INTERACTIVE PARALLEL EXPLORATION FOR REINFORCEMENT LEARNING IN CONTINUOUS ACTION SPACES

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, a new interactive parallel learning scheme is proposed to enhance the performance of off-policy continuous-action reinforcement learning. In the proposed interactive parallel learning scheme, multiple identical learners with their own value-functions and policies sharing a common experience replay buffer search a good policy in collaboration with the guidance of the best policy information in the previous search interval. The information of the best policy parameter of the previous search interval is fused in a soft manner by constructing an augmented loss function for policy update to enlarge the overall search space by the multiple learners. The guidance by the previous best policy and the enlarged search space by the proposed interactive parallel learning scheme enables faster and better policy search in the policy parameter space. A working algorithm is constructed by applying the proposed interactive parallel learning scheme to the TD3 algorithm, and numerical results show that the constructed new algorithm outperforms most of the current state-of-the-art reinforcement learning algorithms for continuous action control.

1 INTRODUCTION

Reinforcement learning (RL) for continuous action control is an active research field. In RL, an agent learns a policy through interaction with the environment to maximize the cumulative reward. One of the key issues in RL is the trade-off between exploitation and exploration. Exploitation is to make a best decision based on the already collected information, whereas exploration is to collect more new information about the environment. The balance between the two is important for good RL algorithms. For example, DQN (Mnih et al. (2015)) balances exploitation and exploration by taking actions based on the ϵ-greedy approach. Deep deterministic policy gradient (DDPG) (Lillicrap et al. (2015)) and twin delayed deep deterministic (TD3) (Fujimoto et al. (2018)) policy gradient algorithms promote exploration by adding Ornstein-Uhlenbeck noise and Gaussian noise to the best decision action, respectively. Soft actor-critic (SAC) (Haarnoja et al. (2018)) performs balancing by using a maximum entropy objective. However, most of the previous works focus on exploration to obtain unobserved states or actions.

In this paper, we consider exploration in the policy parameter space by using parallel identical learners for the same environment. By having multiple identical learners for the same environment, we can have increased search capability for a better policy. Parallelism in learning is common in distributed RL (Nair et al. (2015), Mnih et al. (2016), Horgan et al. (2018)), and is shown to be effective in sample generation efficiency and diverse experience. In this paper, we apply parallelism to RL setting to enhance the learning performance.

The proposed algorithm is intended for any off-policy RL algorithms and is composed of a chief, N environment copies of the same environment, and N identical learners with a shared common experience replay buffer and a common base algorithm, as shown in Fig. 1. Each learner has its own value function(s) and policy, and trains its own policy by interacting with its own environment copy with some additional interaction with the chief, as shown in Fig. 1. At each time step, each learner performs an action to its environment copy by using its own policy, stores its experience to the shared common experience replay buffer. Then, each learner updates its value function parameter

and policy parameter by drawing mini-batches from the shared common replay buffer by minimizing its own value loss function and policy loss function, respectively.

One way to parallel learning under the above setup is to implement N fully independent parallel learning without interaction among the learners until the end of time steps and to choose the policy from the learner with the maximum accumulated reward at the end for future use. However, this independent method ignores possible benefit from mutual interaction among the learners during training. In order to harness the benefit of mutual interaction among the learners in parallel learning, we exploit the information from the best learner among all learners periodically during training. Suppose that the value parameter and the policy parameter of each learner are initialized and learning is performed as described above for M time steps. At the end of M time steps, we can determine who is the best learner based on the average of the most recent E_r episodic rewards for each learner. Then, the policy parameter information of the best learner can be used to enhance the learning of other learners for the next M time steps. This information can help learners stuck in local pitfalls escape from the pitfalls and guide other learners for better direction.

One simple way to exploit this best policy parameter information is that we reset the policy parameter of each learner with the policy parameter of the best learner at the beginning of the next M time steps, make each learner perform learning from this initial point in the policy parameter space for the next M time steps, select the best learner again at the end of the next M time steps, and repeat this procedure every M time steps. However, this method has the problem that the search area covered by all N searching policies is very narrow around the previous best policy point. To overcome such drawback, instead of resetting the policy parameter with the best policy parameter every M time steps, we here propose using the policy parameter information of the best learner in a soft manner to enhance the performance of the overall parallel learning. In the proposed scheme, the shared best policy information is used only to guide the policies of other learners. The policy of each learner is updated by improving the performance around a certain distance from the shared guiding policy. The chief periodically determines the best policy among the policies of all learners and distributes the best policy parameter to all learners so that the learners search for better policies around the previous best policy. The chief also enforces that the N searching policies are spread in the policy parameter space with a given distance from the previous best policy point so that the search area in the policy space by all N learners maintains a wide area and does not collapse into a narrow region.

The proposed interactive parallel exploration (IPE) learning method can be applied to any off-policy RL algorithms and the implementation is easy. Furthermore, this proposed method can be extended directly to distributed RL systems. We apply our IPE scheme to the twin delayed deep deterministic (TD3) policy gradient algorithm, which is a state-of-the-art off-policy algorithm, as our base algorithm, and the new algorithm is named the IPE-TD3 algorithm. Numerical result shows that the IPE-TD3 algorithm outperforms TD3 in both the speed of convergence and the final steady-state performance.

2 BACKGROUND

2.1 PARALLEL LEARNING FOR A COMMON ENVIRONMENT

The considered parallel learning setting consists of the environment $\mathcal{E}$ and N policies $\{\pi^1, \cdots, \pi^N\}$. The environment $\mathcal{E}$ is described as a Markov decision process (MDP), defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, r \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is the state transition probability, and $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function. There exist N copies $\{\mathcal{E}^1, \cdots, \mathcal{E}^N\}$ of the environment $\mathcal{E}$, i.e., $\mathcal{E}^1 = \cdots = \mathcal{E}^N = \mathcal{E}$, and the N environment copies may have different random initial seeds. The policy π^i interacts with its corresponding environment copy $\mathcal{E}^i$ and builds up its trajectory $\{(s_t^i, a_t^i, r_t^i), t = 1, 2, \cdots\}$ for each $i = 1, \cdots, N$. At time step t, the environment copy $\mathcal{E}^i$ has a state $s_t^i \in \mathcal{S}$. The policy π^i interacts with the environment copy $\mathcal{E}^i$ by taking an action a_t^i according to π^i given the current state s_t^i. Then, the environment copy $\mathcal{E}^i$ yields the reward $r_t^i = r(s_t^i, a_t^i)$ and makes transition to the next state s_{t+1}^i according to $\mathcal{T}$.

In this paper, in order to account for the actual amount of interaction with the environment, we define *environment steps* as the total number of interactions by all N parallel policies with all N environment copies. Suppose that all N policies generate their trajectories simultaneously in parallel, and

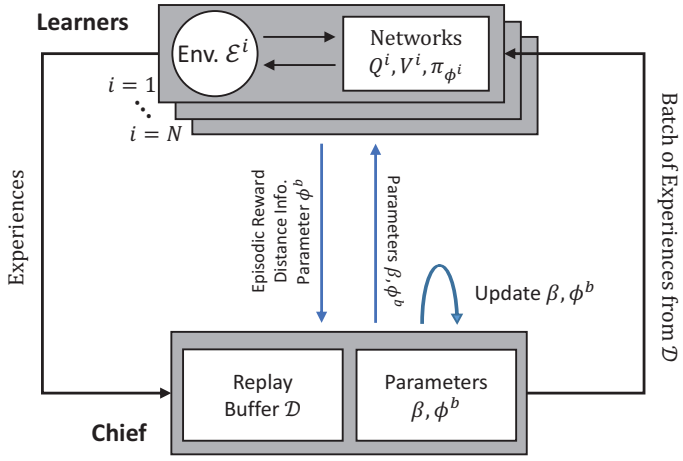


Figure 1: The overall structure of the proposed IPE scheme

suppose that M time steps have elapsed. Then, although the number of elapsed time steps is M, the number of environment steps is NM.

2.2 The Twin Delayed Deep Deterministic Policy Gradient Algorithm

Although the proposed IPE method can be applied to any off-policy RL algorithms, we here consider the TD3 algorithm (Fujimoto et al. (2018)) as our base algorithm in IPE. The TD3 algorithm is a current state-of-the-art off-policy algorithm and is a variant of the deep deterministic policy gradient (DDPG) algorithm (Lillicrap et al. (2015)). The TD3 algorithm tries to resolve two problems in typical actor-critic algorithms: 1) overestimation bias and 2) high variance on the approximation of the Q-function. In order to reduce the bias, the TD3 considers two Q-functions and uses the minimum of the two Q-function values to compute the target value, while in order to reduce the variance on the gradient, the policy is updated less frequently than the Q-functions. Specifically, let Q_{θ_1}, Q_{θ_2} and π_ϕ are two current Q-functions and the current deterministic policy, respectively, and let $Q_{\theta'_1}$, $Q_{\theta'_2}$ and $\pi_{\phi'}$ are the target networks of Q_{θ_1}, Q_{θ_2} and π_ϕ, respectively. The target networks are initialized by the same networks as the current networks. At time step t, the TD3 algorithm takes an action a_t with exploration noise ϵ: $a_t = \pi_\phi(s_t) + \epsilon$, where ϵ is zero-mean Gaussian noise with variance σ^2, i.e., $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Then, the environment returns reward r_t and the state is switched to s_{t+1}. The TD3 algorithm stores the experience (s_t, a_t, r_t, s_{t+1}) at the experience replay buffer $\mathcal{D}$. After storing the experience, the Q-function parameters θ_1 and θ_2 are updated by gradient descent of the following loss functions:

$$L(\theta_j) = \hat{\mathbb{E}}_{(s,a,r,s')\sim\mathcal{D}}\left[(y - Q_{\theta_j}(s,a))^2\right], \quad j = 1, 2 \tag{1}$$

where $\hat{\mathbb{E}}_{(s,a,r,s')\sim\mathcal{D}}$ denotes the sample expectation with an uniform random mini-batch of size B drawn from the replay buffer $\mathcal{D}$, and the target value y is given by

$$y = r + \gamma \min_{j=1,2} Q_{\theta'_j}(s', \pi_{\phi'}(s') + \epsilon), \qquad \epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}^2), -c, c). \tag{2}$$

Here, for the computation of the target value, the minimum of the two target Q-functions is used to reduce the bias. (Note that even in TD3 some kind of parallelism is used.) The procedure of action taking and gradient descent for θ_1 and θ_2 are repeated for d times ($d = 2$), and then the policy and target networks are updated. The policy parameter ϕ is updated by gradient descent by minimizing the loss function for ϕ:

$$L(\phi) = -\hat{\mathbb{E}}_{s\sim\mathcal{D}}\left[Q_{\theta_1}(s, \pi_\phi(s))\right], \tag{3}$$

and the target network parameters θ'_j and ϕ' are updated as

$$\theta'_j \leftarrow (1-\tau)\theta'_j + \tau\theta_j \qquad \phi' \leftarrow (1-\tau)\phi' + \tau\phi. \tag{4}$$

The networks are trained until the number of time steps reaches a predefined maximum.

3 INTERACTIVE PARALLEL POLICY EXPLORATION

We now present the proposed IPE scheme with the parallel environment learning setting described in Section 2.1, and the overall structure is described in Fig. 1.

We have N identical parallel learners with a shared common experience replay buffer $\mathcal{D}$, and all N identical learners employ a common base algorithm, which can be any off-policy RL algorithm. The execution is in parallel. The i-th learner has its own environment $\mathcal{E}^i$, which is a copy of the common environment $\mathcal{E}$, and has its own value function, e.g., Q-function, parameter θ^i and policy parameter ϕ^i. The i-th learner interacts with the environment copy $\mathcal{E}^i$ with some additional interaction with the chief, as shown in Fig. 1. At each time step, the i-th learner performs an action a_t^i to its environment copy $\mathcal{E}^i$ by using its own policy π_{ϕ^i}, stores its experience $(s_t^i, a_t^i, r_t^i, s_{t+1}^i)$ to the shared common experience replay buffer $\mathcal{D}$, for all $i = 1, 2, \cdots, N$. Note that one time step corresponds to N environment steps. Then, at each time step, each learner updates its value function parameter and policy parameter *for N times* by drawing N mini-batches of size B from the shared common replay buffer $\mathcal{D}$ by minimizing its own value loss function and policy loss function, respectively. The N time updates for each learner for each time step is to exploit the samples provided by other $N-1$ learners stored in the shared common replay buffer.

In order to harness the benefit of mutual interaction among the learners in parallel learning, we exploit the information from the best learner periodically during training. Suppose that the Q-function parameter and the policy parameter of each learner are initialized and learning is performed as described above for M time steps. At the end of M time steps, we determine who is the best learner based on the average of the most recent E_r episodic rewards for each learner. Let the index of the best learner be b. Then, the policy parameter information ϕ^b of the best learner can be used to enhance the learning of other learners for the next M time steps. Here, we propose using the information of ϕ^b in a soft manner to enhance the performance of the overall parallel learning. That is, during the next M time steps, whereas we set the loss function $\tilde{L}(\theta^i)$ for the Q-function to be the same as the loss $L(\theta^i)$ of the base algorithm, we set the loss function $\tilde{L}(\phi^i)$ for the policy parameter ϕ^i of the i-th learner as the following *augmented* version:

$$\tilde{L}(\phi^i) = L(\phi^i) + \mathbf{1}_{\{i \neq b\}} \beta \hat{\mathbb{E}}_{s \sim \mathcal{D}} \left[D(\pi_{\phi^i}, \pi_{\phi^b}) \right] \tag{5}$$

where $L(\phi^i)$ is the policy loss function of the base algorithm, $\mathbf{1}_{\{\cdot\}}$ denotes the indicator function, $\beta (> 0)$ is a weighting factor, $D(\pi, \pi')$ be some distance measure between two policies π and π', and $\hat{\mathbb{E}}_{s \sim \mathcal{D}}$ denotes the sample expectation based on mini-batch drawn randomly from the experience replay buffer $\mathcal{D}$. The augmented loss function $\tilde{L}(\phi^i)$ in (5) is composed of two terms $L(\phi^i)$ and $\mathbf{1}_{\{i \neq b\}} \beta \hat{\mathbb{E}}_{s \sim \mathcal{D}} \left[D(\pi^i, \pi^b) \right]$. For the non-best learners in the previous M time steps, the gradient of $\tilde{L}(\phi^i)$ is the mixture of two directions: one is to maximize the return by itself and the other is to follow the previously best learner's policy. Thus, the second term in the right-hand side (RHS) of (5) guides non-best learners towards a good direction in addition to each leaner's self search.

3.1 DESIGN OF THE WEIGHTING FACTOR β

In (5), the weighting factor β is common to all N learners and should be determined judiciously to balance between improving its performance by each learner itself and going towards the previous best policy among the N learners. We adopt an adaptive method to determine the value of β as follows:

$$\beta = \begin{cases} \beta \leftarrow 2\beta & \text{if } \hat{D}_{best} > \min\{\rho \hat{D}_{old}, d_{search}\} \times 1.5 \\ \beta \leftarrow \beta/2 & \text{if } \hat{D}_{best} < \min\{\rho \hat{D}_{old}, d_{search}\}/1.5 \end{cases} \tag{6}$$

where $\hat{D}_{best}$ is the estimated distance between π_{ϕ^i} and π_{ϕ^b} averaged over all $N-1$ non-best learners, and $\hat{D}_{old}$ is the estimated distance between π_{ϕ^i} and the i-th learner's policy $\pi_{\phi^i_{old}}$ at the end of the previous M time steps averaged over all $N-1$ non-best learners, given by

$$\hat{D}_{best} = \frac{1}{N-1} \sum_{i \in I^{-b}} \hat{\mathbb{E}}_{s \sim \mathcal{D}} \left[D(\pi_{\phi^i}, \pi_{\phi^b}) \right], \quad \hat{D}_{old} = \frac{1}{N-1} \sum_{i \in I^{-b}} \hat{\mathbb{E}}_{s \sim \mathcal{D}} \left[D(\pi_{\phi^i}, \pi_{\phi^i_{old}}) \right] \tag{7}$$

Here, $I^{-b} = \{1, \ldots, N\} \setminus \{b\}$, and d_{search} and ρ are predetermined parameters.

This adaptation method is similar to that used in proximal policy optimization (PPO) (Schulman et al. (2017)). The update of β is done every M time steps and the updated β is used for the next M time steps. First, suppose that we do not have the first term $\rho\hat{D}_{old}$ in the minimum of the condition in (6). Then, when the estimated average distance $\hat{D}_{best}$ from the best policy to the remaining policies is smaller than $d_{search}/1.5$, the parameter β is decreased by half. Hence, the movement in the gradient direction of the second term in the RHS of (5) is diminished and the independent movement into the optimization direction for $L(\phi^i)$ becomes more important. So, each policy gradually diverges from the previous best policy serving as the reference point. On the other hand, when $\hat{D}_{best}$ is larger than $d_{search} \cdot 1.5$, the parameter β increases by factor two and the movement towards the previous best policy becomes more important. As time steps elapse, β is settled down so that $\hat{D}_{best}$ is around d_{search}. Hence, the proposed IPE scheme searches a wide area with rough radius d_{search} around the previous best policy in the policy parameter space, as shown in Fig. 2(a). Furthermore, with the first term $\rho\hat{D}_{old}$ in the minimum of the condition in (6), we can control the speed of tracking the best policy. With the first term, if $\frac{\hat{D}_{best}}{\hat{D}_{old}} > \rho$, i.e., the distance from π_{ϕ^i} to π_{ϕ^b} is larger than ρ times the distance from π_{ϕ^i} to $\pi_{\phi^i_{old}}$, then this means that the speed of tracking the best policy is slow. Hence, we increase β by factor two. Otherwise, we decrease β by half. When the search for the current M time steps is finished, the new best policy is selected and a new search for a wide area around the new best policy is performed, as illustrated in Fig. 2(b).

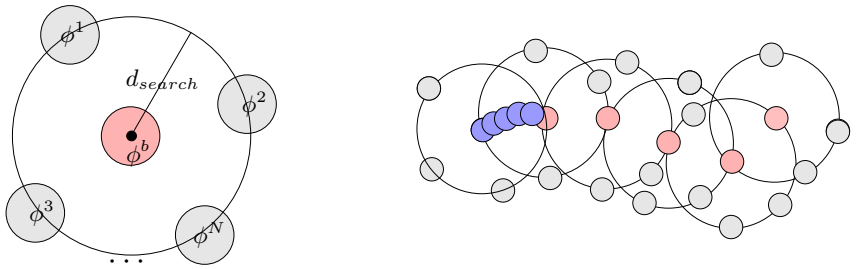


Figure 2: (a) the conceptual search coverage in the policy parameter space by parallel learners (the proper individual search area by each learner may be larger than that in the figure) and (b) an illustration of search comparison: single search (blue) versus the proposed interactive parallel search guided by the best polity at each search interval (pink - best policies during search)

Now, the overall procedure for the proposed IPE scheme is explained with the diagram in Fig. 1. The value function parameter and policy parameter of each learner are initialized. The chief distributes the parameter β and the reference policy parameter ϕ^b, which is the best policy over the previous M time steps, to all learners. At each time step, each learner interacts with its own environment copy by taking its action and receiving the reward and the next state, and stores its experience to the shared common replay buffer $\mathcal{D}$. Then, the i-th learner updates its value function parameter θ^i by minimizing its own value loss function $\tilde{L}(\theta^i)$ which is the same as that of the base algorithm, and updates the policy parameter ϕ^i by minimizing the augmented loss function $\tilde{L}(\phi^i)$ in (5) for N times by drawing N mini-batches from the shared common replay buffer $\mathcal{D}$. Whenever an episode ends for a learner, the learner reports the episodic reward to the chief. The i-th learner reports $\hat{\mathbb{E}}_{s\sim\mathcal{D}}\left[D(\pi_{\phi^i}, \pi_{\phi^b})\right]$ to the chief for computation of $\hat{D}_{best}$ in (7). At every M time steps, the chief updates β according to (6), determines the best policy over the most recent M time steps based on the collected episodic rewards from each learner, obtains the policy parameter ϕ^b from the determined best learner, and distributes the new β and the new best policy parameter ϕ^b to all N learners. This procedure repeats until the time steps reaches the predefined maximum. When the parallel learning based IPE reaches a steady state, we can choose any of the N learners' policies and use the chosen policy for the environment $\mathcal{E}$ in future, since it is expected that at the steady-state the performance of all N policies is more or less similar due to their distance property.

Algorithm 1 The Interactive Parallel Exploration TD3 (IPE-TD3) Algorithm

Require: N: number of learners, T: maximum time steps, M : the best-policy update period, B: size of mini-batch, d: update interval for policy and target networks.

Initialize $\pi_{\phi^1} = \cdots = \pi_{\phi^N} = \pi_{\phi^b}$ randomly.
Initialize $\beta = 1$, $t = 0$
while $t < T$ **do**
  $t \leftarrow t + 1$ (one time step)
  for $i = 1, 2, \cdots, N$ in parallel **do**
    Take an action $a_t^i = \pi^i\left(s_t^i\right) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma^2)$ to environment copy $\mathcal{E}^i$
    Store experience $(s_t^i, a_t^i, r_t^i, s_{t+1}^i)$ to the shared common experience replay $\mathcal{D}$
  end for
  for $i = 1, 2, \cdots, N$ in parallel **do**
    for $k = 1, 2, \cdots, N$ **do**
      Sample a mini-batch $\mathcal{B} = \{(s_{t_l}, a_{t_l}, r_{t_l}, s_{t_l+1})\}_{l=1,\ldots,B}$ from $\mathcal{D}$
      Update θ_j^i, $j = 1, 2$, by gradient descent for minimizing $L(\theta_j^i)$ in (9) with $\mathcal{B}$
      if $k = 0 (\mathrm{mod}\ d)$ **then**
        Update ϕ^i by gradient descent for minimizing $\tilde{L}(\phi^i)$ in (10) with $\mathcal{B}$
        Update the target networks: $(\theta_j^i)' \leftarrow (1-\tau)(\theta_j^i)' + \tau\theta_j^i$, $(\phi^i)' \leftarrow (1-\tau)(\phi^i)' + \tau\phi^i$
      end if
    end for
  end for
  if $t = 0 (\mathrm{mod}\ M)$ **then**
    Select the best policy π_{ϕ^b}
    Adapt β with (6)
  end if
end while

3.2 The IPE-TD3 Algorithm

We apply the proposed IPE scheme with the TD3 algorithm as the base algorithm and the constructed algorithm is named the IPE-TD3 algorithm. With TD3 as the base algorithm, each learner has its own parameters θ_1^i, θ_2^i, and ϕ^i for its two Q-functions and policy. Furthermore, it has $(\theta_1^i)'$, $(\theta_2^i)'$, and $(\phi^i)'$ which are the parameters of the corresponding target networks. For the distance measure between two policies, we use the mean square difference given by

$$D(\pi(s), \tilde{\pi}(s)) = \frac{1}{2} \left\| \pi(s) - \tilde{\pi}(s) \right\|_2^2 . \tag{8}$$

For the i-th learner, as in TD3, the parameters θ_j^i, $j = 1, 2$ are updated every time step by minimizing

$$\tilde{L}(\theta_j^i) = \hat{\mathbb{E}}_{(s,a,r,s')\sim\mathcal{D}} \left[\left(y - Q_{\theta_j^i}(s, a) \right)^2 \right] \tag{9}$$

where $y = r + \gamma \min_{j=1,2} Q_{(\theta_j^i)'}(s', \pi_{(\phi^i)'}(s') + \epsilon)$, $\epsilon \sim \mathrm{clip}(\mathcal{N}(0, \tilde{\sigma}^2), -c, c)$. The parameter ϕ^i is updated every d time steps by minimizing the following augmented loss function:

$$\tilde{L}(\phi^i) = \hat{\mathbb{E}}_{s\sim\mathcal{D}} \left[-Q_{\theta_1^i}(s, \pi_{\phi^i}(s)) + \mathbf{1}_{\{i \neq b\}} \frac{\beta}{2} \left\| \pi_{\phi^i}(s) - \pi_{\phi^b}(s) \right\|_2^2 \right] . \tag{10}$$

With these loss functions, all procedure is the same as the general IPE procedure described previously. The pseudocode of the IPE-TD3 algorithm is shown in the Algorithm 1.

4 Experiments

We evaluated the performance of the proposed IPE-TD3 and current state-of-the-art on-policy and off-policy baseline algorithms on several MuJoCo enrivonments (Todorov et al. (2012)). The baseline algorithms are Proximal Policy Optimization (PPO) (Schulman et al. (2017)), Soft Actor-Critic (SAC) (Haarnoja et al. (2018)), and TD3 (Fujimoto et al. (2018)).

4.1 PARAMETER SETTING

All parameters we used for evaluation are the same as those in the original papers (Schulman et al. (2017); Haarnoja et al. (2018); Fujimoto et al. (2018)).

TD3 The networks for two Q-functions and the policy have 2 hidden layers. The first and second layers have sizes 400 and 300, respectively. The non-linearity function of the hidden layers is ReLU, and the activation functions of the last layers of the Q-functions and the policy are linear and hyperbolic tangent, respectively. We used the Adam optimizer with learning rate 10^{-3}, discount factor $\gamma = 0.99$, target smoothing factor $\tau = 5 \times 10^{-3}$, the period $d = 2$ for updating the policy. The experience replay buffer size is infinite (this means we use all experiences to update parameters), and the mini-batch size B is 100. The standard deviation for exploration noise σ and target noise $\tilde{\sigma}$ are 0.1 and 0.2, respectively, and the noise clipping factor c is 0.5.

IPE-TD3 In addition to the parameters for TD3, we used $N = 4$ learners, the period $M = 250$ of updating the best policy and β, the number of recent episodes $E_r = 10$ for determining the best policy π_{ϕ^b}. The parameters d_{search} and ρ for the exploration range are 0.8 and 2, respectively.

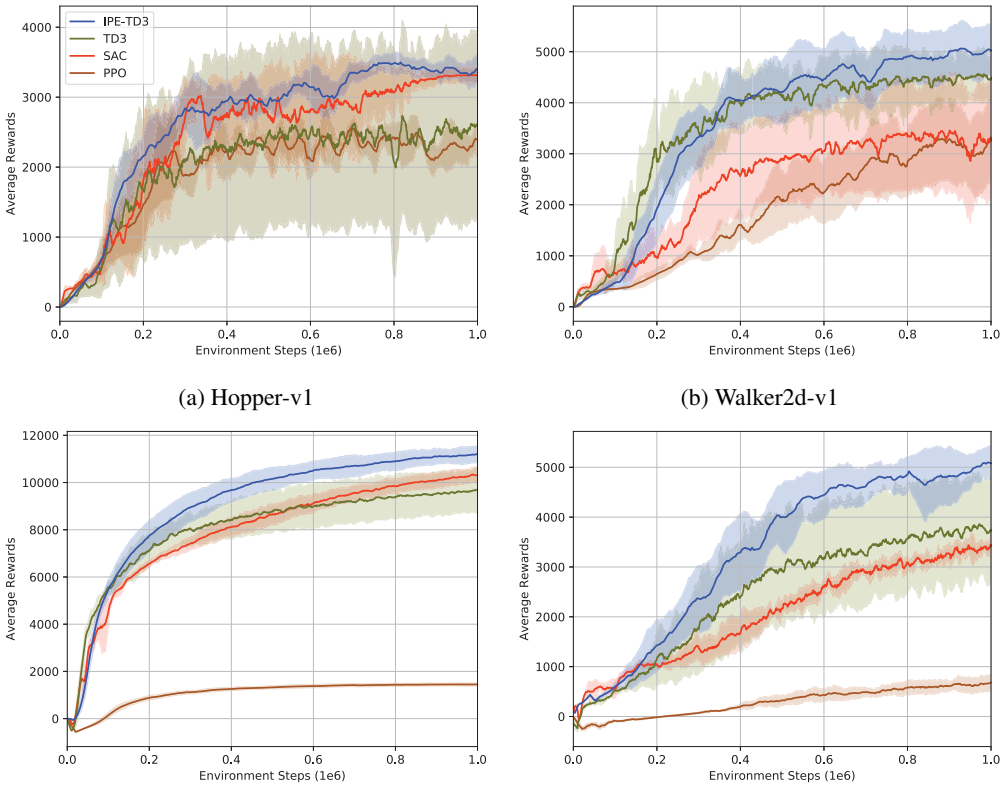
SAC The networks for the state-value function, two Q-functions, and the policy have 2 hidden layers of size 256. The activation functions for the hidden layers and the last layers are ReLU and linear, respectively. We used the Adam optimizer with learning rate 3×10^{-4}, discount factor $\gamma = 0.99$, and target smoothing factor $\tau = 5 \times 10^{-3}$. The algorithm is trained by random mini-batches of size $B = 256$ from the experience replay buffer of the maximum size 10^6. Reward scale for updating Q-functions is 5 for all environments.

PPO The networks of the state-value function and the policy have 2 hidden layers of size 64. The activation functions for the hidden layers and the last layer are hyperbolic tangent and linear, respectively. We used the Adam optimizer with initial learning rate 3×10^{-4}, discount factor $\gamma = 0.99$, and GAE parameter $\lambda = 0.95$. The horizon is 2048 which is the number of samples generated by the same policy at one iteration. The policy and value networks are updated $10 \times 2048/64$ times using random mini-batches of size 64 from the collected 2048 experiences generated by the same policy. The clipping range of importance sampling ratio is 0.2.

4.2 COMPARISON TO BASELINES

To have sample-wise fair comparison among the baseline and IPE-TD3 algorithms, we obtain the performance with respect to environment steps (not time steps), which is defined as the total number of interactions with the environment by the agent. This comparison makes sense because the performance at the same environment steps means that all algorithms use the same number of samples obtained from the environment. The performance is obtained through the evaluation method that is similar to those in (Haarnoja et al. (2018); Fujimoto et al. (2018)). Evaluation of the policies are conducted every R_{eval} environment steps with $R_{eval} = 2048, 4000$, and 1000 for PPO, IPE-TD3, and the other baseline algorithms, respectively. At each evaluation instant, the agent (or learner) fixes its policy as the one at the evaluation instant, and interacts with the same environment separate for the evaluation purpose with the fixed policy to obtain 10 episodic rewards. The average of these 10 episodic rewards is the performance at the evaluation instant. In the case of IPE-TD3, each of the N learners fixes its policy as the one at the evaluation instant, and interacts with the environment with the fixed policy to obtain 10 episodic rewards. First, the 10 episodic rewards are averaged for each learner and the 10-episode-average rewards of the N learners are averaged again to yield the performance at that evaluation instant. We performed this operation for five different random seeds, and the mean and variance of the learning curve are obtained from these five simulations. The policies used for evaluation are stochastic for PPO and deterministic for the others.

Fig. 3 shows the learning curves over one million environment steps for several MuJoCo tasks: Hopper-v1, Walker2d-v1, HalfCheetah-v1, and Ant-v1. First, it is observed that the performance of TD3 here is similar to that in the original TD3 paper (Fujimoto et al. (2018)), and the performance of other baseline algorithms is also similar to that in the original papers (Schulman et al. (2017); Haarnoja et al. (2018)). It is seen that the IPE-TD3 algorithm outperforms the state-of-the-art RL algorithms in terms of both the speed of convergence with respect to environment steps and the final steady-state performance (except in Walker2d-v1, the initial convergence is a bit slower than TD3.) Especially, in the cases of Hopper-v1 and Ant-v1, TD3 has large variance and this means that the



(a) Hopper-v1

(b) Walker2d-v1

(c) HalfCheetah-v1

(d) Ant-v1

Figure 3: Performance for PPO (brown), SAC (red), TD3 (green), and IPE-TD3 (proposed method, blue) on MuJoCo tasks.

performance of TD3 is quite dependent on the initial condition of the environment and it is not easy for TD3 to escape out of a bad local point in certain environments. However, it is seen that IPE-TD3 yields much less variance as compared to TD3. This implies that the wide area search by IPE in the policy parameter space helps the learners escape out of bad local pitfalls. Indeed, it is seen that the wide area search around the previous best policy point in the policy parameter space by IPE yields faster and better policy search.

5 Conclusion

In this paper, we have proposed a new interactive parallel learning scheme, IPE, to enhance the performance of off-policy RL systems. In the proposed IPE scheme, multiple identical learners with their own value-functions and policies sharing a common experience replay buffer search a good policy with the guidance of the best policy information in the previous search interval. The information of the best policy parameter of the previous search interval is fused in a soft manner by constructing an augmented loss function for policy update to enlarge the overall search space by the multiple learners. The guidance by the previous best policy and the enlarged search space by IPE enables faster and better policy search in the policy parameter space. The IPE-TD3 devised by applying the proposed IPE scheme to TD3 outperforms most of the current state-of-the-art continuous-action RL algorithms.

REFERENCES

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1587–1596, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1861–1870, 2018.

Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado Van Hasselt, and David Silver. Distributed prioritized experience replay. In *International Conference on Learning Representations*, 2018.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, pp. 1928–1937, 2016.

Arun Nair, Praveen Srinivasan, Sam Blackwell, Cagdas Alcicek, Rory Fearon, Alessandro De Maria, Vedavyas Panneershelvam, Mustafa Suleyman, Charles Beattie, Stig Petersen, et al. Massively parallel methods for deep reinforcement learning. *arXiv preprint arXiv:1507.04296*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 5026–5033. IEEE, 2012.