Reinforcement Learning from Hierarchical Critics

Zehong Cao, *Member, IEEE*, and Chin-Teng Lin, *Fellow, IEEE*

***Abstract*—In this study, we investigate the use of global information to speed up the learning process and increase the cumulative rewards of reinforcement learning (RL) in competition tasks. Within the framework of actor-critic RL, we introduce multiple cooperative critics from two levels of a hierarchy and propose an RL from hierarchical critics (RLHC) algorithm. In our approach, each agent receives value information from local and global critics regarding a competition task and accesses multiple cooperative critics in a top-down hierarchy. Thus, each agent not only receives low-level details but also considers coordination from higher levels, thereby obtaining global information to improve the training performance. Then, we test the proposed RLHC algorithm against a benchmark algorithm, i.e., proximal policy optimisation (PPO), under four experimental scenarios consisting of tennis, soccer, banana collection, and crawler competitions within the Unity environment. The results show that RLHC outperforms the benchmark on these four competitive tasks.**

***Index Terms*—Reinforcement Learning; Hierarchy; Critics; Competition**

I. Introduction

Many agent training studies concern reinforcement learning (RL) techniques, which provide learning policies to achieve cooperative or competitive tasks by maximising rewards through interactions with the environment [1]. At each training step, the agent perceives the state of the environment and takes an action that leads the environment to transition into a new state. In a competitive game with multiple players, such as a zero-sum game between two agents, the minimax principle is applied, in which each player tries to maximise its benefits under the worst-case assumption that the opponent will always endeavour to minimise that benefit. For example, the minimax-Q algorithm [2] employs the minimax principle to compute strategies and values for stage games and a temporal-difference rule similar to Q-learning to propagate the values across state transitions.

To achieve the goal of maximising rewards, an agent always faces the "exploration-exploitation dilemma" in RL when interacting with a complex environment, particularly in tasks with multi-agent scenarios. The popular hierarchical RL framework decomposes a task into multiple levels with hierarchies to exploit temporally extended actions to make decisions from a higher-dimensional perspective to alleviate the sparse reward problem. To reach a target with fast exploration, a general hierarchical RL sets the high-level strategy to select sub-goals and the low-level strategy to perform primitive operations to achieve these sub-goals. For complex competition environments with multiple agents, such as StarCraft II micromanagement tasks, [3] proposed a joint value-based method based on Q-learning, QMIX, to coordinate between the centralised and decentralised off-policies for discrete actions. Furthermore, [4] presented an adaptation of actor-critic methods that combines value-based methods in the critic and policy gradient methods in the actor. Following the above, [5] recently proposed a new actor-critic method called counterfactual multi-agent (COMA) on-policy gradients that uses a centralised critic to estimate the Q value and decentralised actors to optimise the agents' policies for discrete actions without consideration of the continuous action spaces.

Furthermore, according to a recent review of state-of-the-art RL studies with a hierarchical actor-critic (HAC), most existing methods provide an approach for breaking down tasks that is divided into two strategies. The first strategy is "hindsight experience replay", which helps agents learn goal-based policies more quickly when sparse reward functions are used. References [6], [7] used hindsight action transitions to help agents understand the sub-goal state achieved in hindsight instead of the original sub-goal state and generate valuable hindsight goals that can be accomplished by an agent in the short term. The second strategy is feudal RL [8], [9], which separates knowledge into meta-policies and inter-option policies and generates explicit sub-goals in a latent space to encourage exploration. Specifically, it employs the manager and worker modules for hierarchical RL. The manager sets abstract goals, which are conveyed to and enacted by the worker, who generates primitive actions at each environmental tick. The feudal network structure has been extended to cooperative RL [10], in which the manager learns to communicate sub-goals to multiple workers. Indeed, the ability to extract sub-goals from the manager allows the feudal network to dramatically outperform a potent baseline agent on tasks. In addition, this hierarchical policy structure in RL has been developed for real-world scenarios, such as off-policy correction (HIRO) [11] and mutual information maximisation (adInfoHRL) [12], to improve the sample efficiency in robot movement control.

However, the above two HAC strategies focus only on fast exploration, as agents can learn short policies at each level of the hierarchy, which indicates a policy that allows the agent to learn tasks more quickly by breaking the problem

Z. Cao was with the School of Computer Science, University of Technology Sydney, Sydney, Australia. He is now with the School of ICT, University of Tasmania, Hobart, Australia (e-mail: zehong.cao@utas.edu.au).

C.T. Lin is with the Australian Artificial Intelligence Institute (AAII), School of Computer Science, University of Technology Sydney, Sydney, Australia (e-mail: chin-teng.lin@uts.edu.au).

This research was sponsored in part by the Office of Naval Research Global, US, and was accomplished under Cooperative Agreement Number ONRG - NICOP - N62909-19-1-2058.

down into short sequences of actions, and neither considers the improvement of "exploitation performance". Hence, the previous hierarchical RL studies have ignored the critical fact that giving an agent access to multiple cooperative critics might speed up the learning process and increase the rewards in competitive tasks. In particular, it is frequently the case that high-level agents agree to be assigned different observations that work in combination with low-level agents to benefit hierarchical cooperation. Inspired by the feudal network structure with the manager and worker modules, this new structure may provide low-level agent access to multiple cooperative critics by the measurement of different-level observation spaces along with the maximum value function.

Thus, in this study, we introduce multiple cooperative critics from two levels of the hierarchy and propose an RL from hierarchical critics (RLHC) algorithm. If we can obtain a good value function that allows selecting the action with the highest value in this state, then naturally this strategy is better. The main contributions of our proposed approach are as follows:

- An agent receives information from both local and global critics regarding a competitive task.
- The agent receives not only low-level details but also global information to consider coordination from higher levels to increase the exploitation performance.
- We define multiple cooperative critics in a top-to-bottom hierarchy, called RLHC. We assume that RLHC is a potential generalised RL and is thus more suitable for speeding up training and improving exploitation learning for agents. These benefits could potentially be obtained when using any type of hierarchical RL algorithm.

The remainder of this paper is organised as follows. In Section 2, we introduce the RL background for developing the multiple cooperative critic framework in the agent-competition domain. Section 3 describes the baseline and proposes the RLHC algorithm. Section 4 presents four experimental designs based on competitive tennis, soccer, banana collection, and crawler tasks with observation settings within the Unity environment. Section 5 reports the training performance results of the benchmark algorithm and the proposed RLHC algorithm. Finally, we summarise the paper in Section 6.

II. Preliminaries

A. Revisiting RL

In a standard RL framework [13], an agent interacts with the external environment over a number of time steps. Here, s is the set of all possible states, and a is the set of all possible actions. At each time step t, the agent in state s_t perceives the observation information O_t from the environment, takes an action a_t, and receives feedback from the reward source R_t. Then, the agent transitions to a new state s_{t+1}, and the reward R_{t+1} associated with the transition (s_t, a_t, s_{t+1}) is determined. The agent can choose an action from the last state visited. The goal of an RL agent is to collect the maximum possible reward with minimal delay.

Next, we revisit the primary components of the learning process: the Markov decision process (MDP) and the policy gradient.

In the MDP, a state s_t is a Markov state if and only if

$$\mathbb{P}[s_{t+1}|s_t] = \mathbb{P}[s_{t+1}|s_1, ..., s_t]. \tag{1}$$

The future state is independent and unrelated to the past states. The state transition matrix P is defined to present the transition probabilities from all states s to all subsequent states s':

$$P_{ss'} = \mathbb{P}[s_{t+1} = s'|s_t = s]. \tag{2}$$

A Markov reward process is a tuple $< s, a, P, R, \gamma >$, where s is a finite set of states, a is a finite set of actions, and γ is a discount factor, $\gamma \in [0, 1]$.

- P is a state transition probability matrix from Equation (2):

$$P^a_{ss'} = \mathbb{P}[s_{t+1} = s'|s_t = s, a_t = a]. \tag{3}$$

- In addition, r is a reward function that represents the expected reward after the transition from P:

$$r^a_s = \mathbb{E}[r_{t+1}|s_t = s, a_t = a]. \tag{4}$$

The return R_t, defined as the sum of future discounted rewards, is

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}. \tag{5}$$

To estimate "how good" it is to be in a given state, the state value function of the reward $V_\pi(s)$ is defined as the expected return starting with state s under policy π:

$$V_\pi(s) = \mathbb{E}[R_t|s_t = s, \pi], \tag{6}$$

where policy π is as follows:

$$\pi(a|s) = \mathbb{P}[a_t = a|s_t = s].$$

It is useful to define the action value of the reward function $Q_\pi(s, a)$:

$$Q_\pi(s, a) = \mathbb{E}[R_t|s_t = s, a_t = a, \pi]. \tag{7}$$

Following the introduction of the value function, we can generate a gradient-ascent-based RL, called the policy gradient. As a gradient ascent strategy, this approach models and optimises the policy directly. The policy is usually modelled by a parameterised function with respect to θ, with $\pi_\theta(s, a)$. The value of the reward function depends on this policy and various other algorithms, such as REINFORCE (Monte Carlo policy gradient) [14], the deep deterministic policy gradient (DDPG) [15], and the asynchronous advantage actor-critic (A3C) [16]. Proximal policy optimisation (PPO) [17] can be applied to optimise θ to acquire the greatest reward.

The fundamental reward function is defined as follows:

$$J(\theta) = \mathbb{E}_{\pi\theta}[\pi_\theta(s, a)Q^{\pi_\theta}(s, a)], \tag{8}$$

and then, the gradient is computed:

$$\bigtriangledown_\theta J(\theta) = \mathbb{E}_{\pi\theta}[\bigtriangledown_\theta log\pi_\theta(s, a)Q^{\pi_\theta}(s, a)]. \tag{9}$$

B. Actor-critic

The actor-critic strategy aims to take advantage of the best characteristics from both value-based and policy-based approaches while eliminating all their drawbacks and underlies recent modern RL methods from A3C to PPO. To understand the learning strategies, the value function can facilitate policy updates, such as by reducing gradient changes in the original strategy gradient, which is what actor-critic methods do. Specifically, actor-critic methods consist of two models that can optionally share parameters: (a) a critic updates the value function parameters w, which could be an action-value function $Q_w(s,a)$ or a state value function $V_w(s)$; (b) the actor updates the policy parameters θ for $\pi_\theta(s,a)$ in the direction suggested by the critic.

1) Asynchronous Advantage Actor-critic (A3C) : The A3C structure [16] can be applied to a variety of continuous motor control tasks and learn general game exploration strategies purely from observations. A3C maintains a policy ($\pi_\theta(s_t,a_t)$) and an estimate of the value function ($V(s_t;\theta_w)$). Thread-specific parameters are synchronised with the global parameters: $\theta' = \theta$ and $w' = w$. This variant of actor criticism can operate in the forward view and uses the same mix of n-step returns to update both the policy and the value function.

The update reward function can be written as follows:

$$\bigtriangledown_{\theta'} J(\theta') = \bigtriangledown_{\theta'} log\pi_{\theta'}(s_t,a_t)\hat{A}(s_t,a_t;\theta,\theta_w), \tag{10}$$

where $\hat{A}$ is an estimate of the advantage function given by

$$\hat{A}(s_t,a_t;\theta,\theta_w) = \sum_{i=0}^{k-1} \gamma^i r^{t+i} + \gamma^k V(s_{t+k};\theta) - V(s_t;\theta_w), \tag{11}$$

in which k varies from state to state and has an upper bound t_{max}.

The parameters θ (of the policy) and θ_w (of the value function) are shared even when they are shown to be separate for generality. For example, a convolutional neural network has one softmax output for the policy $\pi_\theta(s_t,a_t)$ and one linear output for the value function $V(s_t;\theta_w)$, and all its non-output layers are shared.

2) Proximal Policy Optimisation (PPO) : PPO [17] represents a new family of policy gradient methods for RL that alternate between sampling data through interactions with the environment and optimising a surrogate objective function using stochastic gradient ascent. PPO imposes a constraint by forcing $r(\theta')$ to remain within a small interval of approximately 1, that is, $[1-\varepsilon, 1+\varepsilon]$, where ε is a hyper-parameter. The function $clip(r(\theta'), 1-\varepsilon, 1+\varepsilon)$ clips the ratio to within $[1-\varepsilon, 1+\varepsilon]$.

The objective function measures the total advantage over the state visitation distribution and actions:

$$J(\theta') = \mathbb{E}[r(\theta')\hat{A}^\theta(s,a)], \tag{12}$$

where $r(\theta') = \pi_{\theta'}(s,a)/\pi_\theta(s,a)$ represents the probability ratio between the new and old policies.

To approximately maximise each iteration, the "surrogate" objective function is as follows:

$$J(\theta') = \mathbb{E}[min(r(\theta'))\hat{A}^\theta(s,a), clip(r(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}^\theta(s,a)]. \tag{13}$$

III. Method

To propagate the critics in the hierarchies, we propose RLHC, which considers multiple cooperative critics in two levels of the hierarchy. RLHC aims to speed up the learning process and increase the cumulative rewards, which it achieves by having each agent receive information from both local and global critics. The novelty of this study is that it supports the concept that considering information from multiple critics at different levels is beneficial for training in a hierarchical RL framework. The assumption is that a higher-level critic will be beneficial for an agent who was previously able to use only the critic in its surrounding layer. Thus, we address the modified advantage achieved by the maximum function in a union set based on the baseline, i.e., the benchmark PPO algorithm.

A. Baseline: the Benchmark PPO

PPO performs comparably to or better than other state-of-the-art RL methods and became the benchmark RL algorithm at OpenAI [1] and Unity [2] due to its ease of use and good performance. Here, we use PPO both as a baseline to validate the experiments and as a starting point to develop the novel RLHC algorithm.

B. Learning in Multi-critics

To apply PPO to the problem of agents with variable attention to more than one critic, we consider the argument for resolving the multiple-critic learning problem. For each critic i, the corresponding advantage function is $A^{\theta_i}(s_i,a)$, generated from the state value function $V^i(s_i,\theta)$, depending on the different scale observations O (expressed as the state s) and the network parameter θ. Consistent with existing work, the advantage function is extended from the value function and measures the value of the agent's actions.

For multiple critics (e.g., two critics, $i=2$), we work with the argument of the minimum objective function to find the minimum advantage of choosing a specific action instead of following the current policy. The argument of the minimum objective function can be written as follows:

$$argmin(\bigtriangledown_J(\theta')) = argmin(\mathbb{E}[\bigtriangledown_r(\theta')\hat{A}^\theta(s,a)]). \tag{14}$$

To achieve a minimised $\hat{A}^\theta(s,a)$, we need to maximise the current state value function $V(s;\theta)$ extracted from Equation (11), which can be written as

$$min[\hat{A}(s,a)] \rightarrow max[V(s;\theta)]. \tag{15}$$

In other words, we seek the set $\hat{V_i}^\theta$ of the given argument of objective function $J(\theta')$ for which the value of the given expression attains its maximum value. Because the maximum

[1] https://openai.com/blog/openai-baselines-ppo

[2] https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Training-PPO.md

$\hat{V}(s,\theta)$ indicates that action a is a better choice than the current policy $\pi(\theta)$, we measure the advantage achieved by collecting individual $\hat{V}^i(s,\theta)$ and choosing the maximum $\hat{V}(s,\theta)$. The corresponding updated value function can be written as follows:

$$\hat{V}(s,\theta) = max \bigcup_{i=2}^{m} \hat{V}^i(s,\theta), \tag{16}$$

where m is the total number of critics.

If we consider the n time-step intervals of multiple critics, then m in Equation (16) can be replaced with h_t, where $h_t = h_{t+kT}$, in which $h_t = m$, $k = 2, 3, 4, ...$, and T is a time period with n time steps; otherwise, $h_t = 2$.

C. RLHC

Because the environmental state is too complicated, we assume that the agent can obtain only the partial environmental state, which we term partial observability. By introducing a manager that can observe more environmental states, we can develop an RLHC framework to speed up the learning process and achieve more cumulative rewards. In terms of propagating the critics in the hierarchies, we are the first to develop an RL strategy from hierarchical critics that allows a worker agent i to receive information from multiple critics computed both locally and globally. The manager is responsible for collecting the broader observations and estimating the corresponding global critic, which it sends to the worker agent. To clarify our proposed algorithm, we present the pseudo-code of our proposed RLHC below.

Here, we apply the RLHC algorithm in PPO. Successful RLHC model training requires tuning of the trained hyper-parameters, which enables obtaining an output of the training process that contains the optimised policy. This investigation allows criticism from the manager to improve the training performance.

Furthermore, in Fig. 1, we present a simplified version of the RLHC algorithm constructed using a two-level hierarchy for one worker agent with a manager. The local and global critics are implemented by the maximum function illustrated in the "Learning in Multi-critics" section. For the modified state value function we propose, the manager and worker share the actors, but they provide different critics from the two layers (which we consider hierarchical), which correspond to the arrows and the maximum function in Fig. 1. The manager receives the shared action space from the worker but provides only high-level criticism to the worker. This strategy allows us not only to estimate the value of multiple critics from different levels but also to use weighted approaches to fuse critics from different layers or to optimise the temporal scaling of critics in separate layers. For simplicity, the experiments in the following section generally involve two-level hierarchies, such as a multi-agent hierarchy with up to 2 managers and 6 worker agents for competition.

Algorithm 1 RLHC

1: Define the observation environment for each worker agent $i \rightarrow O_w^i$ and manager $\rightarrow O_m$
2: Initialise $\rightarrow$ the state of each worker agent $s_0^{w_i}$, the state of the manager s_0^m and the policy parameter θ_0
3: Initialise $\rightarrow$ the critic networks of manager $\hat{A}_0^m$ and each worker $\hat{A}_0^{w_i}$ and the actor network for each worker $\pi_0^{w_i}$ to determine action a
4: **for** Iteration $= 1, 2, ...$ **do**
5: **for** Actor $= 1, 2, ..., i$ **do**
6: Run policy π_θ in the environment for $T \in t$ time steps
7: Use θ' to interact with the environment to collect s_t, a_t and compute the advantage function $\hat{A}_t^\theta$
8: Minimise the gradient of the objective function $\bigtriangledown_J(\theta')$
9: $\rightarrow$ measure the probability ratio $r(\theta')$ between new and old policies
10: $\rightarrow$ find the maximum current value function $\hat{V}_t^\theta$ to achieve the minimum advantage function $\hat{A}_t^\theta$ selected from the advantage estimate of worker agent i or the manager

$$argmin \bigtriangledown_J(\theta') =$$

$$\mathbb{E}[min[(\bigtriangledown_r^w(\theta')\hat{A}_t^{\ \theta}(s_t^{w_i}, a_t), \bigtriangledown_r^m(\theta')\hat{A}_t^{\ \theta}(s_t^m, a_t)]$$

$$\rightarrow max[V(s_t^{w_i};\theta), V(s_t^m;\theta)]$$

11: Choose the maximum value:

$$\hat{V}_t^\theta = max \bigcup_{i=1} (V(s_t^{w_i};\theta), V(s_t^m;\theta))$$

12: Use the maximum value $\hat{V}_t^\theta$ to calculate the advantage estimates $\hat{A}_1^\theta$,..., $\hat{A}_T^\theta$
13: **end for**
14: Optimise the "surrogate" objective function from PPO
15: Update $\theta' \rightarrow \theta$
16: **end for**

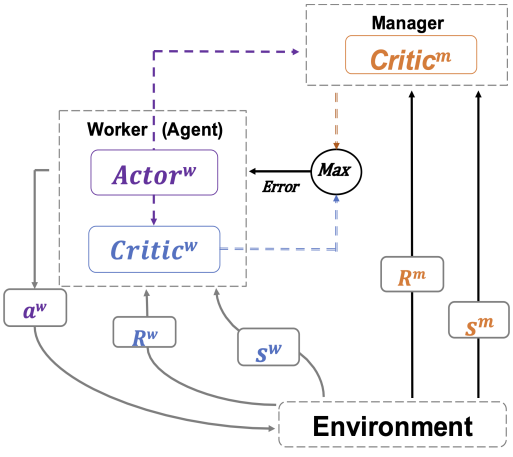

Figure 1. The RLHC algorithm

IV. Experiment

We applied our proposed RLHC algorithm to four scenarios in which up to 6 agents compete. We empirically show the success of our RLHC compared with the benchmark PPO method in competitive scenarios such as tennis, soccer, banana collection, and crawling.

To be consistent with RLHC as addressed in Section III, in the application stage, we also included two types of observations (agent/worker O_w and manager O_m): the worker can observe only the local environmental state s_w, while the manager can observe the global environmental state s_m. We have released codes for both the model and the environments on GitHub [3] for replication purposes.

A. Unity Platform for RL

Because many existing platforms (e.g., OpenAI Gym) lack the ability to flexibly configure a simulation for multiple agents, the simulation environment becomes a black box from the perspective of the learning system. The Unity platform, a new open-source toolkit, has been developed for creating and interacting with simulation environments. Specifically, the Unity machine learning agent toolkit (ML-Agents Toolkit) [18] is an open-source Unity plug-in that enables games and simulations to serve as environments for training multiple intelligent agents. The toolkit supports dynamic multi-agent interaction, and agents can be trained using RL through a straightforward Python API.

B. Scenario 1: Tennis Competition

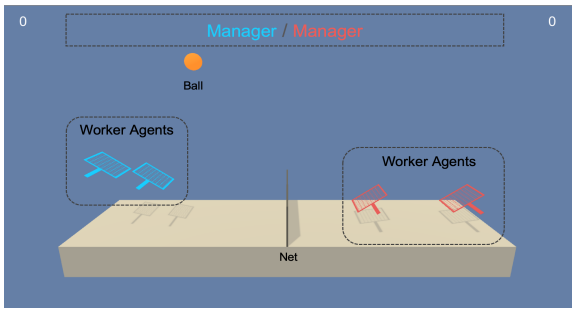

Figure 2. 2 vs. 2 tennis competition in Unity

In this game, agents control rackets to bounce a ball over a net. We constructed a new training environment in Unity under 2-2 worker and 1-1 manager settings (a doubles-tennis scenario), as shown in Fig. 2. In Table I (Part 1), the goal, agent reward function, and behaviour parameters, including the action and observation spaces, are set up for the tennis agents. Note that we set extended local individual observations, where the low-level agents (racket workers) can also access the distance and velocity difference between teammates to avoid duplicate policies and actions. The manager observations include additional variables, such as the distance between the ball and the racket and information gained from the worker agent's observations. In other words, we set two sub-types of observations with and without teammate communication information, which represent 1) agent/worker observation without teammate communication O_w^1 and manager observation without teammate communication O_m^1 and 2) agent/worker observation with teammate communication O_w^2 and manager observation without teammate communication O_m^2, as shown in Table I (Part 1).

[3] https://github.com/czh513/RL-Hierarchical-Critics

C. Scenario 2: Soccer Competition

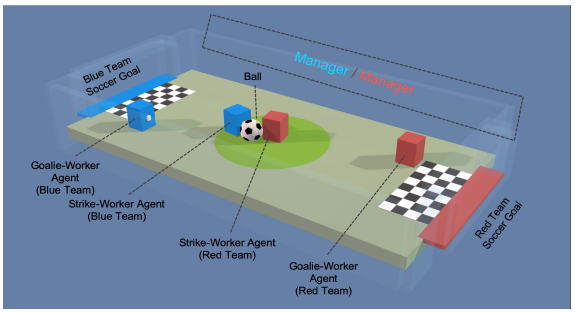

Figure 3. 2 vs. 2 soccer competition in Unity

This scenario involves 4 agents competing in a simplified soccer game in Unity. Fig. 3 shows the environment, where 4 agents compete in a 2 vs. 2 soccer game. This game has two types of players, offensive and defensive, which need to be controlled differently. We use "multi-brain training" in Unity because each team contains one striker agent and one goalie agent, and each is trained using separate reward functions; thus, each type has its own observation and action spaces. As presented in Table I (Part 2), the goals, agent reward function, and behaviour parameters, including the action and observation spaces, are set up for the soccer agents.

D. Scenario 3: Banana Collection Competition

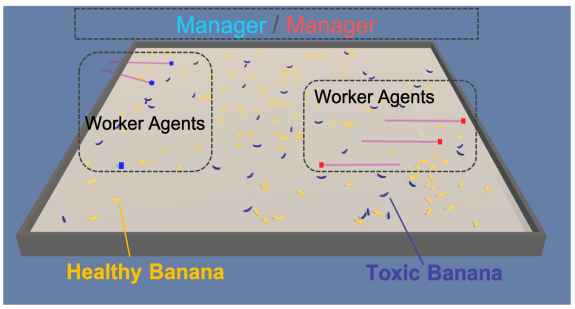

Figure 4. 3 vs. 3 banana collection competition in Unity

In this competition, 6 agents compete in a banana collection game in Unity. Fig. 4 shows the environment, where 6 agents compete in a 3 vs. 3 banana collection game. As presented in Table I (Part 3), the goals, agent reward function, and behaviour parameters, including the action and observation spaces, are set up for the banana collection agents.

E. Scenario 4: Crawler Competition

This competition involves 2 agents competing in a simplified fighting game in Unity. Fig. 5 shows the environment, where 2 agents compete in a 1 vs. 1 crawler game. As presented in Table I (Part 4), the goals, agent reward function, and behaviour parameters, including the action and observation spaces, are set up for the crawler agents.

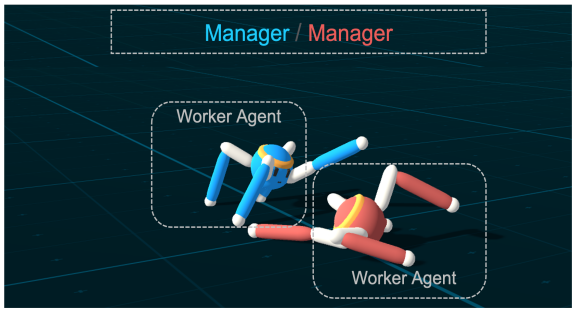

Figure 5. 1 vs. 1 crawler competition in Unity

F. Training Settings and Metrics

1) Training Settings: The hyper-parameters for the RL used for training are specified in Table II, which provides the initialisation settings that we used to interact with four different competition scenarios. Specifically, the batch size and buffer size represent the number of experiences that occur during each gradient descent iteration and the number of experiences to collect before updating the policy model, respectively. Beta controls the strength of entropy regularisation, and epsilon influences how rapidly the policy can evolve during training. Gamma and lambda indicate the reward discount rate for the generalised advantage estimator and the regularisation parameter, respectively. A random seed is used to ensure that the results are reproducible.

2) Training Metrics: We saved some statistics during the learning session and viewed them using a TensorFlow utility named TensorBoard. Here, we measure four metrics to assess the training performance. Specifically, *Cumulative Reward* indicates the mean cumulative episode reward accrued by all agents interacting with the environment. *Reward Statistics* compare the performance of each RL algorithm (PPO vs. RLHC) under different conditions. We first collected the cumulative reward values of the final 10K time steps, as this period is generally considered to yield the best cumulative reward with convergence. Then, we used the independent T-test for statistical analysis and plotted the distribution of the kernel density estimation (KDE) to estimate the probability density function of the final cumulative rewards and set $p < 0.01$ as the minimum significant level.

V. Results

We provide here the training performances of the RLHC algorithm and the baseline benchmark algorithm (PPO). PPO uses an independent local critic for each agent and does not share information, thus rendering the environment non-stationary from a single-agent perspective. However, our RLHC includes a semi-centralised critic, which it obtains by hierarchically assigning a critic to estimate the updated value function; this can be beneficial for independent learners, which are known to struggle in hierarchically cooperative settings.

The following findings show that RLHC is both more efficient and more general than PPO; consequently, we choose four example scenarios for use with up to 6-player tennis, soccer, banana collection and crawler competitions. We set the smoothing parameter $= 0.8$ in TensorBoard, which is used as the exponential moving average to reduce the variations in the values for better presentation.

A. Tennis Competition

For the tennis competition (the doubles scenario), we use both the type 1 observation space (without teammate communication) and type 2 observation space (with teammate communication) for training purposes. As shown in Table I (Part 1), we set 2 observation space categories, worker and manager, consisting of the type 1 observation space and type 2 observation space, which are denoted as $Ob1$ and $Ob2$, respectively. We explore whether the type 2 observation space, which adds the extended observations, is beneficial in achieving a higher reward. We also compare the performance metrics of RLHC and the benchmark PPO in terms of both types of observation space.

As shown in Fig. 6-A, considering the type 1 observation space, RLHC achieves a higher cumulative reward (mean $\pm$ standard deviation) with short training steps than achieved by PPO. Furthermore, as shown in Fig. 6-B, for the type 2 observation space, which adds the extended observations, the cumulative reward (mean $\pm$ standard deviation) achieved by PPO further increases compared with RLHC without extended observations, indicating that the extended observations that consider teammate relationships are significant in the training process. Additionally, we include the extended observations in RLHC and PPO to compare their training performances. The metrics and cumulative reward show that our RLHC performs better than PPO, and we find that the cumulative rewards of PPO and RLHC in the final training period have significant differences ($p < 0.001$) in both types of observation space, indicating that our RLHC achieves a significant improvement in the training process of the tennis scenario.

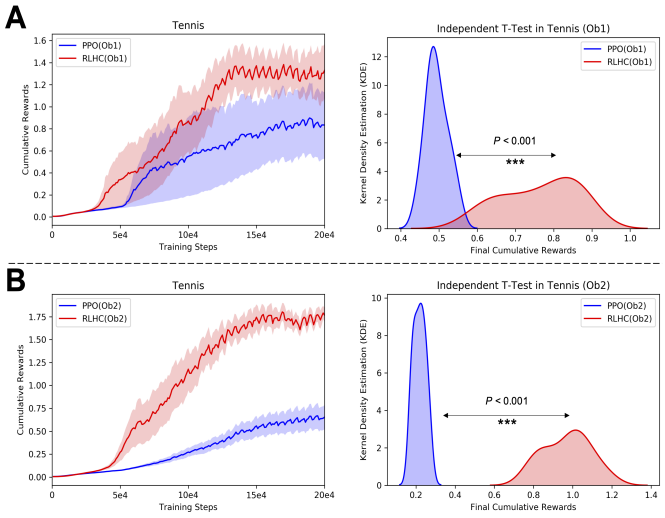

Figure 6. The training metrics for the tennis competition

B. Soccer Competition

For the soccer competition, we set the observation spaces for the worker and the manager to assess a different view, as shown in Table I (Part 2). During the training stage, we

Part 1: Settings of the Tennis Competition Scenario		
Setting	Description	
Objective	Agents cannot miss the ball or let the ball fall outside the court area during the event when striking the ball over the net into the opponents' court.	
Reward	+0.1 when the ball is hit over the net. -0.1 when agents miss the ball or the ball falls outside the tennis court.	
Action Space	Movement forward or away from the net and jumping (3 variables).	
Observation Space		Description
Type 1: Without teammate communication	Agent/Worker Observation Space O_w^1	Position and velocity information of the ball and racket (8 variables).
	Manager Observation Space O_m^1	Position and velocity information of the ball and racket and the distance between the ball and racket (10 variables).
Type 2: With teammate communication	Agent/Worker Observation Space O_w^2	Position and velocity information of the ball and racket and the distance and velocity difference between teammates (12 variables).
	Manager Observation Space O_m^2	Position and velocity information of the ball and racket, the distance between the ball and racket, and the distance and velocity difference between teammates (14 variables).
Part 2: Settings of the Soccer Competition Scenario		
Setting	Description	
Objective	Striker agents need to calculate a method to kick the ball into the opponent's goal. Goalie agents need to learn to defend against the opponent and to avoid the ball being kicked into their own goal.	
Reward	Striker: +1 when the ball enters the opponent's goal, -0.1 when the ball enters the team's own goal. Goalie: -1 when the ball enters the team's own goal, +0.1 when the ball enters the opponent's goal.	
Action Space	Striker: Forward, backward, rotation, and sideways movement (6 variables). Goalie: Forward, backward, and sideways movement (4 variables).	
Agent/Worker Observation Space O_w	Seven types of object detection, with distance information covering a 180 degree view (112 variables).	
Manager Observation Space O_m	Eight types of object detection, with distance information covering a 270 degree view (200 variables).	
Part 3: Settings of the Banana Collection Competition Scenario		
Setting	Description	
Objective	Agents must learn to collect as many healthy bananas as possible while avoiding toxic bananas.	
Reward	+1 when an agent collects a yellow healthy banana. -1 when an agent collects a purple toxic banana.	
Action Space	4 branches of action – movement branch: forward or backward; side-motion branch: left or right; rotation branch: rotate left or rotate right; laser branch: emit a laser (42 variables).	
Agent/Worker Observation Space O_w	Velocity of agents and the ray-based angle information of the objects in front of each agent: 7 raycast angles with 7 measurements for each angle (53 variables).	
Manager Observation Space O_m	Velocity and distance of agents, with the ray-based angle information of the objects in front of the agents: 7 raycast angles with 8 measurements for each angle (60 variables).	
Part 4: Settings of the Crawler Competition Scenario		
Setting	Description	
Objective	Agent must learn to maintain their body balance and not touch the ground and to fight against the opponent to make the challenger lose their balance.	
Reward	+1 if opponent's body touches the ground. -1 if agent's body touches the ground. +0.03 times the body velocity towards the opponent's direction. +0.01 times the body direction alignment with the opponent's direction.	
Action Space	Rotation of joints (20 variables).	
Agent/Worker Observation Space O_w	Position, rotation, velocity, and angular velocity of each limb, plus the acceleration and angular acceleration of the body (117 variables).	
Manager Observation Space O_m	Position, rotation, velocity, distance, and angular velocity of each limb, plus the acceleration and angular acceleration of the body (119 variables).	

TABLE I: Settings of the four competition scenarios

	Tennis	**Soccer**	**Collection**	**Crawler**		**Tennis**	**Soccer**	**Collection**	**Crawler**
Parameter	Value	Value	Value	Value	**Parameter**	Value	Value	Value	Value
batch size	1024	128	512	128	beta	0.005	0.01	0.01	0.01
buffer size	10240	2000	6000	5000	epsilon	0.2	0.2	0.2	0.2
gamma	0.99	0.99	0.99	0.99	hidden units	128	256	256	256
lambda	0.95	0.95	0.95	0.95	learning rate	0.0003	0.001	0.0005	0.001
max steps	200 K	500 K	200 K	200 K	memory size	256	256	256	256
random seed	5	5	5	5	num. epochs	3	3	3	3
num. layers	2	2	2	2	time horizon	64	128	128	128
sequence len.	64	64	64	64	summary freq.	1000	2000	2000	2000

TABLE II: Training parameter settings

trained two brains: one brain with a negative reward for the ball entering their goal and another brain with a positive reward for the ball entering the opponent's goal. Because the mean reward is the inverse between the striker and goalie and alternates during training, we demonstrate only the training metrics for the striker agent, as shown in Fig. 7. The corresponding training metrics for the goalie agent are simply the opposite of those for the striker agent.

In terms of the striker's performance, Fig. 7 shows that the cumulative reward (mean $\pm$ standard deviation) of PPO increased around the starting points and then decreased after 500K training steps, suggesting that this trial does not have a reliable learning process. However, our RLHC can achieve a positive result with higher cumulative rewards than those of PPO. Moreover, the cumulative rewards of PPO and RLHC in the final training period show significant differences ($p < 0.001$), suggesting that our RLHC achieves a significant improvement in the training process of the soccer scenario.

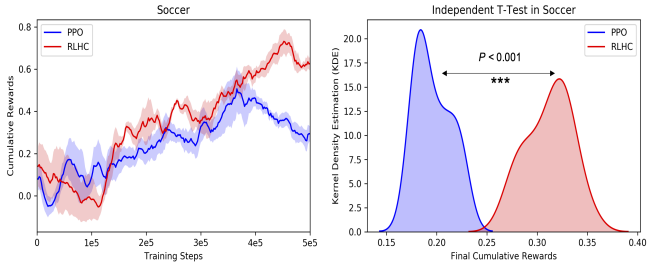

Figure 7. The striker's training metrics for the soccer competition

C. Banana Collection Competition

For the banana collection competition, we set the observation spaces for the worker and the manager to obtain a different perspective, as shown in Table I (Part 3), i.e., a multi-agent environment where agents compete to collect bananas. We expect that the two teams' agents can learn to collect as many healthy yellow bananas as possible while avoiding toxic purple bananas.

In terms of each team's performance, Fig. 8 shows that the cumulative rewards (mean $\pm$ standard deviation) of PPO and the proposed RLHC stably increase over the whole 200K training steps. However, our RLHC can achieve higher cumulative rewards than those of PPO, and the cumulative rewards of PPO and RLHC in the final training period show significant differences ($p < 0.001$), suggesting that our RLHC achieves a significant improvement in the training process of the banana collection scenario.

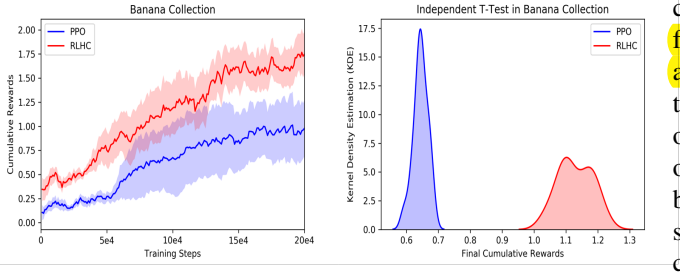

Figure 8. The training metrics for the banana collection competition

D. Crawler Competition

For the crawler competition, we set the observation spaces for the worker and the manager to obtain additional insight, as shown in Table I (Part 4). Each agent is created with 4 arms and 4 forearms and requires body movement to fight against an opponent without falling.

Fig. 9 shows the high cumulative reward (mean $\pm$ standard deviation) of PPO around the starting points relative to that of the proposed RLHC. After 200K training steps, our RLHC can achieve higher cumulative rewards and a rapidly rising curve compared with those of PPO. Moreover, the cumulative rewards of PPO and RLHC in the final training period show significant differences ($p < 0.001$), suggesting that our RLHC achieves a significant improvement in the training process of the crawler scenario.

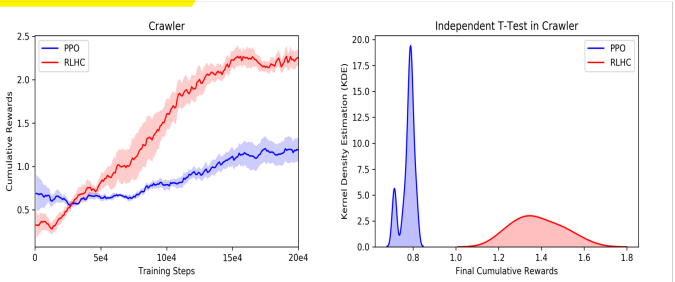

Figure 9. The training metrics for the crawler competition

VI. Conclusions

In this study, we developed an RLHC algorithm to consider global information to speed up the learning process and increase the cumulative rewards. In RLHC, the agent is allowed to receive information from both local and global critics in competitive tasks. We tested the proposed RLHC on four tasks, i.e., a 2-player crawler competition, 4-player tennis and soccer, and a 6-player banana collection competition, in the Unity environment by comparing its results with those of the benchmark PPO algorithm. The results showed that our proposed RLHC outperforms the non-hierarchical critic baseline PPO on agent-competition tasks. The novelty of this study is that it offers a proof of concept indicating that the consideration of multiple critics from different levels can be beneficial for training in a hierarchical RL framework. We selected a simple scenario as evidence, and the preliminary outcomes showed that improved performance can be achieved by considering the criticism of higher-level critics.

REFERENCES

[1] L. Busoniu, R. Babuška, and B. De Schutter, "Multi-agent reinforcement learning: An overview," *Innovations in multi-agent systems and applications-1*, vol. 310, pp. 183–221, 2010.

[2] M. L. Littman, "Value-function reinforcement learning in markov games," *Cognitive Systems Research*, vol. 2, no. 1, pp. 55–66, 2001.

[3] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Qmix: monotonic value function factorisation for deep multi-agent reinforcement learning," *arXiv preprint arXiv:1803.11485*, 2018.

[4] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in Neural Information Processing Systems*, pp. 6379–6390, 2017.

[5] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[6] Z. Ren, K. Dong, Y. Zhou, Q. Liu, and J. Peng, "Exploration via hindsight goal generation," in *Advances in Neural Information Processing Systems*, pp. 13485–13496, 2019.

[7] A. Levy, G. Konidaris, R. Platt, and K. Saenko, "Learning multi-level hierarchies with hindsight," in *International Conference on Learning Representations*, 2018.

[8] P. Dayan, "Improving generalization for temporal difference learning: The successor representation," *Neural Computation*, vol. 5, no. 4, pp. 613–624, 1993.

[9] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu, "Feudal networks for hierarchical reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3540–3549, JMLR. org, 2017.

[10] S. Ahilan and P. Dayan, "Feudal multi-agent hierarchies for cooperative reinforcement learning," *arXiv preprint arXiv:1901.08492*, 2019.

[11] O. Nachum, S. Gu, H. Lee, and S. Levine, "Data-efficient hierarchical reinforcement learning," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 3307–3317, 2018.

[12] T. Osa, V. Tangkaratt, and M. Sugiyama, "Hierarchical reinforcement learning via advantage-weighted information maximization," in *International Conference on Learning Representations*, 2018.

[13] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[14] D. Silver and G. Tesauro, "Monte-carlo simulation balancing," in *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 945–952, ACM, 2009.

[15] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[16] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, pp. 1928–1937, 2016.

[17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[18] A. Juliani, V.-P. Berges, E. Vckay, Y. Gao, H. Henry, M. Mattar, and D. Lange, "Unity: A general platform for intelligent agents," *arXiv preprint arXiv:1809.02627*, 2018.