IJCAI–23 Formatting Instructions

Author Name
Affiliation
email@example.com

Abstract

Reinforcement learning usually models the environment as a Markov Decision Process (MDP), but many practical problems are Continuous-Time Markov Decision Process (CTMDP), in which the agent may need to increase the decision frequency to respond quickly to the environment, but the excessive frequency may reduce the training efficiency. In this paper, we proposed Randomness Reuse Algorithm (RRA) to solve CTMDP. We theoretically prove the instantaneity, and stability of our RRA. We have evaluated our algorithm in *Starcraft* and compared it with the SOTA methods. The empirical results show that our algorithm can better deal with CTMDP.

1 Introduction

Sequential decision problem is a classic research field in artificial intelligence, which covers a wide range of possible applications with the potential to impact many domains, such as robotics, finance, and self-driving cars. Sequential decision problem is often modeled as Markov Decision Processes (MDP) by eliminating some factors of the actual situation, but these eliminated factors may affect the finding of the optimal policy. To model the real-world application more accurately, we model sequential decision problem with Continuous-Time Markov Decision Process (CTMDP).

Continuous-time problems are seldom studied in common Reinforcement Learning (RL), but are often studied in adaptive control [Doya, 2000; Vrabie *et al.*, 2009; Modares *et al.*, 2015; He *et al.*, 2019; Vamvoudakis and Lewis, 2010; Wang *et al.*, 2020; Du *et al.*, 2020], which can be regarded as a special class of RL problems. In adaptive control, the states are generally assumed to be differentiable with respect to time or even linear. With these assumptions, learning of policies will relatively become easier. The optimal policies are usually learned through iterative optimization from an initial policy, without random exploration such as epsilon-greedy. These assumptions may be satisfied when the state is composed of physical quantities, but not satisfied when the state is discrete or is an image. Without the differentiability assumption, policies must be learned through other methods.

In RL, most methods transform CTMDP to a discrete-time MDP by sampling, but some tasks require agents to respond as quickly as possible, such as self-driving. Theoretically, the response time of the model can be improved by increasing the sampling frequency. However, we argue that if the sampling frequency is increased, the training efficiency will be decreased. This is mainly due to two reasons: First, in some cases, the efficiency of exploration will be reduced. Second, the relative error of the advantage function will be larger.

To solve this contradiction, we proposed Randomness Reuse Algorithm (RRA), which is triggered by the change of the output of the policy function. The output of the policy function is a distribution of action. The greater the change of this distribution, the greater the probability of updating the action. In this way, if the state changes significantly and changes the output of the policy function, the agent can respond immediately, and if the state changes slowly, the action can remain stable to ensure the efficiency of exploration and training.

In summary, the contributions are as follows:

1. We proposed the Randomness Reuse Algorithm to reduce the response time without affecting the training. In this algorithm, agent responds to the change of state by updating a hidden state, which is a distribution of action. The hidden state will change as little as possible under the premise of ensuring the response speed, so as to make the policy stable.

2. We theoretically proved our model has two good properties: First, our model can immediately respond to the change of state. Second, under appropriate conditions, our model can keep stable if the frequency tends to infinity.

2 Continuous-Time Markov Decision Process

Different from discrete-time MDP, the states and actions in tracks of CTMDP are not sequences, but functions of time. Here, we will define the state chain and action chain.

Definition 2.1. ***State chain*** *$s_{t_1}^{t_2}$ denotes all states at time $t \in [t_1, t_2]$. $\forall t \in [t_1, t_2]$, there exists a state $s(t)$ corresponds to t.*

Definition 2.2. ***Action chain*** *$a_{t_1}^{t_2}$ denotes all actions at time $t \in [t_1, t_2]$. $\forall t \in [t_1, t_2)$, there exists an action $a(t)$ corresponds to t.*

State space and action space can be continuous or discrete.

Definition 2.3. ***Continuous Time Markov Decision Process*** *(CTMDP) is a quintet $< \mathbb{S}, \mathbb{A}, p, r, \pi >$, where $\mathbb{S}$ denotes the state space; $\mathbb{A}$ denotes the action space; $p(s_{t+\tau}|s_t, a_t^{t+\tau})$ is transition probability function and define the transition probability from time t to time $t+\tau$, where $\tau \in \mathbb{R}$ is time interval to next state; $r(s_t^{t+\tau}, a_t^{t+\tau})$ is reward from time t to time $t+\tau$; and $\pi(a_t|s_t)$ is policy function.*

For the transition function in Definition 2.3, there are Property 2.1 and Property 2.2.

Property 2.1. ***Markov property*** *Given state s_{t_1} at time t_1 and action chain $a_{t_1}^{t_1+\tau_1}$ from time t_1 to time $t_1+\tau_1$, the state $s_{t_1+\tau_1}$ is independent of the previous states s_{t_a}, $0 \leq t_a < t_1$, and previous actions a_{t_b}, $0 \leq t_b < t_1$.*

Property 2.2. *For any state s_{t_1}, s_{t_3}, action chain $a_{t_1}^{t_3}$, and time $t_2 \in (t_1, t_3)$, Equation (1) is satisfied.*

$$p(s_{t_3}|s_{t_1}, a_{t_1}^{t_3}) = \sum_{s_{t_2} \in \mathbb{S}} p(s_{t_3}|s_{t_2}, a_{t_2}^{t_3}) \cdot p(s_{t_2}|s_{t_1}, a_{t_1}^{t_2}) \quad (1)$$

For the reward function in Definition 2.3, there is Property 2.3. The object of CTMDP is finding the best policy π^* to maximize the summary of rewards.

Property 2.3. *For any state chain $s_{t_1}^{t_3}$, action chain $a_{t_1}^{t_3}$ and time $t_2 \in (t_1, t_3)$, $r(s_{t_1}^{t_3}, a_{t_1}^{t_3}) = r(s_{t_1}^{t_2}, a_{t_1}^{t_2}) + r(s_{t_2}^{t_3}, a_{t_2}^{t_3})$.*

The transition of CTMDP is as follows: Because operation of agents costs time, there is a time interval τ_i between decision time t_i and time t_{i+1}. From time t_i to time $t_{i+1} = t_i + \tau_i$, agent outputs action a_{t_i} follows the distribution $\pi(a_{t_i}|s_{t_i})$. Then, the action will change from $a_{t_{i-1}}$ to a_{t_i} after time t_i and before time t_{i+1}. Then, the state transfers to $s_{t_{i+1}}$ based on $p(s_{t_{i+1}}|s_{t_i}, a_{t_i}^{t_{i+1}})$, and produce reward $r_{t_i}^{t_{i+1}} = r(s_{t_i}^{t_{i+1}}, a_{t_i}^{t_{i+1}})$. If $s_{t_{i+1}}$ is final state or time t_{i+1} is greater than max time, this process will terminate. To reduce the delay of agent response for state, the time interval should be small. If the operation speed is fast enough, the time intervals can be set as a constant close to zero.

3 The problem of RL method in solving CTMDP

To explain the problem in CTMDP, we will introduce a simple example of driving. There are two states: $safe$ and $danger$. The state transitions randomly between these two states. There are two different actions: $forward$ and $stop$.

If the current state is same with last state $s_{t_i} = s_{t_{i-1}}$, there is a $p_c = 1 - \sum_a \pi(a|s)^2$ probability of action change $a_{t_i} \neq a_{t_{i-1}}$, and average frequency of action change is $\frac{p_c}{\tau}$, where τ is the time interval between the decision times of agent. There is a delay $d = d_1 + d_2$ from state change to action change as shown in Figure 1, where the expect of delay d_1 is $\frac{\tau}{2}$, and d_2 is the operation time of agent, $d_2 < \tau$. To reduce the delay, the intervals need to be lower, which will increase the frequency of action changes. For most RL methods, if intervals tend to zero, the frequency of changing action will tends to infinity, which will cause two problems.

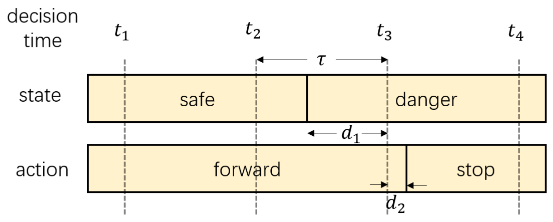

Figure 1: A simple example

First, the efficiency of exploration may be reduced. There is an example: Suppose the transfer function of state is as shown in Equation (2).

$$s_{t+\Delta t} = s_t + f(s,a) \cdot \Delta t + o(\Delta t) \quad (2)$$

where t is time, $f(s,a)$ is derivative of state with respect to time and $o(\Delta t)$ is high-order infinitesimal of Δt.

Ideally, the policy function should explore different states by outputting different actions. However, according to the law of large numbers, the state after Δt will tends a deterministic state if interval τ tends to zero as shown in Equation (3). Without efficient exploration, it will take longer to find the optimal policy.

$$\begin{aligned}\lim_{\tau \to 0} s_{t+\Delta t} &= s_t + o(\Delta t) + \lim_{\tau \to 0} \sum_{i=0}^{\lfloor \frac{\Delta t}{\tau} \rfloor} f(s_{t+i\tau}, a_{t+i\tau})\tau \\ &= s_t + \sum_a \pi(a|s_t) f(s,a) \cdot \Delta t + o(\Delta t)\end{aligned} \quad (3)$$

where $\pi(a|s_{t_1}) \cdot f(s_{t_1}, a)$ is assumed continuous with respect to time t_1, $t_1 \in [t, t+\Delta t]$.

Second, there will be greater error in estimating the optimal policy. If the frequency of changing action will tends to infinity, the advantage function used to estimate the optimal policy, as shown in Equation (3), will tends to zero.

$$A(s,a) = Q(s,a) - \sum_a \pi(a|s) Q(s,a) \quad (4)$$

where $A(s,a)$ is the advantage function, $Q(s,a)$ is the expected value of sum of the rewards if agent executes action a at state s.

Assume we use n-step training, and the decision frequency becomes k times. The expect value of $A(s,a)$ will become about one k-th of the previous value. In order to maintain the training speed, the training method need to become nk-step, which means the variance of $A(s,a)$ is basically unchanged. In policy updating, we need to calculate the average of $A(s,a)$, denoted by $\overline{A}(s,a)$. According to Equation (5), the relative error of the advantage function will be greater, and the error in estimating the optimal policy will be greater.

$$\begin{aligned} Var(\overline{A'}(s,a)) &\geq \frac{Var(\overline{A}(s,a))}{k} \\ \frac{STDEV(\overline{A'}(s,a))}{E(\overline{A'}(s,a))} &\geq \frac{STDEV(\overline{A}(s,a))\sqrt{k}}{E(\overline{A}(s,a))} \end{aligned} \quad (5)$$

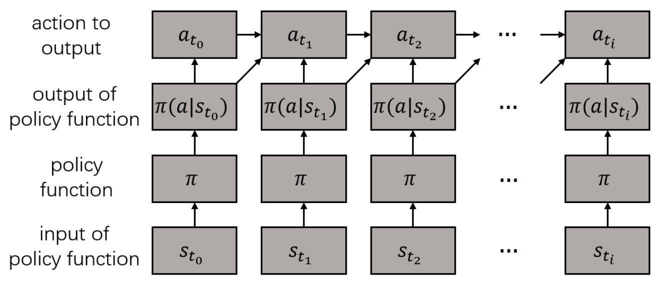

Figure 2: Algorithm of discrete action, where s_{t_i} is state, the output of the policy function $\pi(a|s_{t_i})$ at time t_i is a distribution of action a.

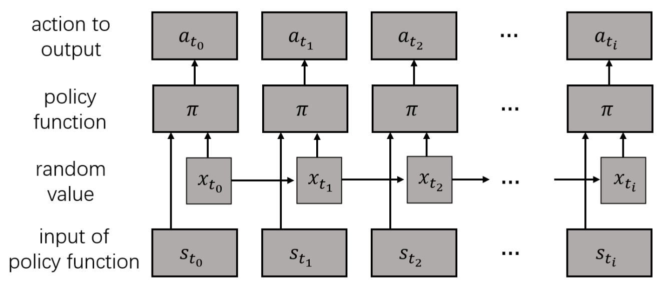

Figure 3: Algorithm of continuous action.

where $\overline{A}(s,a)$ is the previous average of $A(s,a)$, $\overline{A'}(s,a)$ is the average of $A(s,a)$ after frequency is increased, Var means variance and $STDEV$ means standard deviation.

To solve these two problems caused by increased frequency of action change, we proposed an adaptive action changing algorithm which can reduce the delay without changing actions too frequently.

4 Randomness Reuse Algorithm

In CTMDP, the frequency of action change should be independent of the decision frequency. To achieve this, we propose Randomness Reuse Algorithm (RRA). In RRA, the frequency of action change depends on the speed of state change, which is defined as $\frac{d(\pi(\cdot|s_{t_i}),\pi(\cdot|s_{t_{i+1}}))}{\tau_i}$, where d is the distance function of two distributions. Different from most RL methods, action of RRA depends not only on the current state, but also on the last state and last action. In other words, the RRA can be discribed as a conditional probability function: $p(a_{t_i}|s_{t_{i-1}},a_{t_{i-1}},s_{t_i})$, where t_i is current time and t_{i-1} is last time. In order to consistent with the policy π, $p(a_{t_i}|s_{t_{i-1}},a_{t_{i-1}},s_{t_i})$ should meet the following the condition in Equation (6). The RRA used for continuous action spaces and discrete action spaces are different.

$$\sum_{a_{t_{i-1}}\in\mathbb{A}} p(a_{t_i}|s_{t_{i-1}},a_{t_{i-1}},s_{t_i})\cdot\pi(a_{t_{i-1}}|s_{t_{i-1}}) = \pi(a_{t_i}|s_{t_i}) \tag{6}$$

4.1 Randomness Reuse Algorithm of Discrete Action

When the action space is discrete, the RRA is illustrated by Figure 2. At time t_0, action a_{t_0} is sampled from $\pi(a|s_{t_0})$. At other time t_i, $i=1,2,\ldots$, the probability of action a_{t_i} can be represented as $p(a_{t_i}|a_{t_{i-1}},s_{t_{i-1}},s_{t_i})$, where $a_{t_{i-1}}$ is last action, $\pi(a|s_{t_{i-1}})$ is last output of policy function and $\pi(a|s_{t_i})$ is current output of policy function.

In RRA, the action changes in two cases, otherwise a_{t_i} equals $a_{t_{i-1}}$. First, if $\pi(a_{t_{i-1}}|s_{t_i}) < \pi(a_{t_{i-1}}|s_{t_{i-1}})$, the action will has a $1-\frac{\pi(a_{t_{i-1}}|s_{t_i})}{\pi(a_{t_{i-1}}|s_{t_{i-1}})}$ probability of changing to a new action. The new action follows the distribution in Equation (7).

$$p_3(a_{t_i}|s_{t_{i-1}},s_{t_i}) = \frac{relu(\pi(a_{t_i}|s_{t_i})-\pi(a_{t_i}|s_{t_{i-1}}))}{\sum_{a\in\mathbb{A}} relu(\pi(a|s_{t_i})-\pi(a|s_{t_{i-1}}))} \tag{7}$$

where $relu(x)=max(x,0)$.

Second, if action didn't change in first step, action has a $p_L(a_{t_{i-1}},s_{t_i})\cdot\tau_1$ probability of changing to a new action from the distribution $p_4(a_{t_i}|s_{t_{i-1}},s_{t_i})$ in Equation (8). The function $p_L(a_{t_{i-1}},s_{t_i})$ is the minimal frequency of action change. It is used to prevent the action from being constant. A native method is set p_L as a constant.

$$\begin{aligned} &p_2(a_{t_i}|s_{t_{i-1}},s_{t_i})\\ =&min(p_L(a_{t_i},s_{t_i})\tau_1,1)min(\pi(a_{t_i}|s_{t_{i-1}}),\pi(a_{t_i}|s_{t_i}))\\ &p_4(a_{t_i}|s_{t_{i-1}},s_{t_i}) = \frac{p_2(a_{t_i}|s_{t_{i-1}},s_{t_i})}{\sum_{a_1\in\mathbb{A}} p_2(a_{t_i}|s_{t_{i-1}},s_{t_i})}\end{aligned} \tag{8}$$

4.2 Randomness Reuse Algorithm of Continuous Action

When the action space is continuous and huge, there will be an infinite number of possible actions. As a result, the action can't be directly sampled from $\pi(a|s)$. In this case, the action can be generated by random value: $a=f(s,x)$ where x is random variable in any form, and the distribution of x is $p_x(x)$. If so, the RRA is illustrated by Figure 3. At time t_0, random variable x_{t_0} is generated according to $p_x(x)$. At other time t_i, $i=1,2,\ldots$, random variable x_{t_i} is generated from RRA. In RRA of continuous time, the distance of state change is defined as d_i in Equation (9), $d_i\in[0,1)$.

$$d_i = 1-e^{-d(\pi(\cdot|s_{t_{i-1}}),\pi(\cdot|s_{t_i}))} \tag{9}$$

x_{t_i} has a $d_i+(1-d_i)p_L(s)\cdot\tau_1$ probability of generated according to $p_x(x)$, where $p_L(a_{t_{i-1}},s_{t_i})$ is the minimal frequency of action change. Otherwise x_{t_i} equals to $x_{t_{i-1}}$.

4.3 Instantaneity and Stability

In this section, we prove the following properties of our model with RRA.

(1) instantaneity, which means the agent can immediately change its action according to the new state at every decision time. If so, the response time is only related to the decision frequency.

(2) stability, which means the policy of the model is convergent when the frequency tends to positive infinity. To perform well at any frequency, stability is required to be guaranteed.

Instantaneity We say the model is instant, if the probability of action a_{t_i} at time t_i is always corresponding to the current state chain $s_0^{t_i}$ such as $p(a_{t_i}) = \pi(a_{t_i}|s_{t_i})$ instead of corresponding to a previous state chain such as $p(a_{t_i}) = \pi(a_{t_i}|s_{t_j}), j < i$. For our model with RRA, theorem 1 holds.

Theorem 1. *If* $p(a_{t_{i-1}}|s_0^{t_{i-1}}) = \pi(a|s_{t_{i-1}})$*, then* $p(a_{t_i}|s_0^{t_i}) = \pi(a|s_{t_i})$.

According to theorem 1, it can be proved by mathematical induction that $p(a_t|s_0^t) = \pi(a_t|s_t)$, which means the probability of a is always corresponding to the current state, so the model can immediately respond to the change of state.

Stability The main different between MDP and CTMDP is the time intervals of CTMDP are random and may be very little. To solve CTMDP, the policy should converge to the optimal policy when time intervals tend to zero. However, for most RL methods, the policies are not convergent. That's because, when time intervals tend to zero, the agent will change the output action very frequently. Different from previous RL methods, the probability of our agent changing its action in a period of time is convergent.

Theorem 2. *For state chain* s_0^T*, if* $\pi(a|s_t)$ *is Lipschitz continuous with respect to time t,* $t \in [0, T]$*, the expected number of action changes from time* 0 *to time* T *is convergent when the max time interval tends to zero.*

4.4 Train

Our policy function is trained by policy gradient algorithm.

We employ Algorithm 1 to train the policy function, where $\alpha_V, \alpha_\theta, \lambda_V, \lambda_\theta$ are hyperparameters, and θ is the parameters of policy function. Because we assume the time is continuous and the time interval could be random, we use γ^τ and λ^τ instead of γ and λ. This algorithm is modified from Generalized Advantage Estimator (GAE) [Schulman *et al.*, 2015].

Algorithm 1 train

Input: Trajectory $s_0^{t_{end}}$, $a_0^{t_{end}}$, time sequence $t_0, t_1 \dots t_n$, last parameters θ.
Output: Next parameters θ

```
1: $G_V \leftarrow 0$
2: $G_\theta \leftarrow 0$
3: for i = $n-1$ to 0 do
4:     $\tau \leftarrow t_{i+1} - t_i$
5:     $r \leftarrow R(s_{t_i}^{t_{i+1}}, a_{t_i}^{t_{i+1}})$
6:     $G_V \leftarrow G_V \cdot (\gamma \cdot \lambda_V)^\tau + r_t + \gamma^\tau \cdot V(s_0^{t_{i+1}}) - V(s_0^{t_i})$
7:     $V(s_0^{t_i}) \leftarrow V(s_0^{t_i}) + \alpha_V \cdot G_V$
8:     $G_\theta \leftarrow G_\theta \cdot (\gamma \cdot \lambda_\theta)^\tau + r_t + \gamma^\tau \cdot V(s_{t_{i+1}}) - V(s_{t_i})$
9:     $\theta \leftarrow \theta + \alpha_\theta \cdot \nabla log(\sum_{a \in a[s_{t_i}^{tn}, a_{t_i}^{tn}]} \pi(a|s_0^{t_i}; \theta)) \cdot G_\theta$
10: end for
```

5 Experiment

To explore whether our model can better deal with continuous time environment. We apply our model on Multi-Agent Proximal Policy Optimization (MAPPO) [Yu *et al.*, 2021], which is a state-of-the-art model based on policy gradient. We compared our model with other state-of-the-art models at different frequencies.

5.1 Environment

We test our method in *Starcraft* II Learning Environment (SC2LE) [Vinyals *et al.*, 2017], which is an environment of Real-Time Strategy (RTS) game, and is often used as an experimental environment for RL [Usunier *et al.*, 2016]. In this game, the player controls multiple units at the same time, so it is also often used in experiments of multi-agent reinforcement learning [Samvelyan *et al.*, 2019]. Although the time of *Starcraft* is discrete, since the states of two adjacent frames are very similar, we can regard the time of this game as continuous. In this game, the operation speed is important.

5.2 Task

The task of our experiment is from *Starcraft* Multi-Agent Challenge (SMAC)[Samvelyan *et al.*, 2019], which is a common benchmark focusing on micro-management of *Starcraft* II. SMAC includes multiple challenges. In each challenge, there are multiple units that required multiple independent agents to control them. These units battle with opposing units under the centralized control of the game's built-in scripted AI. SMAC also defines the states, observations, actions, and rewards of agents. The action space of agents is dispersed. In our experiment, we chose the 3m task to test models.

5.3 Compared model

We have compared our model with MAPPO, Value Decomposition Networks VDN [Sunehag *et al.*, 2017] and QMix [Rashid *et al.*, 2018].

MAPPO is a policy-based model of Multi-Agent RL. MAPPO applied Proximal Policy Optimization (PPO) in cooperative multi-agent environment and achieved strong performance in *Starcraft* multi-agent challenge [Yu *et al.*, 2021].

VDN and QMix are value-based models of Multi-Agent RL. VDN decomposes the central state-action value into a sum of individual agent values, and QMix employs a network that estimates central state-action value as a complex nonlinear combination of per-agent values.

5.4 Setting

We add our algorithm to MAPPO as MAPPO+RRA. In fairness, all details of MAPPO and MAPPO+RRA are same. The policy function is designed as above, and f_l is initialized as constants. In SMAC, agents have access to the last actions of allied units that are in the field of view. However, in our algorithm, the agents do not take the previous actions as input, so we canceled this access, which also means that the agents cannot see the actions of other agents.

To compare the performance of models at different frequencies, we use game frames instead of agent steps to measure training time. The training time of all models is 2,560,000 frames.

5.5 Result

We have compared the performance of different models at different frequencies in Figure 4. For MAPPO and MAPPO+RRA, we have computed the median win rate of the final five batches. For VDN and QMix, we have tested 32 episodes every 10000 steps and computed the median win rate of the final three tests. We have measured the median of win rates over 5 seeds for all models.

For MAPPO and MAPPO+RRA, The win rate is basically positively related to the frequency. For QMix and VDN, the win rate starts to decrease when the frequency exceeds 16 frames per step. If the frequency is 2 or 1 frames per step, the win rate of QMix and VDN will be zero. Theoretically, the higher the frequency, the higher the upper limit of the model. However, value-based models perform poorly when the frequency is high. It may be that the increase of frequency increases the relative error of the advantage function, and the value-based models are sensitive to the error.

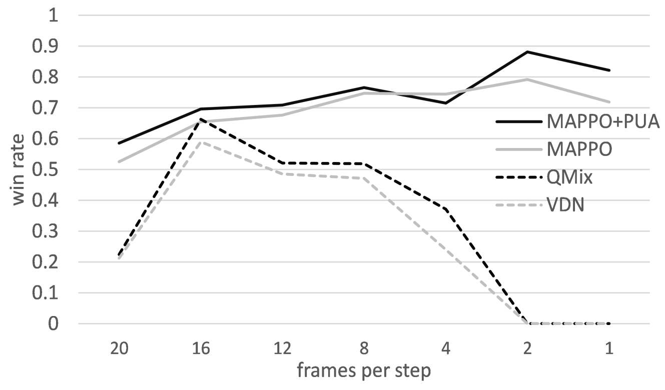

Figure 4: Win rates at different frequencies.

The win rate of MAPPO+RRA is generally higher than that of MAPPO. That's because RRA increases the convergence speed of MAPPO. We have compared the convergence rate in detail in Figure 5, where the convergence speed of MAPPO+RRA is obviously faster than that of MAPPO.

6 Conclusion

To improve the ability of reinforcement learning to solve Continuous Time Markov decision processes, we proposed Randomness Reuse Algorithm, which is an RL model based for CTMDP. This model can reduce the response time of the agent without affecting the training efficiency. We proved the instantaneity, stability, and optimality of this model. In the experiment, we applied our model on MAPPO in *starcraft*, and the empirical results show that our algorithm can perform better.

7 Future Work

Hierarchical Reinforcement Learning (HRL)[Bacon *et al.*, 2017; Vezhnevets *et al.*, 2017; Eysenbach *et al.*, 2018; Li *et al.*, 2019] may be an appropriate method.

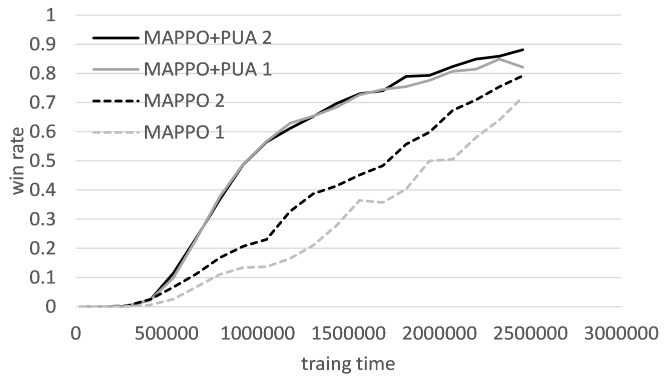

Figure 5: Comparison of convergence speed between MAPPO and MAPPO+RRA. 2 means 2 frames per step, and 1 means 1 frame per step.

A Proof of Theorem 1

Proof. If $p(a_{t_{i-1}}|s_0^{t_{i-1}}) = \pi(a|s_{t_{i-1}})$, $p(a_{t_{i-1}}|s_0^{t_{i-1}})$ can be represented as shown in Equation (10)

$$\begin{aligned} p(a_{t_i}|s_0^{t_i}) &= \sum_{a_0^{t_i}} p(a_{t_i}|a_{t_{i-1}}, s_{t_{i-1}}, s_{t_i}) \cdot p(a_0^{t_i}|s_0^{t_i}) \\ &= \sum_{a_{t_{i-1}} \in \mathbb{A}} p(a_{t_i}|a_{t_{i-1}}, s_{t_{i-1}}, s_{t_i}) \cdot \pi(a_{t_{i-1}}|s_{t_{i-1}}) \end{aligned} \tag{10}$$

(i) If $\pi(a_{t_i}|s_{t_i}) < \pi(a_{t_i}|s_{t_{i-1}})$.

$$\begin{aligned} & p(a_{t_i}|s_0^{t_i}) \\ =& \frac{\pi(a_{t_i}|s_{t_i})}{\pi(a_{t_i}|s_{t_{i-1}})} relu(1 - p_L(a_{t_{i-1}}|s_{t_i})\tau_1) \cdot \pi(a_{t_i}|s_{t_{i-1}}) \\ & + p_4(a_{t_i}|s_{t_{i-1}}, s_{t_i}) \sum_{a_{t_{i-1}} \in \mathbb{A}} min(p_L(a_{t_{i-1}}|s_{t_i})\tau_1, 1) \\ & \cdot min(\pi(a_{t_{i-1}}|s_{t_i}), \pi(a_{t_{i-1}}|s_{t_{i-1}})) \\ =& \pi(a_{t_i}|s_{t_i}) relu(1 - p_L(a_{t_{i-1}}|s_{t_i})\tau_1) \\ & + \pi(a_{t_{i-1}}|s_{t_i}) min(p_L(a_{t_{i-1}}|s_{t_i}), 1) \\ =& \pi(a_{t_i}|s_0^{t_i}) \end{aligned} \tag{11}$$

(ii) Else if $\pi(a_{t_i}|s_{t_i}) \geq \pi(a_{t_i}|s_{t_{i-1}})$

$$\begin{aligned}
&p(a_{t_i}|s_0^{t_i}) \\
=&\pi(a_{t_i}|s_{t_{i-1}})relu(1 - p_L(a_{t_{i-1}}|s_{t_i})\tau_1) \\
&+ p_3(a_{t_i}|s_{t_{i-1}}, s_{t_i}) \sum_{a_{t_{i-1}} \in \mathbb{A}} relu(\pi(a_{t_{i-1}}|s_{t_{i-1}}) - \pi(a_{t_{i-1}}|s_{t_i})) \\
&+ min(p_L(a_{t_i}, s_{t_i})\tau_1, 1)\pi(a_{t_i}|s_{t_{i-1}}) \\
=&\pi(a_{t_i}|s_{t_{i-1}})relu(1 - p_L(a_{t_{i-1}}|s_{t_i})\tau_1) \\
&+ relu(\pi(a_{t_i}|s_{t_i}) - \pi(a_{t_i}|s_{t_{i-1}})) \\
&+ min(p_L(a_{t_i}, s_{t_i})\tau_1, 1)\pi(a_{t_i}|s_{t_{i-1}}) \\
=&\pi(a_{t_i}|s_{t_i})
\end{aligned} \tag{12}$$

□

B Proof of Theorem 2

Proof. If the agent decisions at time sequence $t_1, t_2, \ldots, t_n$, the expected number of action changes from time 0 to time T is as shown in Equation (13).

$$\begin{aligned}
&C(s_0^T, t_1, t_2, \ldots, t_n) \\
=&\sum_{i=1}^{n} \sum_{a \in \mathbb{A}} p(a_{t_i}|s_0^{t_{i-1}}) \Big(relu(1 - \frac{\pi(a|s_{t_i})}{\pi(a|s_{t_{i-1}})}) \\
&+ min(\frac{\pi(a|s_{t_i})}{\pi(a|s_{t_{i-1}})}, 1) min(p_L(a_{t_i}, s_{t_i})\tau_{i-1}, 1) \Big) \\
=&C_1(s_0^T, t_1, t_2, \ldots, t_n) + C_2(s_0^T, t_1, t_2, \ldots, t_n)
\end{aligned} \tag{13}$$

$$\begin{aligned}
&C_1(s_0^T, t_1, t_2, \ldots, t_n) \\
=&\sum_{i=1}^{n} \sum_{a \in \mathbb{A}} relu(\pi(a|s_{t_{i-1}}) - \pi(a|s_{t_i})) \\
&C_2(s_0^T, t_1, t_2, \ldots, t_n) \\
=&\sum_{i=1}^{n} \sum_{a \in \mathbb{A}} \Big(min(\pi(a|s_{t_i}), \pi(a|s_{t_{i-1}})) \\
&\cdot min(p_L(a_{t_2}, s_{t_2})\tau_1, 1) \Big)
\end{aligned} \tag{14}$$

Obviously, C_2 is convergent, then consider C_1: Because $\pi(a|s_t)$ is Lipschitz continuous, $\exists M s.t. \frac{|\pi(a|s_{t_i}) - \pi(a|s_{t_{i+1}})|}{|t_i - t_{i+1}|} < M$. For two time sequences $t_1, t_2, \ldots, t_n$, and $t'_1, t'_2, \ldots, t'_m$, we can prove the Equation (15) according to Equation (14).

$$C_1(s_0^T, t_1, \ldots, t_n) - C_1(s_0^T, t'_1, \ldots, t'_m) < \Delta t \cdot n \cdot M \tag{15}$$

where Δt is the max interval of $t'_1, \ldots, t'_m$.

Let $\{t^*[n]\}$ is a arithmetic sequence with length n in Equation (16). For the sequences in Equation (17), because it is incremental and bounded, it is convergent. Assume the limit of this sequences is C^*.

$$\{t^*[n]\} = \frac{1T}{n}, \frac{2T}{n}, \ldots, T \tag{16}$$

$$C_1(s_0^T, \{t^*[1]\}), C_1(s_0^T, \{t^*[2]\}) \ldots C_1(s_0^T, \{t^*[2^i]\}) \ldots \tag{17}$$

Because the sequences in Equation (17) convergence to C_1^*. $\forall \varepsilon > 0$, $\exists n_1$ s.t. $C_1^* - C_1(s_0^T, \{t^*[2^{n_1}]\}) < \frac{\varepsilon}{2}$. Let $\Delta t_1 = \frac{\varepsilon}{2 \cdot 2^{n_1} \cdot M}$. If the time intervals of time sequence is less than Δt_1, we can get Equation (18).

$$\begin{aligned}
C^* >&C(s_0^T, t_1, t_2, \ldots, t_n) \\
>&C(s_0^T, \{t^*[2^{n_1}]\}) - \Delta t_1 cdot 2^{n_1} \cdot M \\
>&C^* - \varepsilon
\end{aligned} \tag{18}$$

Therefore, $C_1(s_0^T, t_1, t_2, \ldots, t_n)$ converges to C_1^*.

□

References

[Bacon *et al.*, 2017] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[Doya, 2000] Kenji Doya. Reinforcement learning in continuous time and space. *Neural computation*, 12(1):219–245, 2000.

[Du *et al.*, 2020] Jianzhun Du, Joseph Futoma, and Finale Doshi-Velez. Model-based reinforcement learning for semi-markov decision processes with neural odes. *Advances in Neural Information Processing Systems*, 33:19805–19816, 2020.

[Eysenbach *et al.*, 2018] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

[He *et al.*, 2019] Shuping He, Haiyang Fang, Maoguang Zhang, Fei Liu, and Zhengtao Ding. Adaptive optimal control for a class of nonlinear systems: the online policy iteration approach. *IEEE transactions on neural networks and learning systems*, 31(2):549–558, 2019.

[Li *et al.*, 2019] Siyuan Li, Rui Wang, Minxue Tang, and Chongjie Zhang. Hierarchical reinforcement learning with advantage-based auxiliary rewards. *arXiv preprint arXiv:1910.04450*, 2019.

[Modares *et al.*, 2015] Hamidreza Modares, Frank L Lewis, and Zhong-Ping Jiang. Tracking control of completely unknown continuous-time systems via off-policy reinforcement learning. *IEEE transactions on neural networks and learning systems*, 26(10):2550–2562, 2015.

[Rashid *et al.*, 2018] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International conference on machine learning*, pages 4295–4304. PMLR, 2018.

[Samvelyan *et al.*, 2019] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

[Schulman *et al.*, 2015] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[Sunehag *et al.*, 2017] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

[Usunier *et al.*, 2016] Nicolas Usunier, Gabriel Synnaeve, Zeming Lin, and Soumith Chintala. Episodic exploration for deep deterministic policies: An application to starcraft micromanagement tasks. *arXiv preprint arXiv:1609.02993*, 2016.

[Vamvoudakis and Lewis, 2010] Kyriakos G Vamvoudakis and Frank L Lewis. Online actor–critic algorithm to solve the continuous-time infinite horizon optimal control problem. *Automatica*, 46(5):878–888, 2010.

[Vezhnevets *et al.*, 2017] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pages 3540–3549. PMLR, 2017.

[Vinyals *et al.*, 2017] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017.

[Vrabie *et al.*, 2009] Draguna Vrabie, O Pastravanu, Murad Abu-Khalaf, and Frank L Lewis. Adaptive optimal control for continuous-time linear systems based on policy iteration. *Automatica*, 45(2):477–484, 2009.

[Wang *et al.*, 2020] Haoran Wang, Thaleia Zariphopoulou, and Xun Yu Zhou. Reinforcement learning in continuous time and space: a stochastic control approach. *J. Mach. Learn. Res.*, 21(198):1–34, 2020.

[Yu *et al.*, 2021] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative, multi-agent games. *arXiv preprint arXiv:2103.01955*, 2021.