagged as replaceable from the top-M actions. Meanwhile, the top-m actions are chosen from the redundant actions to replace the previously tagged actions. After mutation, a new action space called hybrid action space is obtained, containing no more than M actions, which serve as the final candidates.

State Transition Probability. In the context of the MDP, the current state is $S_t = (u, e_t, h_t)$. Upon taking the action $a_t = (r_{t+1}, e_{t+1})$, the agent transitions to the next state. The probability of this state transition is defined as:

$$\mathbb{P}[S_{t+1} = (u, e_{t+1}, h_{t+1}) | S_t = (u, e_t, h_t), a_t = (r_{t+1}, e_{t+1})], \tag{3}$$

where $\mathbb{P}$ represents the probability; S_t and S_{t+1} are the states at consecutive time steps; a_t is the action taken at state S_t, and (u, e_t, h_t) and (u, e_{t+1}, h_{t+1}) represent the user u, entity e, and history h at times t and $t+1$, respectively.

Discount Factor. The discount factor γ, ranging between 0 and 1, balances the importance of immediate versus future rewards.

Reward. The terminal reward R_T is computed using a soft reward mechanism based on a multi-hop scoring function $f(u, i)$. This function evaluates the desirability of recommending item i to user u. The terminal reward R_T is calculated as follows:

$$R_T = \begin{cases} \max\left(0, \frac{f(u, e_T)}{\max_{i \in I} f(u, i)}\right), & \text{if } e_T \in I \\ 0, & \text{otherwise,} \end{cases} \tag{4}$$

where R_T is normalized within the range $[0, 1]$. The set I contains all items, and e_T is the entity (item) at the terminal state. This normalization ensures that the reward reflects the relative quality of the terminal entity's score compared to the best possible score for the user u within the item set I.

To avoid the pitfalls of short-term or single rewards, which may lead to local optima, a long-term incremental evaluation is employed. This evaluation integrates Temporal Difference (TD) increments across various future states, which allows for a more comprehensive assessment of the policy's performance over time.

Based on the terminal reward R_T, a reward accumulation mechanism that dynamically adjusts the discount factor γ is introduced. This approach accounts for varying levels of importance across different time steps in the sequence, providing a more nuanced method for reward calculation. The accumulated reward G_t at each time step t is defined as follows:

$$\begin{aligned} G_t &= R_t + \sum_{k=t+1}^{T} \gamma_k R_k, \\ \gamma_k &= \gamma_{k-1} - (\gamma_{k-1} - \gamma_{\text{end}}) \times \frac{k}{T}, \end{aligned} \tag{5}$$

where γ_k represents the discount factor at time step k, gradually decreasing and approaching the terminal discount value γ_{end}. This approach ensures that the reward in the early period is relatively large, while it becomes smaller in the later stages.

3.2.3. Multi-hop scoring function

In exploring multi-hop scoring functions to refine the action space and enhance reward calculations in KGs, we present a scoring function,

$f(e_0, e_k|\tilde{r}_{k,i})$, designed to include reverse edges within KG paths. The function is defined as:

$$f(e_0, e_k|\tilde{r}_{k,i}) = \langle e_0 + \sum_{s=1}^{i} r_s, e_k + \sum_{s=i+1}^{k} r_s \rangle + b_{e_k}, \tag{6}$$

where $\tilde{r}_{k,i}$ represents a 1-reverse k-hop pattern, where the sequence spans from e_0 to e_k with possible directional shifts at each (r_i). The dot product $\langle \cdot, \cdot \rangle$ integrates the embedding vectors of entities e and relations r, alongside a bias term b_{e_k} for the terminal entity. This adaptive formula calculates both immediate and terminal rewards by adjusting k and i, tailoring it to various relational contexts within the KGs.

For action pruning and reward assessment, this formulation leverages the inherent structure of KG, where the types of entities and relations dictate the subsequent entity types, to efficiently navigate the action space. This process involves selecting the smallest k that validates a 1-reverse k-hop pattern between a user–item pair, optimizing action selection based on the dynamic interplay of forward and backward relations. The overarching goal is to improve the accuracy of predicting relationships between entities by training their embedding vectors. This is accomplished through a negative sampling strategy that enhances computational efficiency, achieving a balance between theoretical depth and practical utility in the KG applications. This streamlined approach ensures a focused and effective utilization of the multi-hop scoring function, emphasizing its critical role in KG analysis and application.

3.2.4. Actor-critic network

Actor-Critic network integrates policy optimization and value function estimation to learn a policy that can maximize cumulative rewards in given environments. Actor-Critic network primarily consists of two parts: actor and critic.

Actor Network. Actor is responsible for generating actions. It outputs an action from an action probability distribution according to the current state. The target of the actor is to find the action that brings higher rewards. Generally, actor is realized by a parameterized policy function, the policy of actor is defined as:

$$\pi(a_t, s_t; \theta_\pi) = \text{softmax}(\theta_\pi^T f(s_t)), \tag{7}$$

where $\pi(a_t, s_t; \theta_\pi)$ denotes the probability of choosing action a_t under state s_t; θ_π signifies the parameter of policy network; $f(s_t)$ indicates the feature representation of state s_t.

Critic Network. The critic's role is to evaluate the actions of the actor by learning a value function to estimate the expected return for taking an action in a particular state. The output of critic is a value function $Q(s_t, a_t; \theta_Q)$:

$$Q(s_t, a_t; \theta_Q) = r_t + \gamma \mathbb{E}_\pi \left[Q(s_{t+1}, a_{t+1}; \theta_Q) \right], \tag{8}$$

where r_t is the instantaneous reward; γ denotes the discount factor; $\mathbb{E}_\pi$ represents the expectation under policy π.

In the natural world, the adaptive ability of individuals is always accompanied by the evolutionary process. Thus we propose a novel method to calculate the advantage ψ by introducing adaptive ability:

$$\psi = \left(\frac{\lambda H\left(\pi\left(s_t\right)\right)}{\log\left(\dim\left(\mathcal{A}\right)\right)} + 1 \right) \times \left(r_t - V\left(s_t\right) \right), \tag{9}$$

where $(r_t - V(s_t))$ represents the standard expression of advantage; $H(\pi(s_t))$ denotes the entropy of policy π under state s_t; $\dim(\mathcal{A})$ stands for the dimension of the action space; λ is a scaling factor that controls the influence of entropy on the advantage. In our method, $\lambda = 1$.

By introducing entropy, our method implements an adaptive mechanism that adjusts the policy update direction and magnitude dynamically based on distribution uncertainty. Since entropy reflects the degree of exploration by the policy and advantage directly affects the policy's improvement margin, incorporating entropy into the advantage calculation helps avoid premature convergence to suboptimal solutions and improves the stability of the policy during training.

3.3. Policy optimization

In the field of RL, policy optimization is a key technique for adjusting the behavioral strategies of agents to accumulate more rewards within the environment. In policy optimization, we minimize the loss function for actor and critic networks as we mentioned previously.

The state s_t is first embedded and then fed into both the actor and critic networks. The initial two Fully Connected (FC) layers in these networks are parameter-sharing. The rationale behind this design is to extract generic features from the state, which lays the groundwork for subsequent policy decisions and value estimations.

The critic network processes the embedded state through an additional FC layer to output a value function $V(s_t)$, which estimates the cumulative reward associated with the state s_t. This estimated value $V(s_t)$ provides feedback to the policy, aiding the actor in refining policy choices. In our method, the loss function is based on the squared advantage ψ. This ψ is specifically an entropy-guided advantage, where the standard difference between the actual return (e.g., r_t or a target including future rewards) and the estimated value $V(s_t)$ is dynamically scaled by a term reflecting the policy's current entropy. This strategic scaling allows the critic's learning signal to be more sensitive and responsive to the outcomes of actions taken in exploratory states, thereby fostering more effective exploration by the actor:

$$\mathrm{L}_{critic} = \sum_a \psi^2. \tag{10}$$

The actor network receives the embedded state s_t and also the original action space as input. Through a sequence of genetic algorithm operations that we previously described, a new action space $\tilde{A}_{u,t}$ is derived. We formulate a policy $\pi_\theta(a_t|s_t, \tilde{A}_{u,t})$ by integrating the network's output with this new action space. This policy not only directs the agent in selecting actions but is also fed back into the loss function to inform the learning process, which can be defined as:

$$\mathrm{L}_{actor} = \mathrm{L}_\pi = -\sum_a \left(\log \pi(a|s) \cdot \psi \right), \tag{11}$$

where $\pi(a|s)$ denotes the probability of choosing action a under state s; $Q(s, a)$ represents the value function estimation of state-action pair (s, a). The target of actor network is to maximize the Q value of expectation i.e., increase the probability of selecting those high-value actions by adjusting the policies.

In the ERLER framework, policy optimization concludes with the definition of a loss function for the actor and critic networks. This function is crucial not only for enhancing learning performance but also for ensuring the explainability of the policies.

$$\mathrm{Total}_{loss} = \mathrm{L}_{actor} + \mathrm{L}_{critic} + \delta \times \mathrm{L}_{entropy}, \tag{12}$$

where L_{actor} and L_{critic} represent the respective losses of the two networks; $\mathrm{L}_{entropy}$ denotes the entropy loss, and δ refers to the weight factor for entropy loss. This last term increases the policy's exploratory capabilities, encouraging the exploration of new possibilities rather than confining the search to previously known policies.

The combined loss function serves as the learning objective, guiding the update of parameters through backpropagation across the networks. During the training process, the actor's objective is to minimize the policy-related loss, whereas the critic strives to reduce the value estimation error. By adopting this method, the ERLER model is able to incrementally refine the policy parameters θ in each iteration, thereby achieving policy optimization.

3.4. Explainable path reasoning

We propose a new approach to achieve recommendations based on KGs through a policy network. For each user u, the goal is not just to identify a candidate item set $\{i_n\}$, but ultimately to provide a clear

Algorithm 1 Training process of ERLER framework.

Input: actor network: $\pi_\theta(\mathbf{s})$, critic network: $\hat{v}(\mathrm{s})$, states: S, the learning rate: Lr, epoch: N, batch size: K, action space: A, action space size: α, gamma lower limit: γ_0, meta path, entity embedding, relation embedding.

Output: π_θ

1: initialize Actor-Critic network parameters, virtual environment of RL.
2: $n \leftarrow 0$
3: **for** n to N **do**
4: using $\pi_\theta(\mathbf{s})$
5: **for** $t \leftarrow 0$ to $(T-1)$ **do**
6: $S \leftarrow S_t$
7: $p(a) = \pi_\theta(a|S)$
8: Actions $A_t \leftarrow \{mutation(a)|rank(p(a)) \geq \alpha\}$
9: **end for**
10: generate an episode: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$
11: $G_t \leftarrow 0$
12: **for** $t \leftarrow 0$ to $(T-1)$ **do**
13: **for** $i \leftarrow (T-1)$ to t **do**
14: $G_t \leftarrow \gamma G_t + R_t$
15: $\gamma = \gamma - (\gamma - \gamma_0) \times (1/T) \times 1.0$
16: $\gamma = \max(\gamma, \gamma_0)$
17: **end for**
18: $\psi = \left(\frac{\lambda H(\pi(s_t))}{\log(\dim(\mathcal{A}))} + 1\right) \times \left(r_t - V\left(s_t\right)\right)$
19: $\mathrm{L}_{critic} = \sum_a \psi^2$
20: $\mathrm{L}_{actor} = -\log\left(\pi_\theta(S)\right) \times \psi$
21: $\mathrm{L}_{entropy} = -H(\pi_\theta)$
22: $\mathrm{Total}_{loss} = \mathrm{L}_{actor} + \mathrm{L}_{critic} + \delta \times \mathrm{L}_{entropy}$
23: **end for**
24: $\nabla_\theta J(\theta) = E_\pi\left[\nabla_\theta \log \pi_\theta(S) \times \psi\right]$
25: **end for**
26: **return** π_θ

Algorithm 2 Mutation function for action selection.

Input: action set A, total action count N, action scores a_s, max actions to keep M, mutation rate η_{custom}

Output: selected action indices after mutation: A_η

1: **if** $N \leq M$ **then**
2: **return** A
3: **end if**
4: $m_{\max} \leftarrow N - M$ # max number of actions that can be mutated
5: $\eta_{\max} \leftarrow \frac{m_{\max}}{M}$
6: $\eta \leftarrow \min(\eta_{\text{custom}}, \eta_{\max})$
7: $m \leftarrow \lfloor \eta \times M \rfloor$ # number of actions to mutate
8: $A_{\text{top}} \leftarrow A[-M:]$ # top-M actions based on a_s
9: $A_{\text{rest}} \leftarrow A[:-M]$ # remaining actions
10: $\mathcal{I}_{\text{mut}} \leftarrow \text{random\_sample}(A_{\text{top}}, m)$
11: $\mathcal{I}_{\text{high}} \leftarrow A_{\text{rest}}[-m:]$ # top-m from rest
12: **for** $i \leftarrow 0$ to $m-1$ **do**
13: $a_{\text{orig}} \leftarrow \mathcal{I}_{\text{mut}}[i]$
14: $a_{\text{new}} \leftarrow \mathcal{I}_{\text{high}}[i]$
15: $\text{pos} \leftarrow \text{find\_position}(A_{\text{top}}, a_{\text{orig}})$
16: $A_{\text{top}}[\text{pos}] \leftarrow a_{\text{new}}$
17: **end for**
18: **return** $A_\eta \leftarrow A_{\text{top}}$

and explainable reasoning path $\{p_n(u, i_n)\}$. Although the policy network $\pi\left(\cdot|s, \widetilde{A}_u\right)$ generates recommendation paths for users, it tends to select those paths that bring the highest cumulative rewards, potentially overlooking other paths.

Inspired by Markov boundary discovery algorithm [54,55], to enhance the explainability of recommendations, we introduced a beam search, which combines action probability and reward. This approach not only helps identify items the user may be interested in but also ensures the transparency of the recommendation process. During path generation, the policy network creates multiple candidate path sets $\mathcal{P}_T$ based on predefined horizon T and the sampling sizes $\{K_1, K_2, \ldots, K_T\}$ for each step. These path sets describe how a user u ultimately reaches the recommended item from the initial state. Additionally, the path sets assign a generative probability $\mathcal{Q}_T$ and a corresponding reward $\mathcal{R}_T$ to each path. The reasoning chains provided by these paths form the core explanation for the recommendation results. Furthermore, the path finally selected by the model is not only based on the path's score or reward but also on the probability during the generation process, ensuring the rationality and transparency of the recommendation decision. The generated path allows the recommender systems to clearly show the causal relationship behind each recommendation decision.

3.5. Model training

The parameters of the Actor-Critic network and the virtual RL environment are initialized in the training process of ERLER. The training is executed in multiple rounds iteratively. The Actor network $\pi_\theta(\mathbf{s})$ is used to generate actions for each state. In each iteration, ERLER selects actions based on the current state through the Actor network and obtains new rewards and states from executing these actions (corresponding to lines 5–7 of Algorithm 1). During this process, ERLER applies a mutation mechanism, which mutates the generated actions through Eqs. (1) and (2), and finally obtains the optimal action set A_t (corresponding to line 8 of Algorithm 1), thereby improving the recommendation performance. Furthermore, ERLER generates the complete state-action-reward sequence (corresponding to line 10 of Algorithm 1) and updates the future cumulative reward by adjusting G_t. During this update, the dynamic discount factor γ is gradually adjusted (corresponding to lines 15 to 16 of Algorithm 1). The initial value is γ_0, and Eq. (5) is applied to adjust the discount factor. The expression $\gamma = \max(\gamma, \gamma_0)$ ensures that the discount factor does not fall below the predefined bound, thus balancing short-term and long-term rewards. After the loop, the Critic network calculates the state value $V(s_t)$ and evaluates the performance of the Actor network using the adjusted Advantage ψ (corresponding to line 18 of Algorithm 1). The adjusted Advantage not only considers the difference between the state and action values but also introduces entropy to enhance exploration diversity. The total loss function combines actor, critic, and entropy losses (corresponding to line 22 of Algorithm 1), which is optimized through gradient descent, gradually refining the recommendation policy.

To enhance the diversity and effectiveness of action selection, ERLER introduces a mutation function as detailed in Algorithm 2. This mutation function is applied when the number of available candidate actions N exceeds the maximum number of actions M that can be retained. It selectively mutates a subset of the top-M actions with other high-scoring actions outside this top set to explore a broader action space. Specifically, the mutation rate η_{custom} controls the proportion of actions to be mutated. The algorithm first computes the upper bound of the mutation rate based on the difference between N and M (corresponding to line 4 of Algorithm 2). Then, it samples m actions from the top-M set to be replaced and selects m alternative actions from the remaining candidates based on their scores (lines 7–11). These replacements are made in-place (lines 12–16), yielding a refined action set A_η that incorporates both high-score exploitation and exploration via mutation.

The time complexity of the ERLER training process is mainly determined by the nested loops in Algorithm 1. For each of the N epochs, the generation of an episode requires $O(T)$ operations, while the backward return calculation involves a double loop over T steps, leading to $O(T^2)$. Consequently, the overall complexity is $O(NT^2)$, which dominates the training cost. The mutation step contributes only $O(N)$ per action selection and does not alter the asymptotic bound.

Table 1
Statistics of datasets applied.

	Beauty	Clothing	Cell
Number of users	22,363	39,387	27,879
Number of items	12,101	23,033	10,429
Relation types	8	8	8
Entity types	5	5	5
Relations	Beauty	Clothing	Cell
Purchase	8.88±8.16	7.08±3.59	6.97±4.55
Mention	806.89±1344.08	440.20±452.38	652.08±1335.76
Described_by	1,491.16±2,553.93	752.75±909.42	1743.16±3482.76
Belong_to	4.11±0.70	6.72±2.15	3.49±1.08
Produced_by	0.83±0.38	0.17±0.38	0.52±0.50
Also_bought	73.65±30.69	61.35±32.99	56.53±35.82
Also_viewed	12.84±8.97	6.29±6.17	1.24±4.29
Bought_together	0.75±0.72	0.69±0.90	0.81±0.77

4. Experiment

4.1. Experimental setup

4.1.1. Datasets

We conducted our experiments using three real-world datasets from the Amazon Review Data repository[2]: Beauty, Clothing and Cell. These datasets, which include product reviews, metadata, and associated links, were sourced from Amazon. Each dataset features five entity types and eight relationship types. Detailed statistics for these datasets are listed in Table 1. In our method, the datasets were split into training and testing sets, following a 3:1 ratio.

4.1.2. Baseline models

We compared our ERLER with the following recommendation baselines.

BPR [35] is a model designed to learn user preferences through pairwise comparisons, primarily relying on user-item interactions to predict interests.

DeepCoNN [76] utilizes a convolutional recommendation model to encode users and products based on review data. It focuses on extracting features from product reviews to improve recommendation accuracy, demonstrating the usefulness of leveraging review information in recommendation systems.

CKE [68] is a framework designed to combine collaborative filtering with KG embedding. It integrates structured information from KGs with latent factor models to enhance recommendation performance by capturing richer semantic relationships between users and items.

JRL [69] is a framework for top-N recommendations that integrates multiple heterogeneous information sources (e.g., reviews, images, and ratings) into a shared representation space.

PGPR [59] is a model that focuses on path reasoning over KGs. It uses RL to navigate multi-hop paths in the graph, starting from the user and ending at potential items, providing explainable reasoning paths for each recommendation.

REKS [53] is a generic reinforced explainable framework designed for session-based recommendation. It integrates session representations from black-box session-based recommendation models with knowledge graphs, using RL to explore multi-hop paths that connect session behaviors to candidate items.

SSRL [67] improves path reasoning by using an enhanced Actor-Critic algorithm with a dual-reward strategy that combines short-term and long-term rewards.

4.1.3. Evaluation metrics

In this evaluation, we assessed the recommendation quality using several key metrics: Precision, Normalized Discounted Cumulative Gain (NDCG), and Hit Ratio (HR) for the top-10 recommendations. Suppose hit_u is the number of correctly recommended items in the top-K list for user u, the evaluation metrics are defined as follows:

$$Precision@K = \frac{1}{m}\sum_{u=1}^{m}\frac{hit_u}{K}, \tag{13}$$

$$NDCG@K = \frac{1}{IDCG@K}\sum_{i=1}^{K}\frac{2^{rel_i}-1}{\log_2(i+1)}, \tag{14}$$

$$HR@K = \frac{\sum_{u=1}^{m} hit_u}{\sum_{u=1}^{m} N_u}, \tag{15}$$

where m stands for the total number of users, N_u represents the number of items that user uu has rated, and $IDCG@K$ denotes the Ideal Discounted Cumulative Gain for the best possible top-K recommendations. Additionally, rel_i is the relevance score of the item at the i-th position in the recommended list.

4.1.4. Implementation details

The construction of the ERLER's MDP environment mainly refers to the PGPR [59] framework, which applies the policy of Actor-Critic network to guide agents to proceed with path searching in KGs. This foundational framework was adopted to ensure experimental consistency and fair comparison with primary baselines like PGPR and SSRL, thereby allowing our contributions to focus on advancing action selection methodologies rather than redesigning the environment itself. The embeddings' dimension of all entities and relations in KGs was set to 100, and the TransE model was utilized for 30 epochs of embedding training.

In the process of RL, the maximum size of action space M was set to 250. Next, we pruned the action space according to multi-hop score function. The corresponding prune rate is 0.5. The initial discount factor of cumulative reward is 0.99. There are two layers in Actor-Critic network, and the sizes of hidden-layer are 512 and 256, respectively. The loss function in ERLER contains actor loss, critic loss and entropy loss. The method used for initializing network was Xavier. The entropy regularization weight was set to 0.001 to encourage exploration during training. There are 50 epochs for model training, Adam was selected as an optimizer, the learning rate is 0.0001, and the batch size is 32. Additionally, a dropout rate of 0.5 was applied to each hidden layer to prevent overfitting during training.

All experiments were conducted on the same hardware platform, to ensure consistency.

4.2. Performance comparison

After conducting several experiments on three different datasets with 6 previous models and the proposed ERLER model, the experimental results are shown in Tables 2–4. The evaluation metrics of the 6 models are roughly sorted in ascending order, with our ERLER model's performance listed at the bottom of the table. ERLER performs well on most datasets and evaluation metrics, consistently outperforming the baseline model (SSRL). Although it falls slightly behind the baseline model in one metric, the gap is minimal, demonstrating that the overall performance of ERLER exhibits relatively good robustness.

In the Clothing dataset, as shown in Table 2, ERLER performs well across all evaluation metrics. Notably, in Precision and HR, ERLER provides improvements of 2.28 % and 2.02 % compared to previous State-of-the-Art (SOTA) results. However, its NDCG is approximately 4.38 % lower than that of REKS. This indicates that, although ERLER may not outperform REKS in terms of NDCG, it achieves higher HR and Precision, demonstrating strong overall competitiveness. The superior NDCG of REKS can be mainly attributed to its use of a dedicated rank-level reward function and the incorporation of product-level auxiliary reasoning, both of which specifically enhance its ranking optimization. In contrast, our approach focuses more on action space diversity, which benefits other evaluation metrics.

[2] https://nijianmo.github.io/amazon/

Table 2
Comparison of model performance on **Amazon Clothing** dataset, with mean±std and corresponding p-values. The results are reported in percentage (%). The Imp. (%) indicates the percentage of improvement compared to previous SOTA result. The best result of each evaluation metric is presented in bold. The previous SOTA results are indicated by *.

Models	Precision	p-value	HR	p-value	NDCG	p-value
BPR	0.188±0.004	1.85 E-13	1.798±0.030	3.50 E-12	0.608±0.014	1.28 E-15
DeepCoNN	0.232±0.010	1.48 E-12	3.343±0.052	6.31 E-11	1.324±0.010	1.55 E-14
CKE	0.394±0.007	1.29 E-11	4.349±0.047	6.63 E-10	1.518±0.013	5.93 E-14
JRL	0.449±0.001	1.87 E-11	4.715±0.067	2.54 E-09	1.754±0.032	4.17 E-12
PGPR	0.739±0.006	9.26 E-01	7.142±0.090	5.78 E-01	2.889±0.041	5.35 E-02
REKS	0.599±0.059	6.00 E-04	5.992±0.096	1.15 E-06	**3.107±0.052***	2.03 E-04
SSRL	0.746±0.013*	–	7.195±0.183*	–	2.938±0.026	–
ERLER	**0.763±0.015**	9.49 E-02	**7.340±0.101**	1.60 E-01	2.971±0.022	6.52 E-02
Imp. (%)	+2.28	–	+2.02	–	−4.38	–

Table 3
Comparison of model performance on **Amazon Beauty** dataset, with mean±std and corresponding p-values. The results are reported in percentage (%). The Imp. (%) indicates the percentage of improvement compared to previous SOTA result. The best result of each evaluation metric is presented in bold. The previous SOTA results are indicated by *.

Models	Precision	p-value	HR	p-value	NDCG	p-value
BPR	1.154±0.011	7.03 E-13	8.291±0.069	9.66 E-12	2.816±0.024	8.52 E-14
DeepCoNN	1.211±0.018	1.59 E-11	9.866±0.167	3.46 E-10	3.436±0.059	5.87 E-12
CKE	1.384±0.012	4.27 E-11	11.110±0.144	2.71 E-09	3.802±0.037	5.68 E-12
JRL	1.560±0.025	5.41 E-07	12.853±0.131	5.14 E-07	4.497±0.102	1.89 E-08
PGPR	1.723±0.024	7.62 E-01	14.488±0.136	8.22 E-01	5.574±0.041	4.03 E-02
REKS	1.119±0.058	1.21 E-08	11.189±0.107	1.95 E-09	**6.124±0.064***	1.69 E-06
SSRL	1.741±0.012*	–	14.601±0.237*	–	5.650±0.056	–
ERLER	**1.784±0.016**	1.26 E-03	**14.785±0.247**	2.63 E-01	5.654±0.127	9.53 E-01
Imp. (%)	+2.47	–	+1.26	–	−7.67	–

Table 4
Comparison of model performance on **Amazon Cell** dataset, with mean±std and corresponding p-values. The results are reported in percentage (%). The Imp. (%) indicates the percentage of improvement compared to previous SOTA result. The best result of each evaluation metric is presented in bold. The previous SOTA results are indicated by *.

Models	Precision	p-value	HR	p-value	NDCG	p-value
BPR	0.407±0.007	4.84 E-12	4.446±0.037	8.73 E-14	1.515±0.035	1.07 E-13
DeepCoNN	0.683±0.012	3.74 E-08	7.026±0.025	6.20 E-10	2.756±0.028	4.71 E-11
CKE	0.732±0.016	2.23 E-06	7.661±0.106	1.84 E-05	3.028±0.031	1.14 E-09
JRL	0.749±0.009	1.01 E-06	7.754±0.085	2.76 E-05	3.308±0.024	5.24 E-08
PGPR	0.871±0.023*	7.38 E-02	8.437±0.087	9.00 E-04	3.822±0.060	8.09 E-04
REKS	0.854±0.061	7.86 E-01	8.539±0.022*	4.94 E-06	4.619±0.069*	2.82 E-09
SSRL	0.846±0.014	–	8.178±0.072	–	3.662±0.083	–
ERLER	**1.316±0.016**	3.37 E-11	**12.336±0.212**	1.25 E-10	**5.187±0.037**	2.29 E-12
Imp. (%)	+51.09	–	+44.47	–	+12.30	–

Regarding the Beauty dataset, as shown in Table 3, ERLER maintains a leading position, with Precision outperforming by 2.47 % and HR by 1.26 % compared to SOTA results. Consistent with the Clothing dataset, NDCG for ERLER shows a 7.67 % decrease compared to REKS, but its overall competitiveness remains strong.

As for the Cell dataset, as shown in Table 4, ERLER achieves notably greater improvements. Specifically, in the metrics of Precision, HR, and NDCG, ERLER outperforms the previous SOTA results by 51.09 %, 44.47 %, and 12.30 %, respectively. To ensure the reliability of these substantial improvements, we re-conducted the experiments after carefully reviewing all parameter settings, and the new results closely matched the original ones. Although the performance of the comparison models in our experimental environment differs slightly from the results reported in their original papers, all models were run under the same hardware and software conditions. Therefore, ERLER's significant improvement on the Cell dataset is legitimate and reasonable.

The superior performance on Cell can be attributed not only to the design of our model structure and optimization strategies, but also to the specific characteristics of the dataset (i.e., dataset size and sparsity). Firstly, according to Table 1, the Cell dataset exhibits a higher frequency of user interactions compared to the Clothing and Beauty datasets. Besides, the entity distribution in the Cell dataset is generally sparser. This sparsity necessitates identifying the correct path from a larger number of candidate paths using fewer entities — a more challenging task than in the other two datasets. Secondly, in terms of entity-relation structure, relationships with user as the head entity are predominantly *Mention* type, while those with item as the head entity are mainly *Described_by*. This imbalance introduces bias during the embedding of entities and relations in the initial MDP environment. As a result, the model tends to favor these frequent relation types, leading to higher scores for *Mention* and *Described_by* and thus prioritizing them during action selection. Overall, by optimizing the action space variation, particularly in the Cell dataset that contains a higher proportion of these relation types and is more sparse, our model can enhance the likelihood of selecting lower-ranked but potentially more informative actions. This approach leads to a more significant improvement in recommendation accuracy in the Cell dataset compared to the other datasets.

Moreover, Fig. 3 shows the comparison of the average rewards of ERLER and SSRL within 50 epochs on the Beauty and Cell datasets. In most epochs, regardless of the dataset, ERLER demonstrated significant superiority compared to SSRL. This superiority can be attributed, in part, to the novel loss function definition of the critic network, which

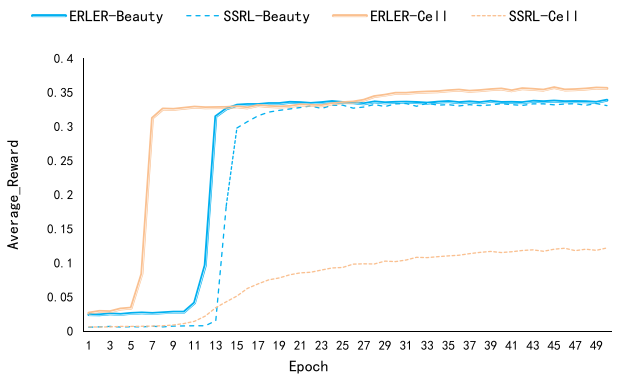


Fig. 3. Average reward per epoch of ERLER vs. SSRL.

enables the actor to execute actions yielding higher rewards. Crucially, the enhanced convergence speed observed in ERLER, as depicted in the comparisons, underscores the benefits of our model's strategic emphasis on enriching action diversity. This deliberate optimization of the action space utilized by ERLER provides a broader spectrum of choices, thereby substantially improving its exploration capability and allowing it to more efficiently navigate towards optimal solutions. This improvement is particularly notable on the Cell dataset, where ERLER not only achieves a significantly higher average reward but also exhibits a substantial lead in convergence speed over SSRL, further verifying the conclusion mentioned earlier that ERLER shows the greatest improvement over the benchmark algorithm on the Cell dataset.

4.3. Ablation study

To validate the significance of the proposed three innovations in ERLER, we conducted an ablation study by designing three variants of ERLER and analyzed their performance on the Beauty and Clothing datasets, as shown in Fig. 4.

- **ERLER-1** removes the dynamic discount factor γ and only applies a fixed discount factor to process path reasoning. In both datasets, ERLER-1 performs the best among all variants, ranking second only to the complete ERLER. This indicates that although the fixed discount factor limits flexibility at different stages of the reasoning process, it still provides relative robustness in certain circumstances.
- **ERLER-2** ignores the re-defined advantage. The ablation experiment results show that ERLER-2 performs the worst among all variants, especially in the HR of the Beauty dataset, where it shows a relatively large gap compared to the other two variants. This demonstrates that the modified advantage is crucial for accurately estimating rewards at each step of the reasoning process, and the traditional calculation of advantage struggles to capture the complex relations within paths.
- **ERLER-3** does not apply the proposed action space optimization. The performance of ERLER-3 is in the middle of the three variants. The gap between ERLER-3 and ERLER-1 on the Clothing dataset is smaller than in other evaluation metrics and datasets. This suggests that action space optimization benefits recommendation performance, with improvements varying across different datasets.

As shown in Fig. 4, the performance of the complete ERLER is significantly better than all variants across both datasets, proving that the innovations contribute to the performance improvement. e.g., on the Beauty dataset, the HR of ERLER increased by 1.5 %, 4.2 %, and 3.1 % compared with ERLER-1, ERLER-2, and ERLER-3, respectively. On the

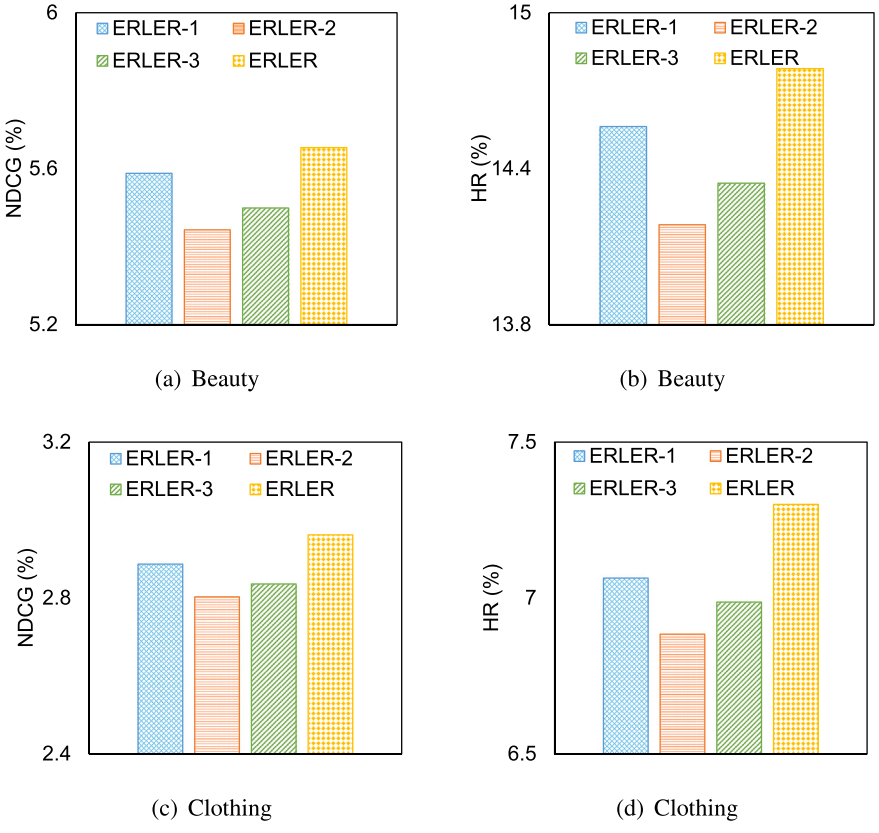
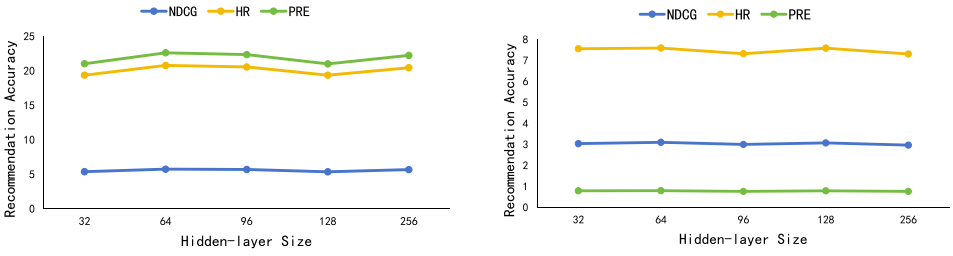


Fig. 4. The recommendation performances achieved by distinct variants of ERLER on Beauty and Clothing datasets.



(a) Beauty (b) Clothing

Fig. 5. Recommendation performance of ERLER vs. hidden-layer size on Beauty and Clothing datasets.

Clothing dataset, the HR of ERLER improved by 3.3 %, 6.0 %, and 4.5 %. Similarly, the NDCG improvement of ERLER is 1.2 %, 3.9 %, and 2.8 % on the Beauty dataset, while on the Clothing dataset, the improvements were 2.6 %, 5.7 %, and 4.4 %.

4.4. Parameters sensitivity

Fig. 5 illustrates the performance of ERLER on the Beauty and Clothing datasets under different hidden-layer sizes. The results show that when the hidden-layer size is set to 64, all evaluation metrics for both datasets reach an optimal point. After this, as the hidden-layer size increases, the performance gradually declines. However, when the hidden-layer size reaches 256, the performance returns to values similar to those observed when the hidden-layer size was set to 64. Therefore, a possible choice for the hidden-layer size could be either 64 or 256.

Fig. 6 presents the average rewards of ERLER at different epoch values, aiming to determine the impact of epoch selection on its performance. In Figure Fig. 6(a) and (c), which correspond to the Beauty and Cell datasets, the optimal epoch choice is around 50. After this point, there is no significant performance improvement, indicating stable convergence. In contrast, for Figure Fig. 6(b), which corresponds to the Clothing dataset, the performance stabilizes after around 20 epochs. Therefore, it can be concluded that the optimal epoch choice for the Clothing dataset is around 20.

Fig. 7 demonstrates the influence of constant β selection on recommendation accuracy across three datasets. As mentioned before, β controls the mutation rate η_{custom}. Normally, mutation rate refers to how many population percentages would mutate randomly. The purpose of mutation is to introduce a certain level of randomness, to help the algorithm escape local optimal solution, and seek greater diversity.

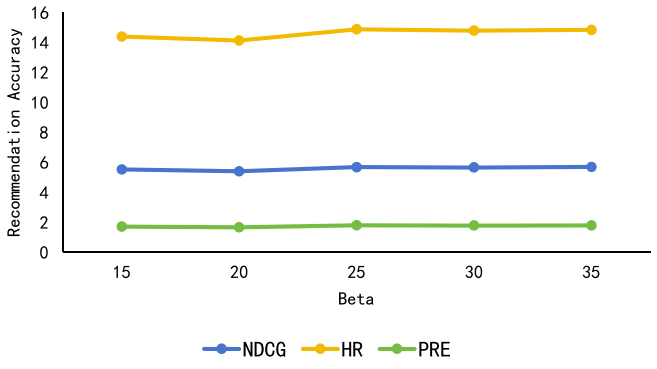
According to the experience, the range of β is selected as $\{\beta|\beta = 15 + 5k, k \in \mathbb{Z}, 0 \leq k \leq 4\}$, that is, $\beta \in \{15, 20, 25, 30, 35\}$. Hence, the results of β sensitivity are shown in Fig. 7. Fig. 7(a) indicates that the recommendation performance becomes stable when β reaches 25. Fig. 7(b) presents that there is a valley between $\beta = 20$ and $\beta = 30$, and then the performance decreases when β becomes large. The HR of Cell dataset in Fig. 7(c) shows an obvious trend of increase before β reaches 25, while Precision remains stable at all times. For most evaluation metrics in most datasets, $\beta = 30$ has been the final choice. Too small β selection may lead to insufficient diversity being introduced, increasing the risk of falling into local optimality. However, too high β selection will make the algorithm biased toward Random Search, which will disrupt the structured information of the population and reduce the evolutionary efficiency. Thus, $\beta = 30$ is a recognized suitable parameter for ERLER.

Besides, we conducted a parameter sensitivity analysis on the action space size using the Beauty and Clothing datasets, with the action space ranging from 200 to 350. The setting of 250 was used in our main experiments and is also consistent with the configurations adopted by baseline methods for comparison. As shown in Fig. 8, the trend across all metrics shows a moderate rise from 200 to 250, followed by a decline at 300, and a slight rebound at 350, with the rebound being less pronounced than the initial peak. On the Beauty dataset, the HR, Precision, and NDCG all

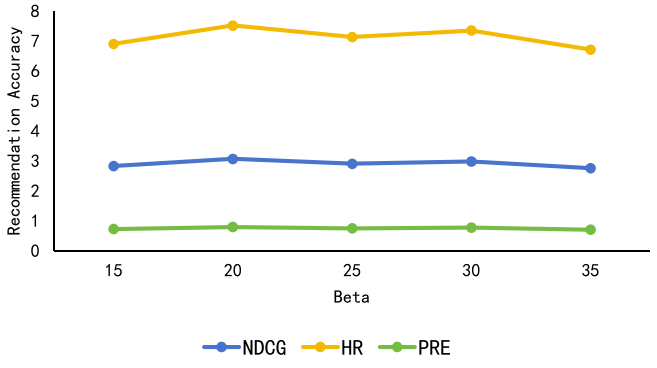
(a) Beauty (b) Clothing (c) Cell

Fig. 6. Average reward of ERLER vs. epochs.

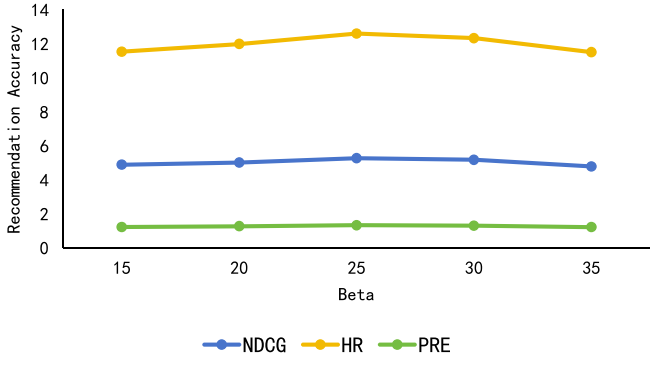
achieve their best performance when the action space size is set to 250. For the Clothing dataset, the results for 200 and 250 are nearly identical, both representing the optimal choices among the tested values. Taking



(a) Beauty



(b) Clothing



(c) Cell

Fig. 7. Recommendation Performance of ERLER vs. β.

both effectiveness and consistency into account, we select 250 as the action space size in our final experiments.

4.5. Robustness test

The ERLER model relies on path inference to predict connections between users and target items, with the entire process constrained by multiple expert-guided paths. As shown in Table 1, certain actions occur with relatively low frequency. If these low-frequency actions are excessively lost or incorrectly recorded, the corresponding inference paths may break. Consequently, some users might receive recommendations based on erroneous paths.

We designed two types of experiments to evaluate such effects. The first involves undifferentiated random loss by dropping [10, 20, 30, 40] actions at each step, while the second targets only actions with frequencies above 250 per step. The results are as follows.

Undifferentiated random loss: This leads to a reduction in already sparse action spaces, breaking critical inference paths and preventing the model from reaching the target item. Consequently, as shown in Table 5, the performance of the recommendation decreases.

Partial random loss (high-frequency actions only): As shown in Table 6, the recommendation performance under this setting is significantly higher than under undifferentiated loss in most cases. This is because our model avoids disrupting sparse action spaces, thus maintaining path connectivity. Moreover, since high-frequency actions (e.g., *Mention* and *Described_by*) dominate the action distribution, randomly removing a portion of these actually encourages exploration of alternative paths. Combined with our mutation strategy, which increases the probability of selection of underrepresented actions, the performance of the recommendation improves.

4.6. Case study

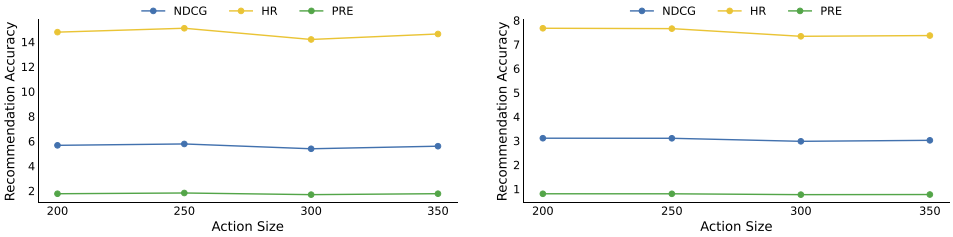
Some case studies shown as Fig. 9 are provided to better present the process of generating recommendation paths. These paths not only reflect the direct relationship between users and items but also reveal the hidden users' behavior modes, e.g., bought together, click history, and product categories. By visualizing the mentioned cases, the decisions based on users' historical behaviors and item characteristics made by the system can be better understood. Each node and edge in the system represents the basis of recommendation decision, which enhances the explainability of ERLER.

Path Patterns. There are two classifications of our path patterns. The first one refers to the path generated by users' purchasing behavior. The other one is introduced by some mentioned features from users' review text. To summarize, nine distinct path patterns were applied in the model training, which represent the complicated mechanism behind the recommender system.

Case 1. In this case, a user purchased a movie from the "Movies & TV" category. Utilizing the "belongs_to" relationship, the system identifies another DVD within the same category and recommends it to the user. The recommendation is driven by the path pattern $\{user \xrightarrow{purchase} item \xrightarrow{belongs\_to} category \xleftarrow{belongs\_to} item\}$, which connects the user's initial purchase to other items that share the same category. By following this path, the system is able to infer that the user's interest in one movie is likely to extend to other related items in the "Movies & TV" category, such as DVDs or similar media. This reasoning process ensures that the recommendation is not random but is based on a structured understanding of the relationships between items, offering suggestions that are both relevant and aligned with the user's potential preferences.

Case 2. In this case, a user mentioned both the keywords "storage" and "vacuum" in user's review. Based on this, the system identified that both terms are associated with a specific "container", which is described by these keywords. From this point, the system further analyzed items that are related to the container. Through the "also_viewed" relationship, the system discovered that users who viewed the container also showed interest in a "storage case". Therefore, the system recommended the "storage case" to the user. This recommendation path follows the pre-defined path pattern $\{user \xrightarrow{mention} feature \xrightarrow{described\_by} item \xrightarrow{also\_viewed} item\}$, which connects the user's keyword mentions with related items and their viewing history. By utilizing the relationships "described_by" and "also_viewed", the system is able to provide a relevant recommendation based on both content features and user behavior.

Case 3. In this example, a user purchased a "teether", which was produced by the brand "Dr. Seuss". The system then identified that



(a) Beauty (b) Clothing

Fig. 8. Recommendation performance of ERLER vs. action space size on Beauty and Clothing datasets.

Table 5
Performance under undifferentiated random action drop, i.e., randomly dropping [10, 20, 30, 40] actions, across datasets.

Dataset	Beauty			Clothing			Cell		
	NDCG	HR	Precision	NDCG	HR	Precision	NDCG	HR	Precision
Drop-10	5.633	14.770	1.801	2.902	7.108	0.737	4.950	11.799	1.261
Drop-20	5.123	13.808	1.650	2.725	6.896	0.717	4.582	11.055	1.187
Drop-30	4.963	14.194	1.734	2.635	6.942	0.728	4.740	11.941	1.295
Drop-40	4.943	14.822	1.875	2.496	6.992	0.734	4.685	12.349	1.348

Table 6
Performance under partial random action drop (high-frequency actions only), i.e., randomly dropping [10,20,30,40] actions, across datasets.

Dataset	Beauty			Clothing			Cell		
	NDCG	HR	Precision	NDCG	HR	Precision	NDCG	HR	Precision
Drop-10	5.679	14.797	1.793	3.061	7.582	0.790	5.042	11.949	1.282
Drop-20	5.542	14.467	1.747	3.072	7.542	0.791	5.185	12.372	1.313
Drop-30	5.780	15.008	1.817	3.061	7.574	0.785	5.119	12.090	1.295
Drop-40	5.625	14.592	1.751	3.075	7.581	0.781	5.046	11.935	1.272

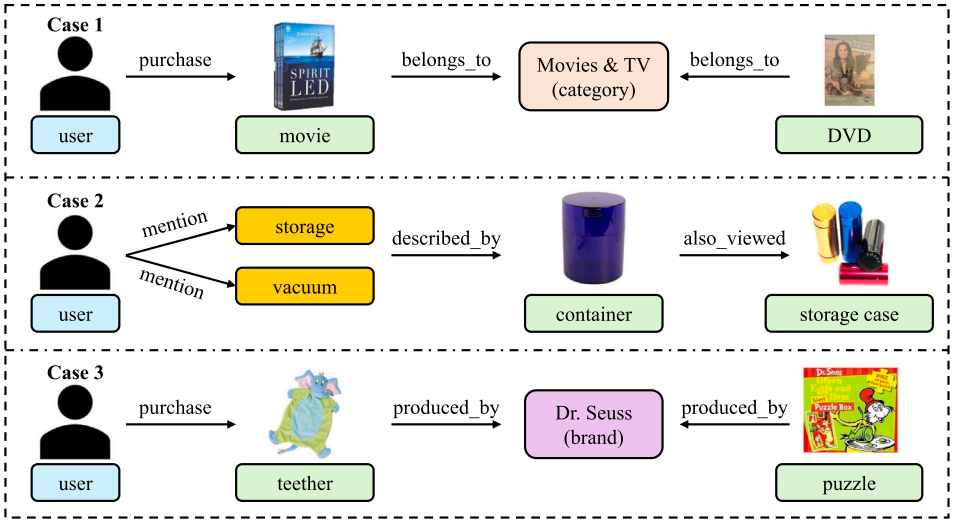


Fig. 9. Case study on generated recommendation path.

the same brand also produces a "puzzle". Based on this connection, the system recommends the "puzzle" to the user. The reasoning here follows the "produced_by" relationship, linking the user's purchase with other products from the same brand, to guess the users' preference. This recommendation path aligns with the pre-defined path pattern $\{user \xrightarrow{purchase} item \xrightarrow{produced\_by} brand \xleftarrow{produced\_by} item\}$, demonstrating how the system can leverage brand-based relationships to suggest relevant products. By tracing the connections through the "produced_by" links, the system can effectively recommend items that are likely to match the user's preferences based on their prior purchases.

While this paper has not listed all the path patterns, the case studies presented demonstrate that all recommendation paths are based on clear and pre-defined logic, ensuring explainability. This shows that the recommendation paths generated by ERLER are easy to understand, which helps improve user trust in the recommendation system.

5. Conclusion and future work

We introduce a recommendation algorithm called Evolutionary Reinforcement Learning for Explainable Recommendation (ERLER). To achieve explainable recommendations, ERLER uses path reasoning in KGs, which helps users understand how the recommendation results are generated, giving the system a clear and understandable logic. Our extensive experiments show that ERLER outperforms the baseline across all evaluation metrics on three real-world datasets. The key innovation of ERLER is incorporating evolutionary algorithms. A mutation mechanism is applied when the candidate action space is large, helping to diversify the available actions. Additionally, ERLER uses a redesigned advantage calculation in the Critic network, which accelerates the convergence of path reasoning. We also introduce a dynamic adjustment of the discount factor to better assess rewards at different time steps. The experimental results confirm that these innovations lead to the improved model performance.

In real-world scenarios, KGs are constantly updated as new entities and relationships emerge. However, RL algorithms typically require stable environments to train effectively. Any substantial change in the KGs could invalidate the learned policy, requiring retraining and re-exploration. For future work, we integrate incremental learning or online RL approaches where the model can update its policy continuously as the KGs evolve, rather than requiring full retraining. Additionally, mechanisms for detecting and handling concept drift (i.e., changes in the underlying distribution of data) can be embedded within the RL algorithm, which boosts robustness in changing conditions and enables clearer decision explanations [56].

Finally, our future work will focus on integrating Large Language Models (LLMs) into the ERLER framework for explainable recommendation. Guided by the EC-LLM roadmap [57], LLMs will act as intelligent operators to mutate and evaluate candidate recommendation policies. Following the design principle transfer [78], LLMs will seed the evolutionary population with high-quality, interpretable agent architectures. Crucially, acknowledging their causal limitations, the evolutionary RL component will retain responsibility for causal pathfinding in KGs, while LLMs will be leveraged for their generative strength in translating the final evolved reasoning paths into natural language explanations for users.

CRediT authorship contribution statement

Yuanguo Lin: Writing – original draft, Methodology, Conceptualization. **Hong Chen:** Writing – original draft, Data curation. **Xiuze Zhou:** Writing – review & editing, Validation. **Wei Zhang:** Software, Project administration, Methodology. **Huanyu You:** Investigation. **Fan Lin:** Supervision, Funding acquisition. **Pengcheng Wu:** Resources, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (No. 61977055), in part by the Natural Science Foundation of Xiamen, China (No. 3502Z202573060) and in part by the Higher Education Reform and Research Project of Fujian Higher Education Research Institute, China (No. FGJG202405).

Data availability

Data will be made available on request.

References

[1] O.S. Ajani, A. Kumar, R. Mallipeddi, Covariance matrix adaptation evolution strategy based on correlated evolution paths with application to reinforcement learning, Expert Syst. Appl. 246 (2024) 123289.
[2] A.A. Aydeniz, R. Loftin, K. Tumer, Novelty seeking multiagent evolutionary reinforcement learning, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2023, pp. 402–410.
[3] C. Bodnar, B. Day, P. Lió, Proximal distilled evolutionary reinforcement learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, 2020, pp. 3283–3290.
[4] Y. Cao, X. Wang, X. He, Z. Hu, T.-S. Chua, Unifying knowledge graph learning and recommendation: towards a better understanding of user preferences, in: The World Wide Web Conference, 2019, pp. 151–161.
[5] Y. Cao, X. Chen, L. Yao, X. Wang, W.E. Zhang, Adversarial attacks and detection on reinforcement learning-based interactive recommender systems, in: Proceedings of the 43rd International ACM SIGIR Conference, 2020, pp. 1669–1672.
[6] N.B. Channappagoudar, R. Singh, Trust based recommendation system using knowledge graph (kgtrs), in: ICIDSSD 2022: Proceedings of the 3rd International Conference on ICT for Digital, Smart, and Sustainable Development, ICIDSSD 2022, 24-25 March 2022, New Delhi, India, European Alliance for Innovation, 2023, pp. 25–36.
[7] H. Chen, X. Dai, H. Cai, W. Zhang, X. Wang, R. Tang, Y. Zhang, Y. Yu, Large-scale interactive recommendation with tree-structured policy gradient, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2019, pp. 3312–3320.
[8] J.D. Co-Reyes, Y. Miao, D. Peng, E. Real, S. Levine, Q.V. Le, H. Lee, A. Faust, Evolving reinforcement learning algorithms, arXiv preprint arXiv:2101.03958, 2021.
[9] H. Cui, T. Peng, R. Han, J. Han, L. Liu, Path-based multi-hop reasoning over knowledge graph for answering questions via adversarial reinforcement learning, Knowl.-Based Syst. 276 (2023) 110760.
[10] M. David, F. Ricci, Harnessing a generalised user behaviour model for next-poi recommendation, in: Proceedings of the 12th ACM Conference on Recommender Systems, 2018, pp. 402–406.
[11] Y. Deng, Y. Li, F. Sun, B. Ding, W. Lam, Unified conversational recommendation policy learning via graph-based reinforcement learning, in: Proceedings of the 44th International ACM SIGIR Conference, 2021, pp. 1431–1441.
[12] L. Eytan, V. Bogina, I. Ben-Gal, N. Koenigstein, Kpar: knowledge-aware path-based attentive recommender with interpretability, ACM Trans. on Recommender Syst. (2024) 1–23.
[13] H. Fan, Y. Zhong, G. Zeng, C. Ge, Improving recommender system via knowledge graph based exploring user preference, Appl. Intell. (2022) 1–13.
[14] F.C. Fernandez, W. Caarls, Parameters tuning and optimization for reinforcement learning algorithms using evolutionary computing, in: 2018 International Conference on Information Systems and Computer Science (INCISCOS), IEEE, 2018, pp. 301–305.
[15] S. Geng, Z. Fu, J. Tan, Y. Ge, G. De Melo, Y. Zhang, Path language modeling over knowledge graphsfor explainable recommendation, in: Proceedings of the ACM Web Conference 2022, 2022, pp. 946–955.
[16] C. Greco, A. Suglia, P. Basile, G. Semeraro, Converse-et-impera: exploiting deep learning and hierarchical reinforcement learning for conversational recommender systems, in: Proceedings of the 16th International Conference on Italian Association for Artificial Intelligence, 2017, pp. 372–386.
[17] Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, H. Xiong, Q. He, A survey on knowledge graph-based recommender systems, IEEE Trans. Knowl. Data Eng. 34 (8) (2020) 3549–3568.
[18] M. Hadhoud, Comparative study of neuroevolution algorithms in reinforcement learning for self-driving cars, Eur. J. Eng. Sci. Technol. (2019).
[19] A. Hur, N. Janjua, M. Ahmed, A survey on state-of-the-art techniques for knowledge graphs construction and challenges ahead, in: 2021 IEEE Fourth International Conference on Artificial Intelligence and Knowledge Engineering (AIKE), IEEE, 2021, pp. 99–103.
[20] S. Kelly, T. Voegerl, W. Banzhaf, C. Gondro, Evolving hierarchical memory-prediction machines in multi-task reinforcement learning, Genet. Program. Evolvable Mach. 22 (4) (2021) 573–605.
[21] W. Lei, G. Zhang, X. He, Y. Miao, X. Wang, L. Chen, T.-S. Chua, Interactive path reasoning on graph for conversational recommendation, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 2073–2083.
[22] Y. Lei, W. Li, Interactive recommendation with user-specific deep reinforcement learning, ACM Trans. Knowl. Discov. Data 13 (6) (Oct 2019) 61.
[23] Y. Li, L. Hou, J. Li, Preference-aware graph attention networks for cross-domain recommendations with collaborative knowledge graph, ACM Trans. Inf. Syst. 41 (3) (2023) 1–26.
[24] Y. Li, X. Sun, H. Chen, S. Zhang, Y. Yang, G. Xu, Attention is not the only choice: counterfactual reasoning for path-based explainable recommendation, IEEE Trans. Knowl. Data Eng. (2024).
[25] P. Liang, Y. Chen, Y. Sun, Y. Huang, W. Li, An information entropy-driven evolutionary algorithm based on reinforcement learning for many-objective optimization, Expert Syst. Appl. 238 (2024) 122164.
[26] Q. Lin, Y. Chen, L. Ma, W.-N. Chen, J. Li, Erl-td: evolutionary reinforcement learning enhanced with truncated variance and distillation mutation, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, 2024, pp. 13826–13836.
[27] Y. Lin, F. Lin, G. Cai, H. Chen, L. Zou, Y. Liu, P. Wu, Evolutionary reinforcement learning: a systematic review and future directions, Mathematics 13 (5) (2025) 833.

[28] Y. Lin, F. Lin, W. Zeng, J. Xiahou, L. Li, P. Wu, Y. Liu, C. Miao, Hierarchical reinforcement learning with dynamic recurrent mechanism for course recommendation, Knowl.-Based Syst. 244 (2022) 108546.
[29] Y. Lin, Y. Liu, F. Lin, L. Zou, P. Wu, W. Zeng, H. Chen, C. Miao, A survey on reinforcement learning for recommender systems, IEEE Trans. Neural Netw. Learn. Syst. (2023) 1–21.
[30] C. Liu, W. Wu, S. Wu, L. Yuan, R. Ding, F. Zhou, Q. Wu, Social-enhanced explainable recommendation with knowledge graph, IEEE Trans. Knowl. Data Eng. (2023).
[31] Y. Liu, Y. Zhang, Q. Wu, C. Miao, L. Cui, B. Zhao, Y. Zhao, L. Guan, Diversity-promoting deep reinforcement learning for interactive recommendation, arXiv preprint arXiv:1903.07826, 2019.
[32] Q. Long, Z. Zhou, A. Gupta, F. Fang, Y. Wu, X. Wang, Evolutionary population curriculum for scaling multi-agent reinforcement learning, arXiv preprint arXiv:2003.10423, 2020.
[33] S. Majumdar, S. Khadka, S. Miret, S. McAleer, K. Tumer, Evolutionary reinforcement learning for sample-efficient multiagent coordination, in: International Conference on Machine Learning, PMLR, 2020, pp. 6651–6660.
[34] L. Peng, Z. Yuan, G. Dai, M. Wang, Z. Tang, Reinforcement learning-based hybrid differential evolution for global optimization of interplanetary trajectory design, Swarm Evol. Comput. 81 (2023) 101351.
[35] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, Bpr: Bayesian personalized ranking from implicit feedback, arXiv preprint arXiv:1205.2618, 2012.
[36] A. Sehgal, H. La, S. Louis, H. Nguyen, Deep reinforcement learning using genetic algorithm for parameter optimization, in: 2019 Third IEEE International Conference on Robotic Computing (IRC), IEEE, 2019, pp. 596–601.
[37] R. Shen, Y. Zheng, J. Hao, Z. Meng, Y. Chen, C. Fan, Y. Liu, Generating behavior-diverse game AIS with evolutionary multi-objective deep reinforcement learning, in: Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence, 2021, pp. 3371–3377.
[38] O. Sigaud, Combining evolution and deep reinforcement learning for policy search: a survey, ACM Trans. Evol. Learn. 3 (3) (2023) 1–20.
[39] Y. Sun, Y. Zhang, Conversational recommender system, in: Proceedings of the 41rd International ACM SIGIR Conference, 2018, pp. 235–244.
[40] Z. Sun, J. Yang, J. Zhang, A. Bozzon, L.-K. Huang, C. Xu, Recurrent knowledge graph embedding for effective recommendation, in: Proceedings of the 12th ACM Conference on Recommender Systems, 2018, pp. 297–305.
[41] S. Tao, R. Qiu, Y. Ping, H. Ma, Multi-modal knowledge-aware reinforcement learning network for explainable recommendation, Knowl.-Based Syst. 227 (2021) 107217.
[42] G. Wan, S. Pan, C. Gong, C. Zhou, G. Haffari, Reasoning like human: hierarchical reinforcement learning for knowledge graph reasoning, in: International Joint Conference on Artificial Intelligence, International Joint Conference on Artificial Intelligence, 2021.
[43] H. Wang, F. Zhang, J. Wang, M. Zhao, W. Li, X. Xie, M. Guo, Ripplenet: propagating user preferences on the knowledge graph for recommender systems, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, 2018, pp. 417–426.
[44] H. Wang, F. Zhang, M. Zhang, J. Leskovec, M. Zhao, W. Li, Z. Wang, Knowledge-aware graph neural networks with label smoothness regularization for recommender systems, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 968–977.
[45] H. Wang, M. Zhao, X. Xie, W. Li, M. Guo, Knowledge graph convolutional networks for recommender systems, in: The World Wide Web Conference, 2019, pp. 3307–3313.
[46] P. Wang, Y. Fan, L. Xia, W.X. Zhao, S. Niu, J. Huang, Kerl: a knowledge-guided reinforcement learning model for sequential recommendation, in: Proceedings of the 43rd International ACM SIGIR Conference, 2020, pp. 209–218.
[47] Q. Wang, Y. Hao, J. Cao, Adrl: an attention-based deep reinforcement learning framework for knowledge graph reasoning, Knowl.-Based Syst. 197 (2020) 105910.
[48] Q. Wang, E. Tragos, N. Hurley, B. Smyth, A. Lawlor, R. Dong, Entity-enhanced graph convolutional network for accurate and explainable recommendation, in: Proceedings of the 30th ACM Conference on User Modeling, Adaptation and Personalization, 2022, pp. 79–88.
[49] Q. Wang, Z. Mao, B. Wang, L. Guo, Knowledge graph embedding: a survey of approaches and applications, IEEE Trans. Knowl. Data Eng. 29 (12) (2017) 2724–2743.
[50] X. Wang, D. Wang, C. Xu, X. He, Y. Cao, T.-S. Chua, Explainable reasoning over knowledge graphs for recommendation, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 5329–5336.
[51] X. Wang, Y. Chen, J. Yang, L. Wu, Z. Wu, X. Xie, A reinforcement learning framework for explainable recommendation, in: Proceedings of IEEE International Conference on Data Mining, 2018, pp. 587–596.
[52] Y. Wang, T. Zhang, Y. Chang, X. Wang, B. Liang, B. Yuan, A surrogate-assisted controller for expensive evolutionary reinforcement learning, Inf. Sci. 616 (2022) 539–557.
[53] H. Wu, H. Fang, Z. Sun, C. Geng, X. Kong, Y.-S. Ong, A generic reinforced explainable framework with knowledge graph for session-based recommendation, in: 2023 IEEE 39th International Conference on Data Engineering (ICDE), IEEE, 2023, pp. 1260–1272.
[54] X. Wu, B. Jiang, K. Yu, H. Chen, et al., Accurate markov boundary discovery for causal feature selection, IEEE Trans. Cybern. 50 (12) (2019) 4983–4996.
[55] X. Wu, B. Jiang, Y. Zhong, H. Chen, Multi-target markov boundary discovery: theory, algorithm, and application, IEEE Trans. Pattern Anal. Mach. Intell. 45 (4) (2022) 4964–4980.
[56] X. Wu, J. Wu, Y. Zhou, L. Feng, K.C. Tan, Towards robustness and explainability of automatic algorithm selection, in: Forty-Second International Conference on Machine Learning, 2025.
[57] X. Wu, S.-H. Wu, J. Wu, L. Feng, K.C. Tan, Evolutionary computation in the era of large language model: survey and roadmap, IEEE Trans. Evol. Comput. (2024).
[58] Y. Xian, Z. Fu, S. Muthukrishnan, G. de Melo, Y. Zhang, Reinforcement knowledge graph reasoning for explainable recommendation, in: Proceedings of the 42nd International ACM SIGIR Conference, 2019, pp. 285–294.
[59] Y. Xian, Z. Fu, S. Muthukrishnan, G. De Melo, Y. Zhang, Reinforcement knowledge graph reasoning for explainable recommendation, in: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2019, pp. 285–294.
[60] X. Xin, A. Karatzoglou, I. Arapakis, J.M. Jose, Self-supervised reinforcement learning for recommender systems, in: Proceedings of the 43rd International ACM SIGIR Conference, 2020, pp. 931–940.
[61] W. Xiong, T. Hoang, W.Y. Wang, Deeppath: A reinforcement learning method for knowledge graph reasoning, arXiv preprint arXiv:1707.06690, 2017.
[62] S. Yan, C. Li, H. Wang, B. Lin, Y. Yuan, Feature interactive graph neural network for kg-based recommendation, Expert Syst. Appl. 237 (2024) 121411.
[63] Y. Yang, C. Zhang, X. Song, Z. Dong, H. Zhu, W. Li, Contextualized knowledge graph embedding for explainable talent training course recommendation, ACM Trans. Inf. Syst. 42 (2) (2023) 1–27.
[64] T. Yu, Y. Shen, R. Zhang, X. Zeng, H. Jin, Vision-language recommendation via attribute augmented multimodal reinforcement learning, in: Proceedings of the 27th ACM International Conference on Multimedia, 2019, pp. 39–47.
[65] J.-C. Zhang, A.M. Zain, K.-Q. Zhou, X. Chen, R.-M. Zhang, A review of recommender systems based on knowledge graph embedding, Expert Syst. Appl. (2024) 123876.
[66] R. Zhang, T. Yu, Y. Shen, H. Jin, C. Chen, Text-based interactive recommendation via constraint-augmented reinforcement learning, in: Advances in Neural Information Processing Systems, 2019, pp. 15214–15224.
[67] W. Zhang, Y. Lin, Y. Liu, H. You, P. Wu, F. Lin, X. Zhou, Self-supervised reinforcement learning with dual-reward for knowledge-aware recommendation, Appl. Soft Comput. 131 (2022) 109745.
[68] W. Zhang, Q. Yuan, J. Han, J. Wang, Collaborative multi-level embedding learning from reviews for rating prediction, in: IJCAI, vol. 16, 2016, pp. 2986–2992.
[69] Y. Zhang, Q. Ai, X. Chen, W.B. Croft, Joint representation learning for top-n recommendation with heterogeneous information sources, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017, pp. 1449–1458.
[70] H. Zhao, Q. Yao, J. Li, Y. Song, D.L. Lee, Meta-graph based recommendation fusion over heterogeneous information networks, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017, pp. 635–644.
[71] K. Zhao, X. Wang, Y. Zhang, L. Zhao, Z. Liu, C. Xing, X. Xie, Leveraging demonstrations for reinforcement recommendation reasoning over knowledge graphs, in: Proceedings of the 43rd International ACM SIGIR Conference, 2020, pp. 239–248.
[72] W. Zhao, B. Wang, M. Yang, J. Ye, Z. Zhao, X. Chen, Y. Shen, Leveraging long and short-term information in content-aware movie recommendation via adversarial training, IEEE Trans. Cybern. 50 (11) (Nov 2020) 4680–4693.
[73] X. Zhao, L. Zhang, Z. Ding, L. Xia, J. Tang, D. Yin, Recommendations with negative feedback via pairwise deep reinforcement learning, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 1040–1048.
[74] G. Zheng, F. Zhang, Z. Zheng, Y. Xiang, N.J. Yuan, X. Xie, Z. Li, Drn: a deep reinforcement learning framework for news recommendation, in: Proceedings of the 27th International Conference on World Wide Web, 2018, pp. 167–176.
[75] H. Zheng, P. Wei, J. Jiang, G. Long, Q. Lu, C. Zhang, Cooperative heterogeneous deep reinforcement learning, Adv. Neural Inf. Process. Syst. 33 (2020) 17455–17465.
[76] L. Zheng, V. Noroozi, P.S. Yu, Joint deep modeling of users and items using reviews for recommendation, in: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, 2017, pp. 425–434.
[77] F. Zhou, R. Yin, K. Zhang, G. Trajcevski, T. Zhong, J. Wu, Adversarial point-of-interest recommendation, in: Proceedings of the 28th International Conference on World Wide Web, 2019, pp. 3462–3468.
[78] X. Zhou, X. Wu, L. Feng, Z. Lu, K.C. Tan, Design principle transfer in neural architecture search via large language models, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 39, 2025, pp. 23000–23008.
[79] Q. Zhu, H. Zhang, Q. He, Z. Dou, Query-aware explainable product search with reinforcement knowledge graph reasoning, IEEE Trans. Knowl. Data Eng. (2023).