TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 11/14 pp 1101–1114
DOI: 10.26599/TST.2022.9010063
Volume 28, Number 6, December 2023

AInvR: Adaptive Learning Rewards for Knowledge Graph Reasoning Using Agent Trajectories

Hao Zhang, Guoming Lu*, Ke Qin, and Kai Du

Abstract: Multi-hop reasoning for incomplete Knowledge Graphs (KGs) demonstrates excellent interpretability with decent performance. Reinforcement Learning (RL) based approaches formulate multi-hop reasoning as a typical sequential decision problem. An intractable shortcoming of multi-hop reasoning with RL is that sparse reward signals make performance unstable. Current mainstream methods apply heuristic reward functions to counter this challenge. However, the inaccurate rewards caused by heuristic functions guide the agent to improper inference paths and unrelated object entities. To this end, we propose a novel adaptive Inverse Reinforcement Learning (IRL) framework for multi-hop reasoning, called AInvR. (1) To counter the missing and spurious paths, we replace the heuristic rule rewards with an adaptive rule reward learning mechanism based on agent's inference trajectories; (2) to alleviate the impact of over-rewarded object entities misled by inaccurate reward shaping and rules, we propose an adaptive negative hit reward learning mechanism based on agent's sampling strategy; (3) to further explore diverse paths and mitigate the influence of missing facts, we design a reward dropout mechanism to randomly mask and perturb reward parameters for the reward learning process. Experimental results on several benchmark knowledge graphs demonstrate that our method is more effective than existing multi-hop approaches.

Key words: Knowledge Graph Reasoning (KGR); Inverse Reinforcement Learning (IRL); multi-hop reasoning

1 Introduction

Knowledge Graph (KGs), such as Freebase[1], YAGO[2], and DBpedia[3], are widely used in natural language processing[4, 5], question-answering[6, 7], recommender systems[8, 9], and other Artificial Intelligence (AI) applications[10–14]. However, incompleteness and uncertainty of KGs lead to performance degradation on upstream assignments, which makes it a popular research direction to predict missing facts through Knowledge Graph Reasoning (KGR)[15, 16]. The mainstream reasoning approaches are divided into neural, symbolic, and neural-symbolic methods[17].

Neural reasoning, also named Knowledge Graph Embedding (KGE), focuses on learning low-dimensional representation of entities and relations[18–22]. Despite the outstanding performance of neural reasoning, it is difficult to identify logic rules in the reasoning process, resulting in the lack of interpretability.

Symbolic reasoning methods, also known as rule-based reasoning, deduce logic rules from KGs and adopt these rules to infer missing facts[23–27]. Figure 1 illustrates an incomplete subgraph that misses fact *Led sports* (*James*, *Cavaliers*), which can be inferred from a logic rule: *Led sports* $\leftarrow$ *Plays in* $\wedge$ *Part of*. Although symbolic methods make a friendly interpretation, they are limited by strict matching and discrete logic operations, in which noise and ambiguity are insupportable.

Neural-symbolic reasoning combines symbolism and connectionism to solve reasoning problems[17]. Popular

• Hao Zhang, Guoming Lu, Ke Qin, and Kai Du are with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China. E-mail: zren@std.uestc.edu.cn; lugm@uestc.edu.cn; qinke@uestc.edu.cn; 202021080517@std.uestc.edu.cn.
* To whom correspondence should be addressed.
Manuscript received: 2022-07-17; revised: 2022-12-02; accepted: 2022-12-07

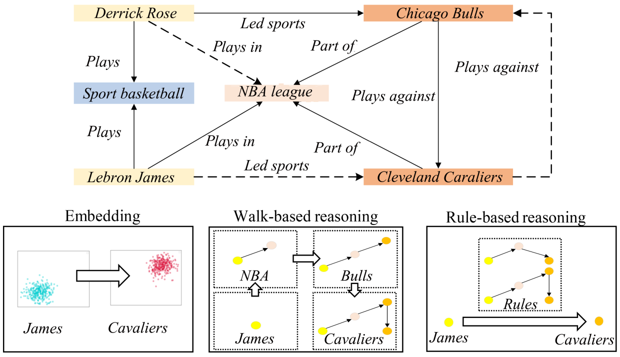


Fig. 1 Example of three king of reasoning methods in an incomplete subgraph of NELL-995[28]. Solid lines are existing facts and dotted lines are facts that require reasoning.

approaches of neural-symbolic reasoning aim at training an agent to infer path and object entity over KGs, which called walk-based reasoning. Recently, numerous walk-based reasoning methods based on RL approaches[28–35] solve KGR as a sequential decision process of reasoning. Both effectiveness and interpretability are demonstrated in these multi-hop methods.

However, RL approaches for KGR suffer from sparse rewards, leading to unstable performance. Moreover, the RL agent is misled by spurious search trajectories, resulting in false-positive inference paths. Recent efforts[30, 32–35] address the challenges via restricting the walking action or introducing heuristic reward functions, most of which depend on pre-trained embeddings and rules. These methods prove effectiveness in solving sparse reward signals, but they also introduce inaccurate and misleading rewards. First, pre-extracted rules result in receiving inaccurate rewards. Especially, in the process of reasoning, incorrect rules mislead the inference paths, and undefined rules fail to supply rewards. In this case, the agent tends to infer paths with rule guide rather than correct ones. Second, heuristic reward strategies for missing facts provide unauthentic rewards for prediction. For example, reward shaping[30] depends entirely on the quality of embeddings, and general rules[34, 35] cannot resolve noise. These rewards lead to the false prediction of the object entity.

This paper proposes an adaptive IRL reasoning framework, named AInvR, to solve the aforementioned problems. First, instead of pre-extracted symbolic rules, we develop an adaptively updated Rule Base (RB) and Candidate Rule Base (CRB) to provide an adaptive rule reward mechanism. The undefined rules are updated to the RB and CRB, and the existing rules will receive feedback if rules guide the agent. Second, we present an Inverse Knowledge Base (IKB) to provide additional negative rewards to balance over-rewards. Non-existing object entities that suffer from unauthentic rewards are more likely to have high prediction probability, these entities are identified as the main target for negative rewards design. Furthermore, to explore diverse paths and mitigate the impact of missing facts, we design a reward dropout mechanism to randomly mask and perturb reward parameters at rewards learning process.

Experimental results demonstrate that our approach achieves better performance on five well-established KGR benchmarks (FB15K-237, WN18RR, CoDEx-S, CoDEx-M, and CoDEx-L). Additionally, our method outperforms traditional rule-based models significantly on large relation-focused datasets.

2 Related Work

2.1 Knowledge graph embedding

Single-hop KGR approaches aim at learning distributed embedding of entities and relations[36, 37]. Translation-based models[18, 20, 38] interpret KGR as a translation of subject entity to object entity via given relation. Multiplicative models[19, 21, 39] measure relation and entity as tensor products. Deep learning models[22, 40, 41] embed entities and relations by various neural networks. Despite their excellence, embedding is uninterpretable because of the absence of pathfinding. Also it is limited to single-hop reasoning scenarios.

2.2 Multi-hop reasoning

To improve interpretability, a large number of multi-hop methods have been proposed. Rule-based approaches[23, 26, 42] aim at generating logic rules from KGs for multi-hop reasoning. Walk-based efforts focus on inferring the paths between subject entity and object entity by learning a pathfinding agent. PRA[43] performs a random-walk based algorithm to aggregate discrete paths with linear regression. DeepPath[28] and MINERVA[29] model the interaction of KG and agent as a Markov Decision Process (MDP)[44], with a 0/1 hit reward indicating whether the prediction is an correct object entity or not. Multi-Hop[30] proposes a soft reward shaping mechanism, based on pre-trained embedding, to avoid sparse reward issue. It also introduces an action dropout to enforce effective path exploration. DacKGR[33] utilizes embedding to provide an additional action space to explore potential inferences. Those approaches are associated with the object entity, but ignore the authenticity of path. RuleGuider[34] and RARL[35] introduce RL frameworks that make use of pre-trained symbolic rules to fight against spurious path problems.

Deep RL based approaches have achieved remarkable progress in KGR[30, 32–35]. However, compare to other AI domains, RL-based approaches in KGR often suffer from a larger discrete action space, resulting in sparse reward signals[28, 45, 46]. Heuristic reward functions[30, 32–34] address this issue to some extent, but they also lead to inaccurate rewards, thus degrading the agent's reasoning ability. In addition, it is difficult for RL-based methods to take advantage of supervised pre-trained models because reasoning often misses correct paths in complex scenario[47].

3 Proposed Method: AInvR

3.1 Problem formulation

The KG is formalized as $G = (E, R, U)$, where E is the entities set, R is the relations set, and U is the set of the facts in KG. A fact $u \in U$ can be represented by a triplet (e_s, r_q, e_o) containing subject entity $e_s \in E$, relation $r_q \in R$, and object entity $e_o \in E$.

Given a query $(e_s, r_q, ?)$, the definition of KGR task is to predict a possible object entity e_o through a k-hop entities-relations path $e_s \xrightarrow{r_1} e_1 \xrightarrow{r_2} e_2 \cdots \xrightarrow{r_k} e_o$. An example is shown in Fig. 1, the possible path for query (*Lebron James*, *Led sports*, ?) is *Lebron* $\xrightarrow{\textit{Plays in}}$ *NBA* $\xrightarrow{\textit{Part of}_{inv}}$ *Cleveland Cavaliers* that concludes the object entity *Cleveland Cavaliers* where $\textit{of}_{inv}$ is the inverse relalion.

3.2 RL framework for reasoning

The interactive process of RL is modeled as a MDP, which is represented by a tuple (S, A, P, R_g), where S is the state space, A is the set of all actions, $P(s_{t+1}|s_t, a_t)$ is the state transition probabilities, and $R_g(s, a)$ refers to the reward function.

(1) State

In the modeling structure of MDP for KGR, the state $s_t \in S$ at step t is defined as a tuple $(e_t, r_t, (e_s, r_q), (h_t^E, h_t^R))$, where e_t is the current entity, r_t is the current relation, (e_s, r_q) is the initial query, and the encoding (h_t^E, h_t^R) represent the historical path of entities and relations, respectively.

(2) Action

The action space $A_t \subseteq A$ refers to the action space at step t. We formulate $A_t = \{(e_{t+1}, r_{t+1})| (e_t, r_{t+1}, e_{t+1}) \in G)\}$ indicating the set of outgoing edges of e_t. In addition, each A_t includes a terminate edge so that the agent can stop walking at the current step.

(3) Transition

The transition function $S \times A \rightarrow S$ defines the probability distribution for the next states: $P(s_{t+1}|s_t, a_t)$. In the current state s_t, the agent reaches the next state s_{t+1} according to the current action a_t.

(4) Reward

In our framework, the reward function is divided into two parts: rule reward R_r and hit reward R_h, the global reward function is shown as below:

$$R_g = R_r + R_h \tag{1}$$

The rule reward R_r represents feedback on the agent's inference path and the hit reward R_h indicates the validity of the reasoning object.

3.3 Architecture for AInvR

The architecture of AInvR includes two parts: the reasoning process and the rewards learning process. The overall architecture is illustrated in Fig. 2.

(1) Policy network

The policy network consists of two interactive sub-policy networks parameterized by fully connected layers: relation policy network $\pi(r_{t+1}|s_t)$ and entity policy network $\pi(e_{t+1}|s_t)$.

The relation policy network infers the next hop relation probability distribution $\boldsymbol{p}_t^R$ based on the information of the current state,

$$\boldsymbol{p}_t^R \sim \pi(r_{t+1}|s_t) = \sigma(\boldsymbol{A}_t^R \times W_1^R \times ReLU(W_2^R[r_q; r_t; h_t^R])) \tag{2}$$

where $\boldsymbol{A}_t^R$ is the stacking of relation action space, W_1^R and W_2^R are trainable parameters, $\sigma(\cdot)$ is the softmax function, r_q is the initial query relation, and r_t is the relation of the current state. The vector h_t^R represents the LSTM-encoded[48] historical information of the relation from the beginning to step t,

$$h_t^R = LSTM(h_{t-1}^R, r_t) \tag{3}$$

$$h_0^R = LSTM(0, r_q) \tag{4}$$

The entity policy network outputs the next entity probability distribution $\boldsymbol{p}_t^E$ after the calculation of the following formula:

$$\boldsymbol{p}_t^E \sim \pi(e_{t+1}|s_t) = \sigma(\boldsymbol{A}_t^E \times W_1^E \times ReLU(W_2^E[e_s; r_q; e_t; h_t^E])) \tag{5}$$

where $\boldsymbol{A}_t^E$ is the stacking of entity action space, W_1^E and W_2^E are trainable parameters. h_t^E is encoded by another

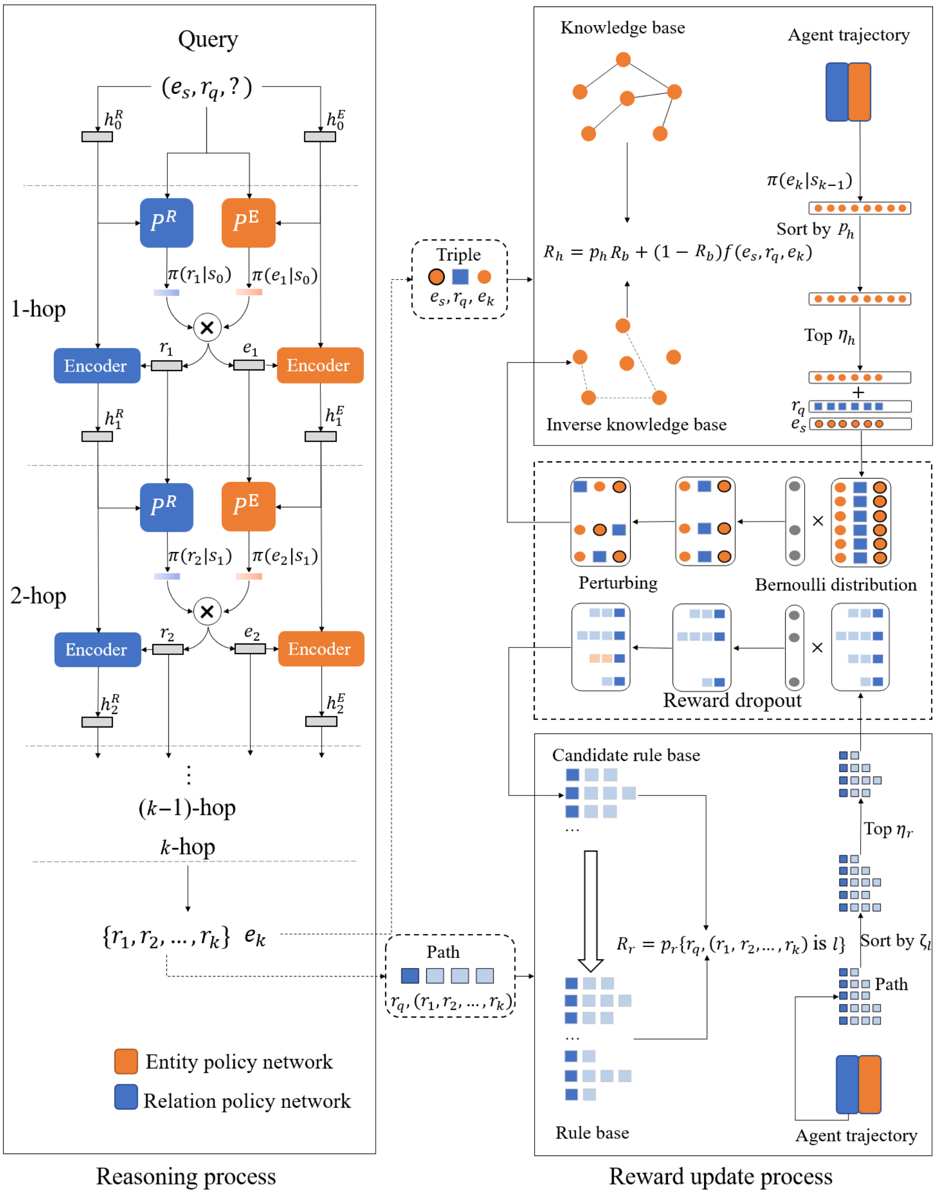


Fig. 2 Overall architecture. In the reasoning process, the entity policy network infers the next entity probability, and the relation policy network infers the next relation probability. Then, the distribution of the next action is calculated by the multiplication of two probabilities, and the historical information is encoded after selecting the action. In the rewards learning process, RB and CRB are updated based on the agent trajectories, rule rewards in RB are updated, and undefined rules are added into the CRB. The negative facts sampled by the agent are selectively updated to the IKB. At each update step, the reward dropout mechanism randomly masks and perturbs the undefined rules and negative facts.

LSTM,

$$h_t^E = LSTM(h_{t-1}^E, e_t) \tag{6}$$

$$h_0^E = LSTM(0, e_s) \tag{7}$$

At each step t, the agent policy $\pi(a_t, s_t)$ is composed of relation policy $\pi(r_{t+1}|s_t)$ and entity policy $\pi(e_{t+1}|s_t)$. The action (e_{t+1}, r_{t+1}) is sampled by the probability distribution $\boldsymbol{p}_t^R \times \boldsymbol{p}_t^E$ until the k-hop inference is completed. Then, the agent output the predicted inference rule $(r_q, (r_1, r_2, \ldots, r_k))$ and object entity e_k. As described above, we design separate reward functions for rules and hits.

(2) Rule base and candidate rule base

We introduce an adaptive RB to store rules and generate rule rewards. We also present a CRB to make use of undefined rules at each iteration. Note that, the rules stored in RB and CRB are different, and there is no intersection between them. We formulate the query relation r_q and relations path $(r_1, r_2, \ldots, r_k)$ as a rule l, which is represented as a quaternion form,

$$l = (r_q, (r_1, r_2, \ldots, r_k), q_l, c_l) \tag{8}$$

where q_l is the times that the agent hits the correct object entity following rule l, and c_l is the total times that the agent traverses rule l. The rule reward R_r provided by RB and CRB is defined as

$$R_r = \lambda_r p_r\{(r_q, (r_1, r_2, \dots, r_k)) \text{ is } l\},$$
$$p_r = \frac{q_l}{c_l} \tag{9}$$

where λ_r is a discount factor. When the agent walks through rule $(r_q, (r_1, r_2, \dots, r_k))$, it receives rule confidence p_r as rule reward.

(3) Knowledge base and inverse knowledge base

We propose the KB and the IKB in the form of KGs with uncertainty[49]. KB is used to store the existing facts in KG, and IKB is designed to store negative facts that are over-rewarded. Each of their edges is represented as a quaternion (e_s, r_q, e_o, p_h) including subject entity e_s, relation r_q, object entity e_o, and prediction confidence p_h. Given a predicted entity e_k, the hit reward R_h is calculated as

$$R_b = \mathbf{1}\{e_k = e_o\} \tag{10}$$

$$R_h = \lambda_h(p_h R_b + (1 - R_b) f(e_s, r_q, e_k)) \tag{11}$$

where λ_h is a discount factor, R_b is a binary reward, which will be set to 1 if the object entity is correctly inferred. $f(e_s, r_q, e_o)$ is a composition function for reward shaping with embeddings[30]. If the reasoning hits the fact in KB or IKB, the hit reward is set to the fact confidence p_h, otherwise, the output probability $f(e_s, r_q, e_o)$ is used as a substitute for the hit reward.

3.4 Rule update algorithm

Walk-based approaches taking advantage of rules[34, 35] show superiority in solving unauthentic paths and the sparse reward problem. However, the performance of these methods relies on the quality of pre-trained symbolic models heavily. In particular, inadequately pre-extracted rules mislead the agent's walking, resulting in erroneous reasoning. To this end, we design a rule update algorithm to adjust the rule rewards and update undefined rules based on the agent's search paths. Specifically, for each rule exploited, the confidence will be adjusted according to the result of path searching. The undefined rules, which are derived from unknown valid paths, are added to the CRB as temporary rules. The detail of the process is shown in Algorithm 1, where γ_r and η_r are hyper-parameters that constrain the capacity of RB, α_r is a hyper-parameter for reward dropout, and L' is the set of rules after paths perturbing. The impact factor ζ_l is calculated by the sum of exploration and exploitation.

Algorithm 1 Rule update algorithm

Require: $\Lambda = \{\Lambda_1, \Lambda_2, \dots\}, \Lambda_i = \{(r_q^i, (r_1^i, r_2^i, \dots, r_k^i), R_r^i, R_h^i)\}$
 for $i = 1$ to $\|\Lambda\|$ **do**
  if $(r_q^i, (r_1^i, r_2^i, \dots, r_k^i))$ is $l \in$ RB or CRB **then**
   $q_l = q_l + R_h^i - R_r^i\{c_l < \gamma_r\}$;
   $c_l = c_l + \mathbf{1}\{c_l < \gamma_r\}$;
  else if $R_h^i = 1$ **then**
   Define $l = (r_q^i, (r_1^i, r_2^i, \dots, r_k^i), q_l, c_l), q_l = 1, c_l = 1$;
   Update rule l to CRB;
  end if
 end for
 Descending sort rules in CRB by impact factor ζ_l;
 Extract the top η_r rules $L = \{l_1, l_2, \dots\}$ in CRB;
 Initialize $m \sim Bernoulli(1 - \alpha_r)$;
 Randomly mask rules L using reward dropout;
 Obtain masked rules $L' = \{l'_1, l'_2, \dots\}$;
 Randomly perturb paths $(r_1, r_2, \dots, r_k)$ from L';
 Add rules $L + L'$ to RB;
 Clear CRB.

$$\zeta_l = \frac{q_l}{c_l} + \frac{c_l}{\max\limits_{u_i \in CRB}(u_i)} \tag{12}$$

Exploration is calculated by the correct rate of rule l, which is shown as the first term of Eq. (12). The second term formulates the exploitation as the normalized frequency of the rule queried.

3.5 Knowledge update algorithm

The potential problem of heuristic rewards is that the global reward function may mislead reasoning. Specifically, the robustness of rules and quality of embeddings are dubious for walk-based methods. For example, a high reward rule *personNationality*$(X, Y)\leftarrow$*birthPlace*(X, Y) does not apply to persons who change nationality. Since the agent is more inclined to predict paths and entities with a higher global reward, as a result, heuristic rewards provide incorrect fact prediction for *personNationality*. Generally, non-existing object entities with high rewards lead to false predictions. To solve this problem, the knowledge update algorithm provides adaptive negative hit rewards for unauthentic entity predictions to balance excessive rewards. We formulate these over-rewarded object entities and initial queries as negative These negative facts are considered as the main target for reward correction and are handled in the IKB. The pseudo-code is shown in Algorithm 2, where η_h and γ_h are hyper-parameters that constrain the bandwidth and algorithm

Algorithm 2 Knowledge update algorithm

Require: $\Lambda = \{\Lambda_1, \Lambda_2, \dots\}, \Lambda_i = \{(r_q^i, (r_1^i, r_2^i, \dots, r_k^i), R_r^i, R_h^i)\}$,
  if $epoch/\gamma_h = 0$ **then**
    Clear IKB;
    for $i = 1$ to $\|\Lambda\|$ **do**
      Set $B = \{(e_k^1, p_k^1), (e_k^2, p_k^2), \dots\} \leftarrow Agent(e_s, r_q, ?)$;
      Ascending sort B by confidence p_h^i;
      Retain top η_h tuple in B;
      Initialize $m \sim Bernoulli(1-\alpha_h)$;
      Randomly mask B using reward dropout;
      for e_s, r_q, e_k^i, p_h^i in $\|B\|$ **do**
        Define negative fact $u = (e_s, r_q, e_k^i, -p_h^i)$;
        Randomly perturb e_s, r_q, and e_o from u;
        if u is not in KB **then**
          Add fact u to IKB;
        end if
      end for
    end for
  end if

interval, respectively, so as to reduce the running time of the knowledge update algorithm. α_h is a hyper-parameter for reward dropout.

3.6 Reward dropout

Undefined rules derived from agent's experience are inadequate to guide agent to explore diverse paths completely. On the other hand, the negative hit rewards of missing facts weaken the prediction ability to a certain extent. To further explore diverse paths and reduce the impact of missing facts, we propose a reward dropout mechanism, which randomly masks and perturbs the element of rules and facts in the reward learning process. The reward dropout used in both algorithm is similar except for the perturbation part.

In the rule update algorithm, reward dropout randomly masks rules based on the Bernoulli distribution with parameter α_r. The paths of masked rules are randomly perturbed. Then, the rules after path perturbing are updated to CRB along with other unmasked rules.

$$\tilde{L} = L \times m + L' \times (1-m),$$
$$m \sim Bernoulli(1-\alpha_r) \tag{13}$$

The path $(r_1', r_2', \dots, r_k')$ of rule l' is randomly rearranged by its initial path $(r_1, r_2, \dots, r_k)$ of rule l.

In the knowledge update algorithm, facts in the prediction set are randomly selected based on the Bernoulli distribution with parameter α_h. The remaining selected facts $\tilde{B}$ are updated to IKB after randomly perturbing the facts,

$$\tilde{B} = B' \times m,$$
$$m \sim Bernoulli\,(1-\alpha_h) \tag{14}$$

where $B' = \{u_1', u_2', \dots\}$ refer to the set of negative facts. The triplet (e_s', r_q', e_k') of facts u' is formulated by the predicted fact (e_s, r_q, e_k) after the random perturbation of the subject, relation, and object position.

3.7 Optimization and training

The optimization process of policy networks aims to maximize the global reward over all queries sampled in KB and IKB,

$$J(\theta) = E_{(e_s, r_q, e_o, p_h) \in \text{KB} \cup \text{IKB}}[E_{a \sim \pi_\theta}[R(s_k|e_s, r_q)]] \tag{15}$$

where a represents the action of π_θ.

We adopt Guided Cost Learning (GCL)[50] to optimize the policy network and reward function alternately. The REINFORCE[51] algorithm is used as a policy gradient optimization by utilizing the reward function, while the reward update algorithm is called using the agent's trajectories. The training procedure is shown in Algorithm 3, where θ denotes the network parameters. To improve the performance of the agent, we use KGE[19, 40] to pre-train the representation of entities and relations. Additionally, other techniques proposed in previous walk-based reasoning methods are adopted in the training procedure, including beam search, action dropout, etc. Implementation details are available in Appendix A.

Algorithm 3 Training procedure

  Initialize θ, KB, RB
  for $epoch = 1$ to $episode$ **do**
    Sample queries $(e_s, r_q, e_o, p_h) \in$ (KB $\cup$ IKB);
    for $j = 1$ to $\|\text{queries}\|$ **do**
      Initialize state s_0 using e_s^j and r_q^j;
      for $t = 0$ to $k-1$ **do**
        Sample action $a \sim \boldsymbol{p}_t^R \times \boldsymbol{p}_t^E$;
        Calculate h_t^R and h_t^H;
        Observe state s_{t+1};
        if a is terminate edge **then**
          break;
        end if
      end for
      Obtain set $\Lambda_j = \{(r_1, r_2, \dots, r_k), e_k, p_h\}$;
      Calculate $R = R_r + R_h$;
      Update θ using
        $\nabla_\theta J(\theta) = \nabla_\theta \sum_{t=1}^{k} R(s_k|e_s, r_q) \log \pi_\theta(a_t|s_t)$;
      Update R_r using Algorithm 1 with set Λ_j;
    end for
    Obtain set $\Lambda = \{\Lambda_1, \Lambda_2, \dots, \Lambda_{\|\text{queries}\|}\}$;
    Update R_h using Algorithm 2 with Λ;
  end for
  Output: $Agent_\theta$

4 Experiment

4.1 Experiment setup

(1) Dataset

We evaluate our proposed AInvR on five appropriately difficult benchmark datasets: FB15k-237[52], WN18RR[40], CoDEx-S, CoDEx-M, and CoDEx-L[53]. Table 1 shows the dataset statistics.

(2) Baseline

We compare our experiments with other reproducible baselines for KGR, both the embedding models ComplEx[19], ConvE[40], QuatE[54], and the multi-hop approaches Neural-LP[42], AnyBURL[26], RNNLogic[55], MINERVA[29], Multi-hop[30], RuleGuider[34], and RARL[35].

(3) Experimental setup

During evaluation, for each test edge (e_s, r_q, e_o), we mask the tail entity and compute the query $(e_s, r_q, ?)$ through AInvR search by beam search. Two popular evaluation metrics are used to evaluate the ranks from prediction, including Hit@N with cut-off values $N = 1$ and 10, and Mean Reciprocal Rank (MRR)[18].

Hit@1 corresponds to the correct rate of the object entity with the highest prediction probability. Hit@10 is the accuracy of the top ten predictions. MRR is the average reciprocal of the first ranking of correct entities.

Table 1 Datasets statistics used in the experiments.

Dataset	Number of Entities	Number of Relations	Number of Triples		
			Train	Valid	Test
FB15k-237	14 541	237	272 115	17 535	20 466
WN18RR	40 943	11	86 835	3034	3034
CoDEx-S	2034	42	32 888	1827	1828
CoDEx-M	17 050	51	185 584	10 310	10 311
CoDEx-L	77 951	69	551 193	30 622	30 622

(4) Hyperparameter

We define the dimension of entity, relation, and historical encoding to 200. During the training steps of the reasoning process, the agent makes a maximum of 3-hops per reasoning. In the reward learning process, we set $\gamma_r = 1000$, $\eta_r = 10\%$, and η_h is in the interval $[0, 300\,000]$. See Appendix A for detailed hyperparameters settings.

4.2 Result

Tables 2 and 3 list the experimental results on multiple datasets. We observe that the AInvR obtains better performance on most metrics across the five benchmarks.

Table 2 Experimental results on FB15k-237 and WN18RR. The best score of multi-hop approaches are bolded and the best score of embeddings are underlined.

(%)

Method	FB15k-237			WN18RR		
	Hit@1	Hit@10	MRR	Hit@1	Hit@10	MRR
Neural-LP	16.6	34.8	22.7	37.6	**65.7**	46.3
AnyBURL	26.9	52.0	–	42.9	53.7	–
RNNLogic (w/o emb.)	17.5	36.3	23.8	41.4	51.9	45.1
MINERVA	21.7	45.6	29.3	41.3	51.3	44.8
Multi-Hop (ComplEx)	**32.9**	54.4	39.3	42.5	52.6	46.1
Multi-Hop (ConvE)	32.7	56.4	40.7	41.8	51.7	45.0
RuleGuider (ComplEx)	31.3	56.4	39.5	44.3	55.5	48.0
RuleGuider (ConvE)	31.6	57.4	**40.8**	42.2	53.6	46.0
RARL	–	–	–	44.2	53.3	46.9
AInvR (ComplEx)	30.2	56.8	39.2	**45.8**	56.9	**49.6**
AInvR (ConvE)	32.1	**58.4**	40.5	44.0	53.1	45.8
DistMult	32.4	60.0	41.7	35.7	38.4	36.7
ComplEx	32.8	61.6	42.5	41.8	48.0	43.7
ConvE	<u>34.1</u>	62.2	<u>43.5</u>	40.3	54.0	44.9
RotateE	32.2	61.6	42.2	42.2	54.1	46.4
QuatE	33.1	<u>62.5</u>	43.0	<u>45.2</u>	<u>58.2</u>	<u>49.9</u>

Table 3 Experimental results on CoDEx-S, CoDEx-M, and CoDEx-L. The best scores of multi-hop approaches are bolded and the best scores of embeddings are underlined.

(%)

Method	CoDEx-S			CoDEx-M			CoDEx-L		
	Hit@1	Hit@10	MRR	Hit@1	Hit@10	MRR	Hit@1	Hit@10	MRR
AnyBURL	33.6	61.6	–	22.9	42.4	–	23.0	42.4	–
RNNLogic(w/o emb.)	19.7	50.4	30.2	15.1	32.8	21.1	–	–	–
Multi-Hop(ComplEx)	37.4	75.2	50.1	34.1	66.9	43.7	**37.6**	65.7	47.0
RuleGuider(ComplEx)	35.6	78.1	49.6	32.2	64.8	42.5	33.2	59.4	42.1
RuleGuider(ConvE)	38.7	79.6	52.5	34.3	69.2	46.2	35.1	62.2	46.0
AInvR(ComplEx)	38.9	79.5	52.1	34.5	68.7	46.0	35.6	66.2	45.9
AInvR(ConvE)	**40.0**	**80.5**	**54.0**	**36.1**	**70.8**	**47.8**	36.7	**68.0**	**47.3**
DistMult	47.2	82.6	59.6	40.7	71.4	51.3	39.4	69.7	49.8
ComplEx	46.3	67.0	59.4	43.4	75.1	54.4	41.8	70.1	51.6
ConvE	<u>48.6</u>	85.8	<u>61.4</u>	<u>43.7</u>	<u>75.4</u>	<u>54.7</u>	<u>43.9</u>	<u>72.3</u>	<u>53.7</u>
QuatE	42.4	<u>86.1</u>	56.8	40.6	<u>75.4</u>	52.3	38.1	69.4	48.7

On FB15k-237, our model achieves the best result in Hit@10 and the suboptimal result in MRR. On WN18RR, our model performs best on Hit@1, Hit@10, and MRR among walk-based models, but Hit@10 is lower than the rule-based approach. On CoDEx-S and CoDEx-M, our model outperforms other baselines in terms of Hit@1, Hit@10, and MRR. On CoDEx-L, our model achieves the best result in terms of Hit@10 and MRR. Moreover, to overall compare the AInvR with other state-of-the-art models in the same experimental conditions, we add the reverse edge in KGs to train and test the agent. The experimental detail and results are shown in Appendix B.

During the tuning process of discount factor λ_h and λ_r, we find that the agent is more dependent on the rule guidance on datasets with fewer relation types (WN18RR and CoDEx). Conversely, although rules are still helpful, the over introduction of rule rewards on dataset with complex relation types (FB15k-237) lead to performance degradation.

The highest prediction probability (Hit@1) of our model performs suboptimally on large KGs (FB15k-237 and CoDEx-L). One potential reason is that there are more missing facts in large KGs. Part of missing facts are extracted into IKB along with negative facts, which affect the effectiveness of IKB.

Walk-based methods show advantages when the relation hierarchy is obvious. The experimental results of comprehensive benchmarks (CoDEx-S, CoDEx-M, and CoDEx-L) support this argument. In contrast to FB15k-237, CoDEx is more suited to evaluate the effectiveness of models learning non-frequency relation patterns[53]. It can be found that walk-based models show astonishingly excellent performance on CoDEx compared to rule-based methods, especially in ultra-large-scale knowledge graph (CoDEx-L). The conclusion is that the walk-based models are more sensitive to non-frequency relations, particularly when the amount of entities is huge, walk-based models show more powerful reasoning ability than others. We consider that it is because the actions inferred by the agent are limited by the action space in the current state, which is mostly miniature, resulting in simpler reasoning.

Note that, 25 percents of the facts in the WN18RR test set cannot be reached within 3-hops, even if the reverse relations are added to KG. Conversely, In FB15k-237 and CoDEx, less than 0.3% of the facts are inaccessible. We also prove that the difference between KGE and walk-based methods shown on WN18RR in Appendix C.

4.3 Ablation analysis

Separately, we respectively train a CRB-only agent and an IKB-only agent to verify the effectiveness of the two adaptive components. Their MRR scores on FB15k-237, WN18RR, CoDEx-S, and CoDEx-M are shown in Table 4.

We observe that freeze IKB or CRB performs worse than the original model, this result supports the validity of the two components from our model. In general, except for WN18RR, deleting IKB cause more serious performance degradation. In other words, the rule rewards in WN18RR provide a more effective boost than other benchmarks. This is because the reasoning of the agent in WN18RR is more dependent on rules, as we mentioned earlier.

4.4 Rewards comparision

To study the impact of the rule update algorithm proposed by our adaptive approach, we compare the rule rewards between RuleGuider[34] and AInvR. AnyBURL[26] is an efficient rules extraction model, which is the rule reward function used in RuleGuider. Table 5 shows the comparison of the rules from other pre-trained methods or our model. Table 6 shows a difference of rules from the heuristic approach and AInvR. Obviously, pre-training does not show enough ability to promote path searching. The rule reward learning mechanism expands multifarious paths for walk-based reasoning, which further enhance the reasoning performance. The average confidence of AnyBURL's rules decreases with the number of extractions because many low-scoring rules are mixed in. Obviously, AInvR provides higher quality rules. The reason is that we only

Table 4 Ablation study results (MRR) on FB15k-237, WN18RR, CoDEx-S, and CoDEx-M. Best scores are bolded in each category.

(%)

Component	FB15k-237	WN18RR	CoDEx-S	CoDEx-M
AInvR	**39.2**	**49.6**	**52.1**	**46.0**
Only IKB	39.0	48.5	50.5	45.8
Only CRB	38.8	48.7	49.7	43.7

Table 5 Comparison of rule number and average confidence extracted from AInvR and rule-based reasoning methods.

Approach	Rule number		Average confidence	
	FB15k-237	WN18RR	FB15k-237	WN18RR
AnyBURL(10 ms)	4659	218	0.229	0.086
AnyBURL(100 ms)	19 166	424	0.189	0.082
AnyBURL(1000 ms)	51 873	646	0.164	0.069
AInvR	**96 833**	**1664**	**0.517**	**0.265**

Table 6 Example of rules extracted from FB15k-237 using AnyBURL and AInvR, bolded are rules that are only extracted by AInvR.

Approach	Rule	Relation	Confidence
AnyBURL	*LegislativeSessions→ LegislativeSessions→ DistrictRepresented*	*District represented*	0.560
	Religion		0.224
	DistrictRepresented		0.116
	Crewmember	*FilmReleaseRegion*	0.233
	FilmReleaseDistributionMedium		0.170
	FilmsDistributed → Film → FilmReleaseRegion		0.223
AInvR	*LegislativeSessions→ LegislativeSessions→ DistrictRepresented*	*District represented*	0.153
	Religion		0.178
	DistrictRepresented		0.878
	DistrictRepresented→ Country → Contains		0.104
	DistrictRepresented→ FirstLevelDivisionOf→ Country$_{inv}$		0.142
	Crewmember	*FilmReleaseRegion*	0.672
	FilmReleaseDistributionMedium		0.648
	FilmsDistributed → Film → FilmReleaseRegion		0.184
	Titles → CountryOfOrigin		0.761
	DubbingPerformances→ Actor → Nationality		0.750
	NominatedFor→ Country		0.434

keep rules with the highest exploration and exploitation during RB update, which guarantees the validity of the rules.

Besides, we consider the hit reward learning mechanism to be similar to the principle of negative sampling. To understand the contribution of the hit reward learning mechanism which is different from negative sampling, we make the corresponding experiment by using negative samples with hit reward $p_h = -1$ to guide the agent. We compare three types of negative sampling strategies[56]: NegSamp, 1vsAll, and KvsAll, and report the convergence curves of the policy network of AInvR in Figs. 3 and 4. For WN18RR, only KvsAll works as well as our strategy. But in terms of stability, Kvsall is still not as good as our strategy, as it shows a performance decline after several rounds of training. For CoDEx-S, other strategies are slightly less effective than our mechanics. In general, the negative sampling strategies may lead to sparse reward signals or excessive negative feedback to degrade reasoning performance. As our method suggests, designing negative rewards with the policy sampling and reward dropout can avoid the problems effectively.

5 Conclusion

In this paper, we propose an adaptive inverse reinforcement learning framework AInvR to address the issue of sparse and inaccurate rewards in walk-based reasoning. Our approach learns policy and rewards alternately via two reward learning mechanisms. Specifically, in the rule reward learning process, the rule rewards of both existing and unknown rules are learned

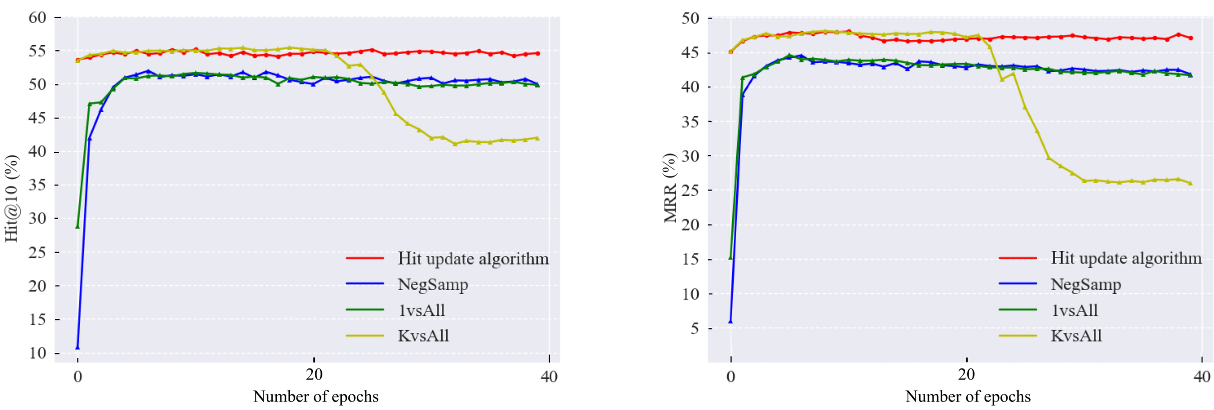
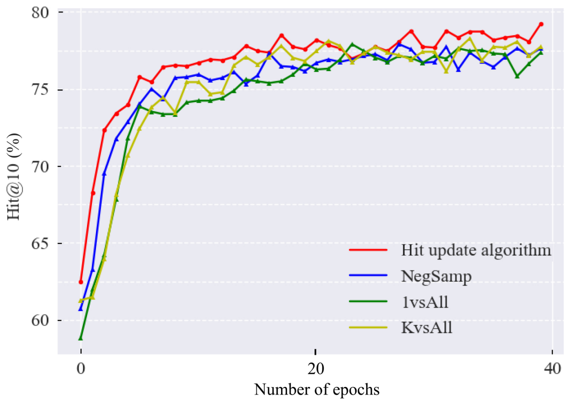


Fig. 3 Convergence rate of reasoning success ratio on WN18RR comparing with negative sampling.



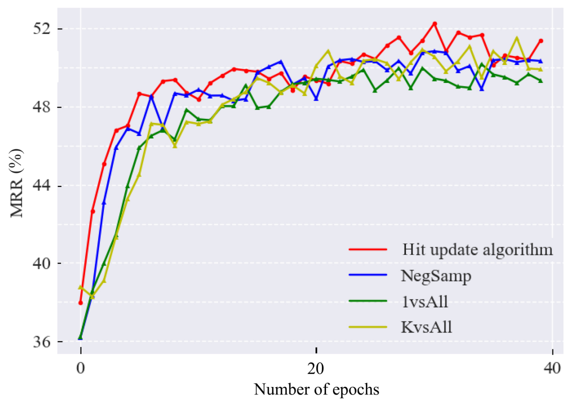


Fig. 4 Convergence rate of reasoning success ratio on CoDEx-S comparing with negative sampling.

based on the agent's inference trajectories, making the agent to counter missing and spurious paths. Meanwhile, the hit reward learning mechanism captures the agent's sampling strategy to offer over-rewarded object entities, and generate negative samples to balance incorrect rewards. Furthermore, we also propose a reward dropout mechanism to explore the diversity of paths and mitigate the influence of missing facts. Our model significantly improves the effectiveness and confidence of rewards. Experimental results on several benchmark knowledge graphs demonstrate that our method is more effective than state-of-the-art walk-based approaches.

In future work, we would like to further investigate the impact of missing facts and address the issues caused by complex rules.

Appendix

A Experimental detail

A1 Extra trick

(1) Pre-trained embedding

It is a consensus that pre-trained embeddings can improve the performance of the multi-hop reasoning. For comparison purposes, we leverage ComplEx[19] and ConvE[40] to pre-train the embedding and reward shaping function $f(e_s, r_q, e_o)$. For each KGs, we set the entity dimension and relation dimension to 200, the training epoch to 1000, the batch size to 512, and the learning rate to 0.003.

(2) Reverse relation

Some inference paths require reverse relations. For example, given a rule *Lebron James* $\xrightarrow{\textit{plays in}}$ *NBA* $\xrightarrow{\textit{part of}_{rev}}$ *Cleveland Cavaliers*, which requires reverse relations *part of*$_{rev}$ to guide the agent to the object entity. Adding reverse relations also helps the agent search more edges that cannot be explored in the original KG due to the limitation of the number of hops.

(3) Beam search

We perform beam search reasoning to infers multiple paths and object entities at once inference process. The agent finally outputs the predicted paths and entities with the maximum score of each beam. This method has proven to be more effective than greedy search.

(4) Action dropout

To enforce effective exploration of paths, we leverage the action dropout mechanism[30] to randomly blocks the action edges of the agent's walking, and replace these edges with a random action extracted from the current action space.

A2 Hyperparameters setting

Part of the hyperparameters used in our model refer to the hyperparameters' settings of previous methods. Besides, some hyperparameters that have a significant impact on performance are searched again. The search bounds are shown in Table A1.

The hyperparameters setting of best models on the five benchmark datasets are presented in Table A2.

B Additional Experiment

Part of models such as M-walk[31] and DRUM[57],

Table A1 Hyperparameters search bound.

Hyperparameter	Search bound
Embedding dropout rate	$[0.1, 0.3]$
Hidden layer dropout rate	$[0.1, 0.3]$
Action dropout rate	$[0.1, 0.5]$
Batch size	$\{128, 256, 512\}$
Number of hops	$\{2, 3\}$
Discount factor λ_h	$\{0.1, 0.3, 0.5, 0.7, 0.9\}$
Discount factor λ_r	$\{0.1, 0.3, 0.5, 0.7, 0.9\}$
IKB update interval γ_h	$[0, 300\,000]$
Fact update number η_h	$\{5, 10, 15, 20, 25, 30\}$
CRB update ratio η_r	$\{0, 0.1, 0.2\}$

Table A2 Hyperparameters used in AInvR.

Hyperparameter	FB15k-237	WN18RR	CoDEx-S	CoDEx-M	CoDEx-L
Entity dimension	200	200	200	200	200
Relation dimension	200	200	200	200	200
History dimension	200	200	200	200	200
Regularization weight	0.02	0.0	0.02	0.02	0.02
Embedding dropout rate	0.3	0.3	0.3	0.3	0.3
Hidden layer dropout rate	0.1	0.1	0.1	0.1	0.1
Action dropout rate	0.5	0.5	0.5	0.5	0.5
Rule reward dropout ratio	0.1	0.1	0.1	0.1	0.1
Fact reward dropout ratio	0.5	0.5	0.5	0.5	0.5
Bandwidth	400	500	400	400	400
Batch size	128	256	128	128	128
Learning rate	0.0015	0.0010	0.0010	0.0010	0.0015
Number of hops	3	3	3	3	3
Beam search size	128	128	128	128	128
Training epoch	100	100	50	100	50
Discount factor λ_r	0.1	0.5	0.3	0.1	0.3
Discount factor λ_h	0.9	0.5	0.7	0.9	0.7
IKB update interval γ_h	20	10	10	20	20
Fact update number η_h	100 000	40 000	30 000	180 000	300 000
Rule sampling number γ_r	1000	1000	1000	1000	1000
CRB update ratio η_r	0.1	0.1	0.1	0.2	0.1

RNNLogic demonstrate excellent performance for KGR. However, those approaches consider a different experience setting for testing and training. To overall compare with the state-of-the-art approaches, we conduct a comparative experiment under the same experimental setting. To be specific, for each triplet (e_s, r_q, e_t) in train set, we reverse the triplet to (e_t, r_q^{inv}, e_s) and add it to the KG. The embeddings and agent are trained by the KG with reverse triples. For each triplet (e_s, r_q, e_t) in test set and valid set, we force the agent to reason two queries $(e_s, r_q, ?)$ and $(e_t, r_q^{inv}, ?)$. The experimental results on FB15k-237 and WN18RR are shown in Table A3.

C Dataset analysis

FB15k-237 is a sub dataset of FB15k[18] extracted from Freebase, where inverse relations are deleted. WN18RR is a subset of WN18[18] with the inverse relations removed, and WN18 is a dataset from WordNet. CoDEx-S, CoDEx-M and CoDEx-L are three datasets containing different amounts of facts taken from Wikipedia and Wikidata.

Table A3 Comparison on FB15k-237 and WN18RR with RNNLogic, M-walk, etc. The best scores of multi-hop approaches are bolded and the best scores of embeddings are underlined.

(%)

Method	FB15k-237			WN18RR		
	Hit@1	Hit@10	MRR	Hit@1	Hit@10	MRR
NeuralLP	17.3	36.1	23.7	36.8	40.8	38.1
DRUM	17.4	36.4	23.8	36.9	41.0	38.2
NLIL	–	32.4	25.0	–	–	–
M-Walk	16.5	–	23.2	41.4	–	43.7
RNNLogic(w/o emb.)	20.8	44.5	28.8	41.4	53.1	45.5
RNNLogic(RotateE)	25.2	53.0	34.4	**44.6**	**55.8**	**48.3**
AInvR(ComplEx)	28.0	53.1	36.3	42.0	51.7	45.1
AInvR(ConvE)	**28.2**	**55.4**	**37.3**	39.1	47.7	41.9
DisMult	15.5	41.9	24.1	39.0	49.0	43.0
ComplEX	15.8	42.8	24.7	41.0	51.0	44.0
ConvE	23.7	50.1	32.5	40.0	52.0	43.0
RotateE	20.5	48.0	29.7	42.2	56.5	47.0
QuatE	27.1	55.6	36.6	43.6	57.2	48.2

Note that, the prediction tasks are more challenging for knowledge graph reasoning with more diversity of facts.

We analyze the number of hops that can be reached using multi-hop reasoning in test sets of five benchmark knowledge graphs, the results are shown in Table A4. In the test sets of FB15k-237 and CoDEx, almost all facts can be reasoned by the agent within 3-hops, which proves that their verification is friendly for multi-hop reasoning. Conversely, at least 25 percents of the facts in WN18RR's test set show non-interpretability within 3-hops, and still 15 percents within 5-hops. Note that, we already complement the reverse relation in those KG.

We compare the differences between walk-based model and embedding model in terms of the inferential and non-inferential test sets. Facts that can be inferred within 3-hops are classified into the inferential test sets and others are classified into the non-inferential test sets. The results are shown in Table A5.

Table A4 Multi-hop reasoning analysis of test sets.

Test set	Triples	2-hops	3-hops	4-hops	5-hops
FB15k-237	20 466	16 261	20 402	20 437	20 438
WN18RR	3134	2007	2335	2475	2658
CoDEx-S	1828	1267	1828	1828	1828
CoDEx-M	10 311	6374	10 306	10 311	10 311
CoDEx-L	30 622	18 931	30 522	30 613	30 622

Table A5 Results of inferential and non-inferential test sets in WN18RR. WN18RR• is the subset of WN18RR's test set, which contains only facts that can be inferred within 3-hops. WN18RR○ is the difference set between WN18RR and WN18RR•.

(%)

Method	Test set	Hit@1	Hit@3	Hit@5	Hit@10	MRR
AInvR	WN18RR	45.8	51.4	53.8	56.9	49.6
	WN18RR•	41.4	44.5	45.9	47.4	43.4
	WN18RR○	41.5	47.2	50.4	54.4	45.6
ComplEx	WN18RR	59.6	66.8	69.9	73.9	64.4
	WN18RR•	55.5	59.4	61.2	63.1	58.1
	WN18RR○	55.4	62.6	66.5	71.0	60.4
ConvE	WN18RR	0	0	0	0.3	0.2
	WN18RR•	0	0	0	0	0
	WN18RR○	1.0	2.4	3.4	6.0	2.5

Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. U19A2059).

References

[1] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, Freebase: A collaboratively created graph database for structuring human knowledge, in *Proc. 2008 ACM SIGMOD Int. Conf. Management of Data*, Vancouver, Canada, 2008, pp. 1247–1250.

[2] F. M. Suchanek, G. Kasneci, and G. Weikum, Yago: A core of semantic knowledge, in *Proc. 16th Int. Conf. World Wide Web*, Banff, Canada, 2007, pp. 697–706.

[3] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, et al., DBpedia—A large-scale, multilingual knowledge base extracted from Wikipedia, *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.

[4] H. Wang, K. Qin, G. Lu, J. Yin, R. Y. Zakari, and J. W. Owusu, Document-level relation extraction using evidence reasoning on RST-GRAPH, *Knowl.-Based Syst.*, vol. 228, p. 107274, 2021.

[5] H. Wang, K. Qin, G. Lu, G. Luo, and G. Liu, Direction-sensitive relation extraction using Bi-SDP attention model, *Knowl.-Based Syst.*, vol. 198, p. 105928, 2020.

[6] Y. Hao, Y. Zhang, K. Liu, S. He, Z. Liu, H. Wu, and J. Zhao, An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge, in *Proc. 55th Annu. Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada, 2017, pp. 221–231.

[7] H. Zhou, T. Young, M. Huang, H. Zhao, J. Xu, and X. Zhu, Commonsense knowledge aware conversation generation with graph attention, in *Proc. 27th Int. Joint Conf. Artificial Intelligence*, Stockholm, Sweden, 2018, pp. 4623–4629.

[8] Y. Cao, X. Wang, X. He, Z. Hu, and T. S. Chua, Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences, in *Proc. of the World Wide Web Conf.*, San Francisco, CA, USA, 2019, pp. 151–161.

[9] H. Wang, F. Zhang, J. Wang, M. Zhao, W. Li, X. Xie, and M. Guo, RippleNet: Propagating user preferences on the knowledge graph for recommender systems, in *Proc. 27th ACM Int. Conf. Information and Knowledge Management*, Torino, Italy, 2018, pp. 417–426.

[10] H. Xu, Z. Cai, R. Li, and W. Li, Efficient CityCam-to-edge cooperative learning for vehicle counting in its, *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 16600–16611, 2022.

[11] Z. Xiong, Z. Cai, D. Takabi, and W. Li, Privacy threat and defense for federated learning with non-i.i.d. data in AIoT, *IEEE Trans. Ind. Inform.*, vol. 18, no. 2, pp. 1310–1321, 2022.

[12] J. Pang, Y. Huang, Z. Xie, Q. Han, and Z. Cai, Realizing the heterogeneity: A self-organized federated learning framework for IoT, *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3088–3098, 2021.

[13] Z. Cai, Z. Xiong, H. Xu, P. Wang, W. Li, and Y. Pan, Generative adversarial networks: A survey toward private and secure applications, *ACM Comput. Surv.*, vol. 54, no. 6, p. 132, 2022.

[14] K. Li, G. Lu, G. Luo, and Z. Cai, Seed-free graph de-anonymiztiation with adversarial learning, in *Proc. 29th ACM Int. Conf. Information & Knowledge Management*, Virtual Event, 2020, pp. 745–754.

[15] Y. Fang, X. Huang, L. Qin, Y. Zhang, W. Zhang, R. Cheng, and X. Lin, A survey of community search over big graphs, *VLDB J.*, vol. 29, no. 1, pp. 353–392, 2020.

[16] M. Zhou, N. Duan, S. Liu, and H. Y. Shum, Progress

in neural NLP: Modeling, learning, and reasoning, *Engineering*, vol. 6, no. 3, pp. 275–290, 2020.

[17] J. Zhang, B. Chen, L. Zhang, X. Ke, and H. Ding, Neural, symbolic and neural-symbolic reasoning on knowledge graphs, *AI Open*, vol. 2, pp. 14–35, 2021.

[18] A. Bordes, N. Usunier, A. Garcia-Durán, J. Weston, and O. Yakhnenko, Translating embeddings for modeling multi-relational data, in *Proc. 26th Int. Conf. Neural Information Processing Systems-Volume 2*, Lake Tahoe, NV, USA, 2013, pp. 2787–2795.

[19] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, Complex embeddings for simple link prediction, in *Proc. 33rd Int. Conf. Machine Learning*, New York, NY, USA, 2016, pp. 2071–2080.

[20] Z. Sun, Z. H. Deng, J. Y. Nie, and J. Tang, RotatE: Knowledge graph embedding by relational rotation in complex space, arXiv preprint arXiv:1902.10197, 2019.

[21] S. M. Kazemi and D. Poole, Simple embedding for link prediction in knowledge graphs, in *Proc. 32nd Int. Conf. Neural Information Processing Systems*, Montréal, Canada, 2018, pp. 4289–4300.

[22] S. Vashishth, S. Sanyal, V. Nitin, and P. Talukdar, Composition-based multi-relational graph convolutional networks, arXiv preprint arXiv:1911.03082, 2020.

[23] L. A. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek, AMIE: Association rule mining under incomplete evidence in ontological knowledge bases, in *Proc. 22nd Int. Conf. World Wide Web*, Rio de Janeiro, Brazil, 2013, pp. 413–422.

[24] P. G. Omran, K. Wang, and Z. Wang, Scalable rule learning via learning representation, in *Proc. 27th Int. Joint Conf. Artificial Intelligence*, Stockholm, Sweden, 2018, pp. 2149–2155.

[25] D. Nathani, J. Chauhan, C. Sharma, and M. Kaul, Learning attention-based embeddings for relation prediction in knowledge graphs, in *Proc. 57th Annu. Meeting of the Association for Computational Linguistics*, Florence, Italy, 2019, pp. 4710–4723.

[26] C. Meilicke, M. W. Chekol, D. Ruffinelli, and H. Stuckenschmidt, Anytime bottom-up rule learning for knowledge graph completion, in *Proc. 28th Int. Joint Conf. Artificial Intelligence*, Macao, China, 2019, pp. 3137–3143.

[27] M. Qu and J. Tang, Probabilistic logic neural networks for reasoning, in *Proc. 33rd Int. Conf. Neural Information Processing Systems*, Vancouver, Canada, 2019, pp. 7712–7722.

[28] W. Xiong, T. Hoang, and W. Y. Wang, DeepPath: A reinforcement learning method for knowledge graph reasoning, in *Proc. 2017 Conf. Empirical Methods in Natural Language Processing*, Copenhagen, Denmark, 2017, pp. 564–573.

[29] R. Das, S. Dhuliawala, M. Zaheer, L. Vilnis, I. Durugkar, A. Krishnamurthy, A. Smola, and A. McCallum, Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning, arXiv preprint arXiv:1711.05851, 2018.

[30] X. V. Lin, R. Socher, and C. Xiong, Multi-hop knowledge graph reasoning with reward shaping, in *Proc. 2018 Conf. Empirical Methods in Natural Language Processing*, Brussels, Belgium, 2018, pp. 3243–3253.

[31] Y. Shen, J. Chen, P. S. Huang, Y. Guo, and J. Gao, M-Walk: Learning to walk over graphs using Monte Carlo tree search, in *Proc. 32nd Int. Conf. Neural Information Processing Systems*, Montréal, Canada, 2018, pp. 6787–6798.

[32] H. Wang, S. Li, R. Pan, and M. Mao, Incorporating graph attention mechanism into knowledge graph reasoning based on deep reinforcement learning, in *Proc. 2019 Conf. Empirical Methods in Natural Language Processing and the 9th Int. Joint Conf. Natural Language Processing*, Hong Kong, China, 2019, pp. 2623–2631.

[33] X. Lv, X. Han, L. Hou, J. Li, Z. Liu, W. Zhang, Y. Zhang, H. Kong, and S. Wu, Dynamic anticipation and completion for multi-hop reasoning over sparse knowledge graph, in *Proc. 2020 Conf. Empirical Methods in Natural Language Processing* (*EMNLP*), Virtual Event, 2020, pp. 5694–5703.

[34] D. Lei, G. Jiang, X. Gu, K. Sun, Y. Mao, and X. Ren, Learning collaborative agents with rule guidance for knowledge graph reasoning, in *Proc. 2020 Conf. Empirical Methods in Natural Language Processing*, Virtual Event, 2020, pp. 8541–8547.

[35] Z. Hou, X. Jin, Z. Li, and L. Bai, Rule-aware reinforcement learning for knowledge graph reasoning, in *Proc. of the Findings of the Association for Computational Linguistics*: *ACL-IJCNLP 2021*, Virtual Event, 2021, pp. 4687–4692.

[36] Y. Wang, Y. Yao, H. Tong, F. Xu, and J. Lu, A brief review of network embedding, *Big Data Mining and Analytics*, vol. 2, no. 1, pp. 35–47, 2019.

[37] H. Wang, Z. Cao, Y. Zhou, Z. K. Guo, and Z. Ren, Sampling with prior knowledge for high-dimensional gravitational wave data analysis, *Big Data Mining and Analytics*, vol. 5, no. 1, pp. 53–63, 2022.

[38] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, Learning entity and relation embeddings for knowledge graph completion, in *Proc. 29th AAAI Conf. Artificial Intelligence*, Austin, TX, USA, 2015, pp. 2181–2187.

[39] M. Nickel, V. Tresp, and H. P. Kriegel, A three-way model for collective learning on multi-relational data, in *Proc. 28th Int. Conf. Machine Learning*, Bellevue, WA, USA, 2011, pp. 809–816.

[40] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, Convolutional 2D knowledge graph embeddings, in *Proc. 32nd AAAI Conf. Artificial Intelligence and 30th Innovative Applications of Artificial Intelligence Conf. and 8th AAAI Symp. Educational Advances in Artificial Intelligence*, New Orleans, LA, USA, 2018, pp. 1811–1818.

[41] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, Modeling relational data with graph convolutional networks, in *Proc. 15th Int. Conf. Semantic Web*, Heraklion, Greece, 2018, pp. 593–607.

[42] F. Yang, Z. Yang, and W. W. Cohen, Differentiable learning of logical rules for knowledge base reasoning, in *Proc. 31st Int. Conf. Neural Information Processing Systems*, Long Beach, CA, USA, 2017, pp. 2316–2325.

[43] N. Lao, T. Mitchell, and W. Cohen, Random walk inference and learning in a large scale knowledge base, in *Proc. Conf. Empirical Methods in Natural Language Processing*, Edinburgh, UK, 2011, pp. 529–539.

[44] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An*

Introduction. 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[45] K. Zhu and T. Zhang, Deep reinforcement learning based mobile robot navigation: A review, *Tsinghua Science and Technology*, vol. 26, no. 5, pp. 674–691, 2021.

[46] B. Fang, X. Wei, F. Sun, H. Huang, Y. Yu, and H. Liu, Skill learning for human-robot interaction using wearable device, *Tsinghua Science and Technology*, vol. 24, no. 6, pp. 654–662, 2019.

[47] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, Sequence level training with recurrent neural networks, arXiv preprint arXiv:1511.06732, 2016.

[48] S. Hochreiter and J. Schmidhuber, Long short-term memory, *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[49] Z. Li, G. Zhang, W. Z. Wu, and N. Xie, Measures of uncertainty for knowledge bases, *Knowl. Inform. Syst.*, vol. 62, no. 2, pp. 611–637, 2020.

[50] C. Finn, S. Levine, and P. Abbeel, Guided cost learning: Deep inverse optimal control via policy optimization, in *Proc. 33^{rd} Int. Conf. Machine Learning*, New York, NY, USA, 2016, pp. 49–58.

[51] R. J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, *Mach. Learn.*, vol. 8, no. 3, pp. 229–256, 1992.

[52] K. Toutanova, D. Chen, P. Pantel, H. Poon, P. Choudhury, and M. Gamon, Representing text for joint embedding of text and knowledge bases, in *Proc. 2015 Conf. Empirical Methods in Natural Language Processing*, Lisbon, Portugal, 2015, pp. 1499–1509.

[53] T. Safavi and D. Koutra, CoDEx: A comprehensive knowledge graph completion benchmark, in *Proc. 2020 Conf. Empirical Methods in Natural Language Processing*, Virtual Event, 2020, pp. 8328–8350.

[54] S. Zhang, Y. Tay, L. Yao, and Q. Liu, Quaternion knowledge graph embeddings, in *Proc. 33^{rd} Int. Conf. Neural Information Processing Systems*, Vancouver, Canada, 2019, pp. 2735–2745.

[55] M. Qu, J. Chen, L. P. Xhonneux, Y. Bengio, and J. Tang, RNNLogic: Learning logic rules for reasoning on knowledge graphs, arXiv preprint arXiv:2010.04029, 2021.

[56] D. Ruffinelli, S. Broscheit, and R. Gemulla, You CAN teach an old dog new tricks! On training knowledge graph embeddings, https://openreview.net/forum?id=BkxSmlBFvr.

[57] A. Sadeghian, M. Armandpour, P. Ding, and D. Z. Wang, DRUM: End-to-end differentiable rule mining on knowledge graphs, in *Proc. 33^{rd} Int. Conf. Neural Information Processing Systems*, Vancouver, Canada, 2019, pp. 15347–15357.

Hao Zhang received the BEng degree from Southwest Jiaotong University, China in 2020. He is now a master student at the School of Computer Science and Engineering, University of Electronic Science and Technology of China. His research interests include natural language reasoning and reinforcement learning.

Guoming Lu received the PhD degree from University of Electronic Science and Technology of China, in 2006. He is now an associate professor at the School of Computer Science and Engineering, University of Electronic Science and Technology of China. His main research interests include high performance computing, natural language processing, and reinforcement learning.

Ke Qin received the PhD degree from University of Electronic Science and Technology of China, in 2010. He is now a professor at the School of Computer Science and Engineering, University of Electronic Science and Technology of China. His main research interests include neural networks, machine learning, and big data.

Kai Du received the BEng degree from Northwestern Polytechnical University, Xi’an, China in 2019. He is now a master student at the School of Computer Science and Engineering, University of Electronic Science and Technology of China. His main research interests include text summarization and reinforcement learning.