Off-Policy Differentiable Logic Reinforcement Learning

Li Zhang[1], Xin Li✉⋆[1], Mingzhong Wang[2], and Andong Tian[3]

[1] School of Computer Science, Beijing Institute of Technology, China
[2] USC Business School, University of the Sunshine Coast, Australia
[3] Ubisoft China AI & Data Lab, China
{3120181069,xinli}@bit.edu.cn,mwang@usc.edu.au,an-dong.tian@ubisoft.com

Abstract. In this paper, we proposed an Off-Policy Differentiable Logic Reinforcement Learning (OPDLRL) framework to inherit the benefits of interpretability and generalization ability in Differentiable Inductive Logic Programming (DILP) and also resolves its weakness of execution efficiency, stability, and scalability. The key contributions include the use of approximate inference to significantly reduce the number of logic rules in the deduction process, an off-policy training method to enable approximate inference, and a distributed and hierarchical training framework. Extensive experiments, specifically playing real-time video games in Rabbids against human players, show that OPDLRL has better or similar performance as other DILP-based methods but far more practical in terms of sample efficiency and execution efficiency, making it applicable to complex and (near) real-time domains.

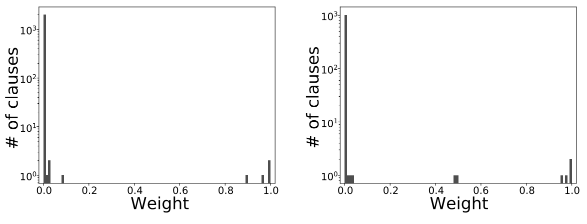
Keywords: Deep Reinforcement Learning · Interpretable Reinforcement Learning · Neural-Symbolic AI.

1 Introduction

Despite the advantages and benefits of Deep Reinforcement Learning (DRL), its successful application and deployment need to address the challenges including: (1) Interpretability. The use of deep neural networks makes the learned policies difficult to interpret and verify, restricting the application of DRL in many real-world domains which require clear scientific interpretation, e.g., healthcare and medical systems. (2) Generalization. The learned policies tend to "over-fit" the training environment, leading the performance of learned polices to drastically decrease even when the test environment slightly changes from the training environment. (3) Sample efficiency. DRL methods generally require massive numbers of samples to explore the environment.

Differentiable Inductive Logic Programming (DILP) [7,8,29,6,22] has been integrated into DRL frameworks to achieve better interpretability and generalization. Trained with standard back-propagation, DILP provides a special formulation of a function approximator, which generates interpretable and verifiable

⋆ Xin Li is the corresponding author.



(a) On (b) Stack

Fig. 1. Weight distributions on two benchmark tasks.

logic rules via training samples. The logic rules also behave as a form of regularization, helping to mitigate over-fitting and improve the generalization ability. Integrating DILP into DRL helps to interpret policies, making the agent's behavior more verifiable and robust.

[7,8] proposed ∂ILP to learn logic rules from noisy data, demonstrating the strength of DILP in interpretability and generalization. [17] introduced Neural Logic Reinforcement Learning (NLRL) which applies DILP in sequential decision making tasks and trains it via vanilla policy gradient [38]. [6] proposed Neural Logic Machines which trades off some interpretability for better scalability in comparison with NLRL.

However, the integration of DILP and DRL suffers from its execution efficiency, stability, and scalability, making it infeasible to many applications requiring real-time or near real-time responses, such as autonomous driving and game playing.

– **Execution efficiency**. DILP takes a top-down and generate-and-test approach, which first generates all potential logic rules and then finds the optimal subset. In general, the number of potential rules is relatively large and the computational cost of DILP is much higher than Multi-layer Perceptron (MLP) networks for policy learning. For an MLP with n linear layers and h hidden neurons at each layer, e.g. in a medium-size problem where $n = 3$ and $h = 256$, its forward computational complexity is $O(nh^2) \approx 2 \times 10^5$. In comparison, the complexity of the DILP, which deduces n steps with l logic rules and each rule matches k cases, m ground atoms, is $O(nmlk) \approx 2 \times 10^7$ for a medium-size problem where $n = 5$, $l = 2000$, $k = 10$, and $m = 200$.
  The number of logic rules in DILP is generally relatively large, resulting in long periods of policy response. This prohibits its application in real-time or near real-time domains. To address this issue, we proposed to reduce the number of rules by at least an order of magnitude. In fact, our study of all potential rules after training revealed that only a small number of rules are non-trivial to induction. Fig. 1 depicts the weight distribution, in which the experiment task "On" showed that 99.60%(1981/1989) rules have weight less than 0.01, thus being negligible. Therefore, we proposed the solution of *approximate inference* which extends the technique of network pruning [9,40,21,20]

to DILP. It measures the importance of logic rules and maintains a dynamic set of rules at run time. Due to the boost of execution efficiency, our model can provide (near) real-time policy responses, which is difficult for previous Neuro-Symbolic methods.
In addition, we developed a distributed reinforcement learning framework[4] which decouples the learning (experience utilization) and acting (experience collection) processes, enabling their parallel execution on different machines to reduce the training time.

– **Stability**. In comparison with MLP, DILP, specifically, after the use of approximate inference, is much harder to be optimized and its learning curve oscillates intensely (see Sec. 4). Therefore, we proposed to adopt Maximum Entropy Reinforcement Learning (MERL) approach [10,11,12,42,39] which augments an entropy term to the objective of standard reinforcement learning (cumulative reward), encouraging policies to consider both optimal and sub-optimal actions. MERL can decrease overall estimation errors to stabilize the training, as demonstrated by [41].
– **Scalability**. It is difficult to apply DILP in large-scale/continuous domains due to its high computational cost. Thus, we further extended DILP with hierarchical reinforcement learning [1,5,35,28], to decompose the entire task to simpler sub-tasks, making it possible to employ DILP in complex domains, such as video games.

Although approximate inference helps to significantly reduce the time required for the policy response, it also causes existing on-policy RL algorithms to fail as agents cannot sample actions from online-policy due to non-trivial feature of approximate inference. Our empirical results showed that the errors can be ignored if the model is trained sufficiently. However, errors are inevitable in the early learning stages. Thus, a well-designed off-policy training method becomes the key to the success of approximate inference. Besides, off-policy training also helps to greatly improve sample efficiency by reusing the samples in the experience replay buffer [25,15,37,16].

The natural solutions of policy gradient algorithms [32,31,33,24] are not directly feasible to train the policy expressed by DILP, or differentiable logic policy (DLP), due to the requirement of off-policy training. Therefore, Q-learning [25,37,15], a classic off-policy algorithm, is adopted. To enable Q-learning to work seamlessly with DLP in the MERL framework, we used the Soft Q-Learning and Soft Policy Iteration theorem [10,11,12] as the bridge connecting the Q-value and policy.

In summary, we proposed Off-Policy Differentiable Logic Reinforcement Learning (OPDLRL), which inherits the benefits of interpretability and generalization ability from DILP but also resolves its weakness of execution efficiency, stability, and scalability, making OPDLRL applicable to complex and (near) real-time domains. The key contributions of the paper include:

– We proposed the use of approximate inference to significantly improve the execution efficiency, making our model feasible in (near) real-time applica-

[4] Details can be found in the Supplementary Material.

tions. To achieve approximate inference and improve sample efficiency, we proposed an off-policy training method in MERL framework, which uses the soft Q-learning and soft policy iteration theorem to connect the policy and Q-value.
- We developed a distributed and hierarchical training framework for DILP, which significantly improves the training efficiency and makes our model feasible in complex application domains.
- We tested OPDLRL extensively with both benchmark tasks and complex domain tasks. The results showed that OPDLRL significantly outperformed other DILP-based DRL algorithms regarding both performance and sample efficiency in Block Manipulation and Car Avoiding tasks, and OPDLRL could learn to play Rabbids[5] video game and competed with human players successfully while MLP-based and other DILP-based solutions failed.

2 Preliminary

2.1 First-Order Logic Programming

An **atom** $\alpha = p(t_1, \dots, t_n)$ consists of the **predicate** (relation) p and **terms** (entities) $t_1, \dots, t_n$, where t_i is a **variable** or a **constant**. A **ground** atom has all terms as constants. An **extensional** predicate is defined by a set of ground atoms while an **intensional** predicate is defined by some clauses. A **clause** $\alpha \leftarrow \alpha_1, \dots, \alpha_n$ consists of the **head** α and **body** $\alpha_1, \dots, \alpha_n$, meaning that the head is true if all atoms of body are true. A **deduction** starts from a set of ground atoms, and applies a set of clauses to generate more ground atoms.

2.2 Differentiable Inductive Logic Programming

An Inductive Logic Programming (ILP) problem [26,27] is a tuple $\{\mathcal{B}, \mathcal{P}, \mathcal{N}\}$, in which $\mathcal{B}, \mathcal{P}, \mathcal{N}$ are sets of ***background***, ***positive*** and ***negative*** atoms, respectively. DILP denotes the truth of an atom as $p \in [0, 1]$, representing the probability that the atom is true. Let G be the set of all ground atoms, and a **valuation** is a vector $\mathbf{v} \in [0, 1]^{||G||}$ representing the probability of all ground atoms. A **language frame** defines target predicates (objective of ILP), extensional predicates, their arity and constants. A **program template** defines available auxiliary predicates, their arity and rule templates. A **rule template** describes whether intensional predicates can be used, and claims the number of existentially quantified variables. With language frames and program templates specified, the set of all possible clauses can be generated, and DILP assigns a probability/confidence for each clause or combination of clauses. The deduction (forward computation) of DILP evaluates the results of all clauses and weights them by corresponding confidence, which is trained to maximize the probability for positive and negative atoms to be satisfied via standard back-propagation. Please refer to [7] for more details.

[5] `https://en.wikipedia.org/wiki/Raving_Rabbids`

2.3 Maximum Entropy Reinforcement Learning

A Markov Decision Process (MDP) is defined as a tuple $\{S, A, T, R\}$, where S is the state space, A is the action space, T is the transition function, and R is the reward function. In standard RL, the objective is to find a policy that can maximize the expectation of cumulative discount reward $E[\sum_{t=0}^{T} \gamma^t r_t^\pi]$, where γ is the discount factor and r_t^π is the reward at time step t. In maximum entropy RL, the objective is augmented with an entropy term: $E[\sum_{t=0}^{T} \gamma^t [r_t^\pi + \alpha \mathcal{H}(\pi(\cdot|s_t))]]$, where α is the temperature/weighting parameter and $\mathcal{H}(\pi(\cdot|s_t))$ is the entropy of action distribution of a policy π given state s_t.

The entropy augmented objective can be optimized via soft policy iteration [11][12][10]. In the soft policy evaluation step, soft Q-value is computed based on policy π to evaluate its performance:

$$\begin{aligned} Q^\pi(s,a) &= r(s,a) + \gamma E_{s' \sim T}[V^\pi(s')] \\ V^\pi(s) &= E_{a \sim \pi}[Q^\pi(s,a) - \alpha \log \pi(a|s)] \end{aligned} \tag{1}$$

In the soft policy improvement step, a new policy is generated by minimizing the Kullback–Leibler (KL) divergence between the action distribution and the exponential of soft Q-value under the old policy π for each state:

$$\pi_{new} = \operatorname*{argmin}_{\pi' \in \Pi} D_{KL}(\pi'(\cdot|s) || \frac{\exp(\frac{1}{\alpha} Q^\pi(s,\cdot))}{Z^\pi(s)}) \tag{2}$$

where Π is the set of all potential policies and $Z^\pi(s)$ is the partition function normalizing the distribution.

3 Off-Policy Differentiable Logic Reinforcement Learning

Fig. 2 shows an overview of the OPDLRL framework. The following sections explain its components.

3.1 Differentiable Logic Policy

To implement logic programming in DRL, symbolic compilers are first required to align logic expressions with reinforcement learning environments. In this paper, a symbolic state is represented by a set of ground atoms $\bar{s} \subseteq G_s$, where G_s is the set of all ground state atoms. A state compiler $S \rightarrow \{0,1\}^{||G_s||}$ translates environment states into symbolic states. A symbolic action is represented by a ground atom $\bar{a} \in G_a$ where G_a is the set of all ground action atoms. Once a symbolic action is decided by policies, it needs to be translated to an environment action by an action compiler $G_a \rightarrow A$. These compilers are usually hand-crafted or initiated by a pre-trained neural network. A set of background atoms is provided to describe the relations of constants regarding the task.

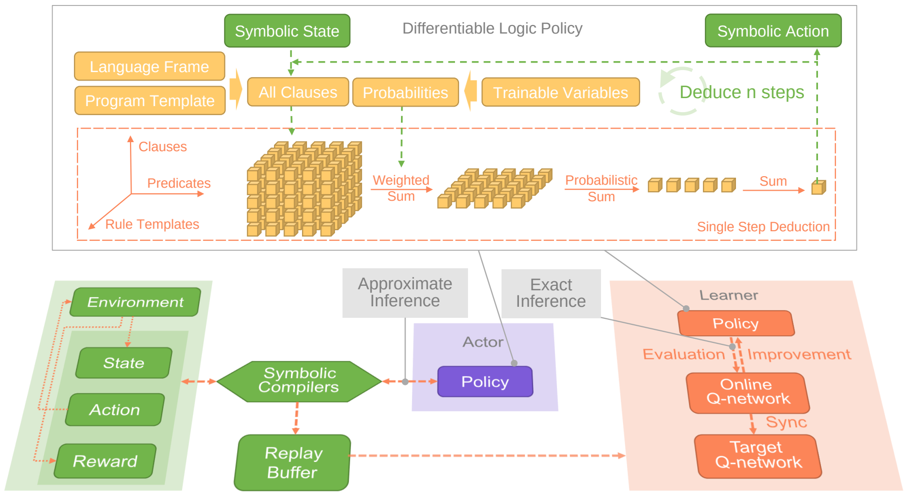


Fig. 2. Framework of Off-Policy Differentiable Logic Reinforcement Learning.

With the symbolic compilers, a MDP becomes a First-order MDP (FOMDP) problem [2,18,30] in a stricter form. We define the target and extensional predicates of the language frame as the action and state predicates of FOMDP respectively, and define the auxiliary predicates that help to represent policies in the program template. Thereafter, all possible clauses can be generated with the language frame and program template.

We denote the policy of the FOMDP as π_θ, the i-th rule template of predicate e as τ_i^e, and the corresponding set of clauses as $\mathit{\Gamma}_i^e$. For each $\mathit{\Gamma}_i^e$, we initialize a vector θ_i^e of length $||\mathit{\Gamma}_i^e||$ by Gaussian distribution with mean 0.00 and standard deviation 0.05. The probability (confidence) of the j-th clause in $\mathit{\Gamma}_i^e$ is parameterized as:[6]

$$p_i^e(j) = \frac{\theta_i^e(j)^2}{||\theta_i^e||^2}. \tag{3}$$

In a single deduction step, the output valuation vector is computed by:

$$\mathbf{v} = \min(1, \mathbf{v}_0 + \sum_e \bigoplus_i \sum_j p_i^e(j)\mathbf{v}_{e,i,j}) \tag{4}$$

where $\mathbf{v}_{e,i,j}$ denotes the valuation inferred by the j-th clause in the clause set $\mathit{\Gamma}_i^e$ according to input valuation of current deduction step. The weighted sum amalgamates the valuations for different clauses but with the same rule template, the probabilistic sum $\oplus$ $(x \oplus y = x + y - xy)$ amalgamates the valuations for different rule templates, and finally, the valuations for different predicates are summed up. $\mathbf{v}_0$ is the initial valuation which is a multi-hot vector defined

[6] The comparison of different parameterization methods can be found in the Supplementary Material.

jointly by the current symbolic state and background atoms, and it is added in each deduction step to steer off the local optima as suggested in [17]. The final valuation is clipped to $[0,1]$ to prevent the overflow and treated as input valuation to the next deduction step.

3.2 Approximate Inference

The inference process in DLP involves all clauses, thus significantly more expensive than an MLP policy regarding the computational cost. We propose approximate inference to reduce the response time of the inference process, given as:

$$\mathbf{v} = \min(1, \mathbf{v}_0 + \sum_{e} \bigoplus_{i} \sum_{k \in \{j | p_i^e(j) > \eta\}} p_i^e(k) \mathbf{v}_{e,i,k}) \tag{5}$$

where η is the threshold to filter out the negligible clauses to the deduction.

3.3 Off-Policy Training

Sec. 1 has shown approximate inference can significantly reduce the computation time of the deduction, but cannot leverage existing on-policy RL algorithms due to its non-trivial feature. Thus, we propose an off-policy approach to train the approximate inference-facilitated DLP.

Our off-policy training process stems from soft policy iteration theorem, which connects Q-learning and policy gradient method in MERL framework. Therefore, the solution benefits from both Q-learning, which implements off-policy to enable approximate inference and improve sample efficiency, and policy gradient, which applies a separate policy network to achieve better interpretability.

For the soft policy evaluation, we use MLP networks to approximate soft Q-value and train it by minimizing TD-error (with an augmented entropy term), given as:

$$\begin{aligned} J_Q(\vartheta) &= E_{(s,a,r,s') \sim D}[\frac{1}{2}(r + \gamma V(s') - Q_\vartheta(s,a))^2] \\ V(s) &= \sum_a \pi_\theta(a|s)[Q_{\bar{\vartheta}}(s,a) - \alpha \log \pi_\theta(a|s)] \end{aligned} \tag{6}$$

where D is the replay buffer, ϑ and $\bar{\vartheta}$ are the parameters of online and target Q-network respectively, and $\pi_\theta(a|s)$ is the action distribution computed by the exact inference of DLP. Target Q-network [25] periodically syncs with online Q-network to stabilize the learning of soft Q-value. Note that our method has two expectation terms in Eq. (1): $E_{s' \sim T}$ is computed by Monte-Carlo estimation (sampling from replay buffer), and $E_{a \sim \pi}$ is computed as the weighted sum of networks' output values for all actions (discrete space).

For the soft policy improvement, we ignore the constant partition function $Z^\pi(s)$ and take the logarithm of Eq. (2) to jointly optimize π_θ and Q_ϑ, given as:

$$J_\pi(\theta) = E_{s \sim D}[\sum_a \pi_\theta(a|s)(\alpha \log \pi_\theta(a|s) - Q_\vartheta(a|s))] \tag{7}$$

The expectation term in KL-divergence in Eq. (2) is computed as weighted sum.

3.4 Hierarchical Policy Implementation

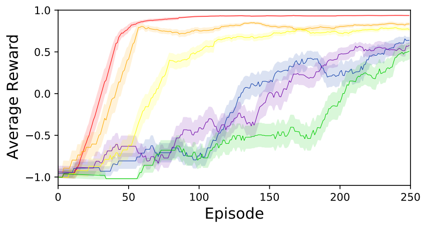
To enable OPDLRL to work effectively for complex tasks with large/continuous state/action space and (near) real-time response requirement, we further develop a hierarchical policy implementation in which the agent's policy is formulated as being hierarchical. OPDLRL works at the top level to determine which sub-policy should be taken according to the symbolic state (high-level state) and tries to minimize external reward. A sub-policy (skill) is pre-trained to achieve a sub-goal and it takes the primary states and outputs primary actions. Symbolic compilers perform the translation between different hierarchies.

4 Experiments

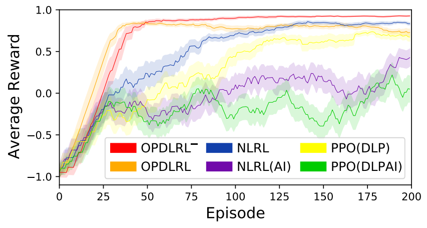
OPDLRL, its variants, and other relevant methods have been extensively tested in two relational RL benchmark tasks, namely Block Manipulation and Car Avoiding, and one real-time task Rabbids, which is a popular video game consisting of many mini-games. In this paper, we used OPDLRL to play the bumper-car game in Rabbids against behavior tree [4] agent and human players.

Table 1. Performance comparison on different tasks. Items in table are the mean and standard deviation for 300 different runs.

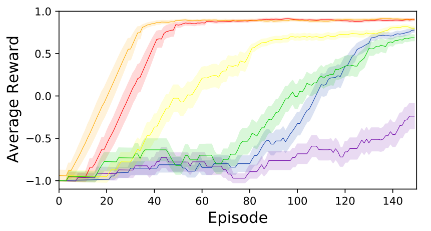
	On	Stack	Unstack	Car Avoiding
OPDLRL$^-$	0.936(±0.01)	0.941(±0.03)	0.950(±0.01)	0.960(±0.00)
OPDLRL	0.934(±0.01)	0.942(±0.03)	0.950(±0.01)	0.960(±0.00)
NLRL	0.881(±0.05)	0.908(±0.05)	0.909(±0.05)	0.947(±0.16)
NLRL(AI)	0.816(±0.08)	0.817(±0.12)	0.902(±0.05)	0.488(±0.82)
SAC	0.933(±0.01)	0.947(±0.02)	0.953(±0.01)	0.960(±0.00)
PPO	0.940(±0.00)	0.957(±0.05)	0.960(±0.00)	0.960(±0.00)
PPO(DLP)	0.940(±0.00)	0.870(±0.09)	0.897(±0.06)	0.952(±0.11)
PPO(DLPAI)	0.888(±0.05)	0.862(±0.09)	0.898(±0.05)	0.953(±0.11)
	On+	**Stack+**	**Unstack+**	**Car Avoiding+**
OPDLRL$^-$	0.936(±0.01)	0.953(±0.03)	0.950(±0.02)	0.960(±0.00)
OPDLRL	0.935(±0.01)	0.948(±0.04)	0.949(±0.02)	0.960(±0.00)
NLRL	0.880(±0.05)	0.908(±0.05)	0.896(±0.06)	0.947(±0.16)
NLRL(AI)	0.805(±0.09)	0.800(±0.13)	0.894(±0.06)	0.465(±0.84)
SAC	−0.985(±0.23)	−0.390(±0.79)	0.295(±0.66)	−0.896(±0.44)
PPO	−0.821(±0.49)	−0.437(±0.77)	−0.989(±0.22)	−0.974(±0.22)
PPO(DLP)	0.940(±0.00)	0.906(±0.08)	0.894(±0.06)	0.946(±0.16)
PPO(DLPAI)	0.877(±0.06)	−0.866(±0.46)	0.905(±0.06)	0.946(±0.16)



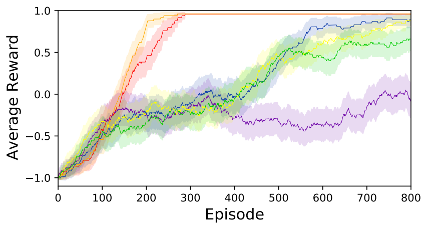
(a) On



(b) Stack



(c) Unstack



(d) Car Avoiding

Fig. 3. Learning curve in different tasks.

4.1 Methods for Evaluation

8 methods have been experimented, including: i) **OPDLRL**: the method we proposed. ii) **OPDLRL$^-$**: OPDLRL without approximate inference. iii) **NLRL**: refer to Sec. 1. iv) **NLRL(AI)**: NLRL with approximate inference. v) **SAC**: Soft Actor Critic [11,12,3], a state-of-the-arts off-policy RL algorithm based on MLP networks. vi) **PPO**: Proximal Policy Optimization [32], a state-of-the-arts on-policy RL algorithm based on MLP networks. vii) **PPO(DLP)**: PPO with differentiable logic policy. viii) **PPO(DLPAI)**: PPO(DLP) with approximate inference. Note that all the compared methods share the same observation space, task hierarchies and pre-trained subpolicies for fairness.

4.2 Experiment Setting

Block Manipulation In Block Manipulation, the agent keeps on moving the top block of a pile until the goal block state is achieved. The constants include $\{a, b, c, d, floor\}$ where a, b, c, d are blocks. The predicates include a state predicate on(X, Y) representing block X is on top of Y and Y can be a block or $floor$, an action/target predicate move(X, Y) representing moving block X to the top of Y, and floor(X) representing whether X is $floor$ and the background is $\{\text{floor}(floor)\}$. The agent will get -0.02 reward for each step if it cannot achieve the goal within 50 steps, and $+1$ reward when it achieves.

There are three kinds of goals in the experiments: (1) **On** task: The goal state is having block X right on top of block Y which is represented as an additional background predicate goal(X, Y). The initial state has all blocks in a pile. (2)

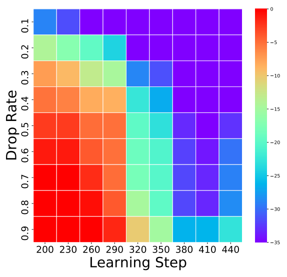


Fig. 4. Impact of approximate inference. Colors are computed via Eq. (10) with log scale.

Stack task: The goal state is having all blocks in a pile and the initial state has all blocks on floor. (3) **Unstack** task: The goal state is having all blocks on floor and the initial state has all blocks in a pile.

Car Avoiding A Car Avoiding task has two cars in a circular platform. The goal of the agent is to control one car to occupy the center without colliding with the opponent car. The platform is divided into 12 regions which are denoted as (X, Y) where $X \in \{a, b, c\}$ describes the distance levels from center and $Y \in \{0, 1, 2, 3\}$ describes quadrants. The predicates include: $\text{me}(X, Y)$ and $\text{enemy}(X, Y)$ for the position of agent and opponent respectively, forward(), backward(), left(), and right() for actions of going forward, backward, turning left and right respectively, and $\text{outer}(X, Y)$ for the relation between a, b, c with the background as $\{\text{outer}(a, b), \text{outer}(b, c)\}$. In the initial state, the agent is at $(c, 0)$ and opponent is at $(a, 0)$. The opponent car will move along $(a, 0) \to (b, 0) \to (c, 0)$ and remain at $(c, 0)$. The agent gets -1 reward if it is hit by its opponent and $+1$ reward if it occupies the center $((a, *))$. The game terminates if the agent is hit or fails to reach the center within 50 steps.

4.3 Results and Analysis

Performance Table. 1 shows the performance results of OPDLRL[7] and other methods. The top section shows the experiments with the same training and test environments. The performance of OPDLRL and OPDLRL$^-$ is very close to the best in all tasks. MLP-based methods (SAC and PPO) perform slightly better than OPDLRL in this setting as MLP has a more flexible structure and is easy to be optimized. Note that the number of trained parameters in MLP is much larger than that in DILP. In the experiments, MLP-based approaches have about 5×10^4 parameters while DILP requires only 2×10^3, indicating a better fitting ability in MLP. In comparison with the DILP-based methods (NLRL, NLRL(AI), PPO(DLP) and PPO(DLPAI)), OPDLRL and OPDLRL$^-$

[7] We used the same setting of hyper-parameters for all tasks.

show evident advantages as they adopt the entropy-augmented reward and optimize with soft policy iteration to reduce the overall estimation error and stabilize the training, as discussed by [41].

Interpretability The logic rules learned by our method in tasks **On** and **Car Avoiding**[8] are as follows (for concise, we only show the rules with probability greater than 0.1):

On:

$$\begin{aligned} 1.00 &: \mathrm{aux2}(X) \leftarrow \mathrm{on}(X,Y), \mathrm{on}(Y,Z) \\ 1.00 &: \mathrm{aux1}(X) \leftarrow \mathrm{aux2}(X), \mathrm{top}(X) \\ 0.97 &: \mathrm{move}(X,Y) \leftarrow \mathrm{floor}(Y), \mathrm{aux1}(X) \\ 0.90 &: \mathrm{move}(X,Y) \leftarrow \mathrm{goal}(X,Y), \mathrm{top}(X) \end{aligned} \tag{8}$$

where aux1 and aux2 are the auxiliary predicates induced by our method. aux2(X) is true if there is a Y such that X is on the top of Y and Y is on the top of Z. Y on Z means Y is not $floor$ (Y is a block), thus aux2(X) means X is on the top of a block (X is not on $floor$). aux1(X) is true if X is not on $floor$ (aux2(X)) and X is the top of a pile (top(X)). The behavior of an agent can be interpreted as: 1) For a pile with 2 or more blocks, move the top block to $floor$. 2) If the goal is to have X on Y and X is movable (X is the top block), then move X to Y.

Car Avoiding:

$$\begin{aligned} 1.00 &: \mathrm{forward}() \leftarrow \mathrm{enemy}(X,Y), \mathrm{outer}(Z,X) \\ 0.58 &: \mathrm{left}() \leftarrow \mathrm{me}(X,Y), \mathrm{enemy}(Z,Y) \\ 0.59 &: \mathrm{right}() \leftarrow \mathrm{me}(X,Y), \mathrm{enemy}(Z,Y) \end{aligned} \tag{9}$$

The goal of an agent is to control one car to occupy the center without colliding with the opponent car. The behavior of an agent can be interpreted as: 1) If the agent and opponent are in the same quadrant, then move left or right. 2) If the opponent is not in $(a, *)$ then move forward.

Generalization The bottom section of Table. 1 shows the experiments with different training and test initial state, thus evaluating the generalization ability of models. Specifically, (1) **On+**: changing the order of blocks in the test. (2) **Stack+**: using 2 piles in the test while 4 piles for the training. (3) **Unstack+**: using 2 piles in the test while 1 pile for the training. (4) **Car Avoiding+**: changing the initial position of two cars in the test. OPDLRL and OPDLRL$^-$ achieve the best performance in this new task settings as our methods utilize logic expressions to capture the essence of task operations, resulting in better generalization. The performance of MLP-based methods (SAC and PPO) decreases drastically due to their over-fitting to the training environment. PPO(DLP) has a slightly better performance than ours in task **On** and **On+** but lags in all other tasks. Note that the loss function of PPO also includes an entropy term but it is used as a regularization term to help exploration.

[8] The induced rules of other tasks are included in the Supplementary Material.

Off-Policy Training Fig. 3 depicts the learning curves of all DILP-based methods on four tasks. OPDLRL and OPDLRL$^-$ substantially outperform others in all tasks regarding sample efficiency (convergence), indicating the effectiveness of the off-policy training and the approximate inference strategy in our framework. For each task, NLRL(AI)/PPO(DLPAI) perform remarkably worse than their corresponding variant without approximate inference because NLRL and PPO(DLP) apply on-policy training but the use of approximate inference requires off-policy, resulting in the outstanding performance loss. The learning curves of OPDLRL and OPDLRL$^-$ are similar as they both apply off-policy training. In fact, OPDLRL converges faster than OPDLRL$^-$ in **Stack**, **Unstack**, and **Car Avoiding** tasks because of efficient *Exploration-Exploitation*: the noisy produced by approximate inference helps policy to diverge from the local-optima in early training stages. Moreover, the use of approximate inference in OPDLRL can significantly reduce the response/deduction time of policies due to dropping a large number of negligible rules (90% potential rules were discarded in the experiments without compromising the performance). SAC and PPO have similar convergence patterns as our methods but with worse generalization. Therefore, they are not included in the comparison.

Impact of Approximate Inference Fig. 4 illustrates the impact of approximate inference by measuring the difference between the policy π with full inference and $\hat{\pi}$ with approximate inference on task **On** for given training steps and drop rates[9]. 1000 states $\{s_1, \cdots, s_{1000}\}$ were sampled from the environment based on π. For each s_i, the KL-divergence between $\pi(\cdot|s_i)$ and $\hat{\pi}(\cdot|s_i)$ was used. The difference between π and $\hat{\pi}$ is computed as:

$$E_{s_i \sim \pi}[D_{KL}(\pi(\cdot|s_i)||\hat{\pi}(\cdot|s_i)) + D_{KL}(\hat{\pi}(\cdot|s_i)||\pi(\cdot|s_i))] \tag{10}$$

The heatmap depicts the log scale value of Eq. (10). The difference caused by approximate inference is only notable in the early training stages. The difference becomes trivial after 440 learning steps even with a large drop rate ($e^{-20} \approx 2 \times 10^{-9}$ when $x = 440, y = 0.9$). The observation supports the discussion in Sec. 1 and the key motivation of the paper: approximate inference with off-policy training can significantly boost the execution efficiency without sacrificing the benefits of DILP.

4.4 Rabbids Game

Experiments Setting To test OPDLRL's capability of (near) real-time inferences in complex domains, we developed a real-time game-play agent with OPDLRL to compete with other models and human players in the bumper-car mini-game of the Rabbids, which requires a policy response within the interval between frames. The state (vector) implies the position coordinates, velocity,

[9] Drop rate represents the percentage of rules ignored in the approximate inference, see Eq. (5).

and orientation of four cars, and the actions are forward, backward, left, right, as well as their combination. The state compiler translates the continuous state vector to high-level symbolic states, and the action compiler translates the high-level symbolic actions to a sub-goal, which a sub-policy tries to achieve. The high-level symbolic environment of Rabbids is a super-set of Car Avoiding with the following extra predicates: danger(X, Y) for whether a region (X, Y) can be reached by an opponent, reach(X, Y) for whether a region (X, Y) can be reached by agent, to(X) for a sub-goal of moving to the opponent X, avoid() for a sub-goal of avoiding the opponent, and tocenter() for a sub-goal of moving to the center. The action predicates in Car Avoiding are also considered as sub-goals to make high-level policy more flexible. danger(X, Y) and reach(X, Y) contains the velocity and orientation information, computed by a pre-trained network. Our agent was trained against behavior tree agents. Agent gets $+1$ reward when an opponent falls from the platform and get -1 reward when agent itself falls. Within one episode, each car has three chances to re-spawn. Agent wins the episode if and only if it stays alive until the end.

Results After the training, both OPDLRL agent and SAC agent achieved higher win rates against the behavior tree agent (OPDLRL:100.0%(124/124), SAC:97.1%(99/102)). To evaluate the generalization ability, we organized a series of competitions against human players, and OPDLRL agent kept 100.0%(21/21) win rate while SAC agent decreased to only 20.0%(4/20). We observed that most loss of SAC agent was caused by states which were unseen during training. For example, SAC agent may act a self-killing behavior when an opponent stays still, as staying still has never occurred during the training against a behavior tree agent. The higher win rates of OPDLRL agent show that our method successfully captures/induces *the key point* via DILP. Unfortunately, other DILP-based approaches failed this test as they cannot compute a practical response within the time limit.

5 Discussion

Mirror Descent Reinforcement learning with interpretable policy representation is challenging because of its highly structured nature of the policy space. Thus, the training of interpretable policies cannot be seen as an unconstrained policy optimization. Mirror descent is an efficient method to solve a constrained optimization problem and [34] has demonstrated how to learn a programmatic policy via mirror descent. SAC is known as a special form of mirror descent [36,23]. The use of SAC to optimize DILP in our paper can thus be interpreted from a perspective of mirror descent: i) learning an unconstrained policy via MLP-based Q-function (Eq. (6)); ii) projecting the unconstrained policy into constrained DILP-based policy space (Eq. (7)). From a theoretical perspective, this view may help understand the advantages of our method and provide insight to further improving Neuro-Symbolic methods with structured nature.

Multi-Agent Rabbids can also be seen as a multi-agent game. Generally, single-agent RL algorithms cannot converge to a robust policy which can rationally respond to various opponent's policy due to the non-stationary feature caused by the change of opponent's policy. Currently, the most popular and successful solution to multi-agent game is self-play [14,19,13]. Instead of using self-play, in this paper, we explored to learn a robust/well-performed single-agent policy for a multi-agent video game via a specified regularization, which restricts the policy within the interpretability structure. The results proved the effectiveness of our solution when dealing with the multi-agent environment.

6 Conclusion

In this paper, we proposed Off-Policy Differentiable Logic Reinforcement Learning (OPDLRL) framework to inherit the benefits of interpretability and generalization ability in DILP and also resolve its weakness of execution efficiency, stability, and scalability. OPDLRL has similar or better performance than other DILP-based methods but far more practical in terms of sample efficiency and execution efficiency, making it applicable to complex and (near) real-time domains. The key contributions include the use of approximate inference to significantly reduce the number of logic rules required for inferences and the well-designed off-policy training process to enable approximate inference. Various experiments, specifically playing real-time video games in Rabbids against human players, demonstrated its strength and practicability.

Acknowledgement

Their work is partially supported by NSFC under Grant (U19B2020 and 61772074) and National Key R&D Program of China under Grant (2017YFB0803300).

References

1. Barto, A.G., Mahadevan, S.: Recent advances in hierarchical reinforcement learning. Discrete event dynamic systems **13**(1-2), 41–77 (2003)
2. Boutilier, C., Reiter, R., Price, B.: Symbolic dynamic programming for first-order mdps. In: IJCAI. vol. 1, pp. 690–700 (2001)
3. Christodoulou, P.: Soft actor-critic for discrete action settings. arXiv preprint arXiv:1910.07207 (2019)
4. Colledanchise, M., Ögren, P.: Behavior trees in robotics and AI: an introduction. CoRR **abs/1709.00084** (2017)
5. Dietterich, T.G.: Hierarchical reinforcement learning with the maxq value function decomposition. Journal of artificial intelligence research **13**, 227–303 (2000)
6. Dong, H., Mao, J., Lin, T., Wang, C., Li, L., Zhou, D.: Neural logic machines. arXiv preprint arXiv:1904.11694 (2019)
7. Evans, R., Grefenstette, E.: Learning explanatory rules from noisy data. Journal of Artificial Intelligence Research **61**, 1–64 (2018)

8. Evans, R., Grefenstette, E.: Learning explanatory rules from noisy data (extended abstract). In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18. pp. 5598–5602. International Joint Conferences on Artificial Intelligence Organization (7 2018). https://doi.org/10.24963/ijcai.2018/792
9. Frankle, J., Carbin, M.: The lottery ticket hypothesis: Finding sparse, trainable neural networks. In: International Conference on Learning Representations (2019)
10. Haarnoja, T., Tang, H., Abbeel, P., Levine, S.: Reinforcement learning with deep energy-based policies. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 1352–1361. JMLR. org (2017)
11. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. arXiv preprint arXiv:1801.01290 (2018)
12. Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al.: Soft actor-critic algorithms and applications. arXiv preprint arXiv:1812.05905 (2018)
13. Heinrich, J., Lanctot, M., Silver, D.: Fictitious self-play in extensive-form games. In: Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37. p. 805–813. ICML'15, JMLR.org (2015)
14. Heinrich, J., Silver, D.: Deep reinforcement learning from self-play in imperfect-information games. CoRR **abs/1603.01121** (2016)
15. Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., Silver, D.: Rainbow: Combining improvements in deep reinforcement learning. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
16. Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., van Hasselt, H., Silver, D.: Distributed prioritized experience replay. In: International Conference on Learning Representations (2018)
17. Jiang, Z., Luo, S.: Neural logic reinforcement learning. In: International Conference on Machine Learning. pp. 3110–3119 (2019)
18. Kersting, K., Otterlo, M.V., De Raedt, L.: Bellman goes relational. In: Proceedings of the twenty-first international conference on Machine learning. p. 59 (2004)
19. Lanctot, M., Zambaldi, V., Gruslys, A., Lazaridou, A., Tuyls, K., Perolat, J., Silver, D., Graepel, T.: A unified game-theoretic approach to multiagent reinforcement learning. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30, pp. 4190–4203. Curran Associates, Inc. (2017)
20. Lin, J., Rao, Y., Lu, J., Zhou, J.: Runtime neural pruning. In: Advances in Neural Information Processing Systems. pp. 2181–2191 (2017)
21. Liu, Z., Sun, M., Zhou, T., Huang, G., Darrell, T.: Rethinking the value of network pruning. arXiv preprint arXiv:1810.05270 (2018)
22. Manhaeve, R., Dumancic, S., Kimmig, A., Demeester, T., De Raedt, L.: Deepproblog: Neural probabilistic logic programming. In: Advances in Neural Information Processing Systems. pp. 3749–3759 (2018)
23. Mei, J., Xiao, C., Huang, R., Schuurmans, D., Müller, M.: On principled entropy exploration in policy optimization. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19. pp. 3130–3136. International Joint Conferences on Artificial Intelligence Organization (7 2019). https://doi.org/10.24963/ijcai.2019/434

24. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. In: International conference on machine learning. pp. 1928–1937 (2016)
25. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
26. Muggleton, S.: Inductive logic programming. New generation computing **8**(4), 295–318 (1991)
27. Muggleton, S., et al.: Stochastic logic programs. Advances in inductive logic programming **32**, 254–264 (1996)
28. Nachum, O., Gu, S.S., Lee, H., Levine, S.: Data-efficient hierarchical reinforcement learning. In: Advances in Neural Information Processing Systems. pp. 3303–3313 (2018)
29. Payani, A., Fekri, F.: Inductive logic programming via differentiable deep neural logic networks. arXiv preprint arXiv:1906.03523 (2019)
30. Sanner, S., Boutilier, C.: Practical solution techniques for first-order mdps. Artificial Intelligence **173**(5-6), 748–788 (2009)
31. Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P.: Trust region policy optimization. In: International conference on machine learning. pp. 1889–1897 (2015)
32. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
33. Sutton, R.S., McAllester, D.A., Singh, S.P., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. In: Advances in neural information processing systems. pp. 1057–1063 (2000)
34. Verma, A., Le, H., Yue, Y., Chaudhuri, S.: Imitation-projected programmatic reinforcement learning. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 15752–15763. Curran Associates, Inc. (2019)
35. Vezhnevets, A.S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., Kavukcuoglu, K.: Feudal networks for hierarchical reinforcement learning. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 3540–3549. JMLR. org (2017)
36. Wang, Q., Li, Y., Xiong, J., Zhang, T.: Divergence-augmented policy optimization. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 6099–6110. Curran Associates, Inc. (2019)
37. Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., De Freitas, N.: Dueling network architectures for deep reinforcement learning. arXiv preprint arXiv:1511.06581 (2015)
38. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning **8**(3-4), 229–256 (1992)
39. Wulfmeier, M., Ondruska, P., Posner, I.: Maximum entropy deep inverse reinforcement learning. arXiv preprint arXiv:1507.04888 (2015)
40. Zhou, H., Lan, J., Liu, R., Yosinski, J.: Deconstructing lottery tickets: Zeros, signs, and the supermask. In: Advances in Neural Information Processing Systems. pp. 3592–3602 (2019)
41. Ziebart, B.D.: Modeling purposeful adaptive behavior with the principle of maximum causal entropy (Jul 2018). https://doi.org/10.1184/R1/6720692.v1
42. Ziebart, B.D., Maas, A.L., Bagnell, J.A., Dey, A.K.: Maximum entropy inverse reinforcement learning. In: Aaai. vol. 8, pp. 1433–1438. Chicago, IL, USA (2008)