Neural Processing Letters (2024) 56:247
https://doi.org/10.1007/s11063-024-11705-x

Policy Optimization Algorithm with Activation Likelihood-Ratio for Multi-agent Reinforcement Learning

Lu Jia[1] · Binglin Su[2] · Du Xu[2] · Yewei Wang[2] · Jing Fang[2] · Jun Wang[3]

Accepted: 5 November 2024 / Published online: 25 November 2024

Abstract

As a ubiquitous on-policy reinforcement learning algorithm, proximal policy optimization (PPO) has achieved the state-of-the-art performance in both single-agent and cooperative multi-agent scenarios. However, it still suffers from the instability and inefficiency of the policy optimization with the non-strictly restricted likelihood-ratio in clipping strategy. In this work, we propose an activation likelihood-ratio (ALR) for solving this issue. The ALR is restricted by a tanh activation function, and it can be employed in multiple functional clipping strategies. The resulted ALR clipping strategy produces a smooth but precipitous objective curve, which can provide high policy update stationarity and efficiency. The ALR clipping strategy is incorporated into the PPO loss function, thus resulting in the method proximal policy optimization with activation likelihood-ratio (PPO-ALR). The rationality and superiority of the ALR-based target function are proved and analyzed. Moreover, experiments on the Pistonball cooperative multi-agent game show that PPO-ALR produces competitive and superior results compared with the standard PPO, PPO with rollback, and PPO smoothed algorithms, especially its high efficiency and success probability in searching optimal policies in multi-agent environments.

✉ Lu Jia
lujia_1028@hotmail.com

Binglin Su
subinglin@mail.hfut.edu.cn

Du Xu
xudu@mail.hfut.edu.cn

Yewei Wang
wangyewei@mail.hfut.edu.cn

Jing Fang
fangjing@hfut.edu.cn

Jun Wang
36110@qzc.edu.cn

[1] College of Physics and Electronic Engineering, Taishan University, Tai'an, China

[2] School of Computer and Information, Hefei University of Technology, Hefei, China

[3] School of Information Engineering, Huangshan University, Huangshan, China

Keywords Proximal policy optimization (PPO) · Multi-agent reinforcement learning (MARL) · Activation likelihood-ratio (ALR) · Clipping mechanism

1 Introduction

Recently, significant advancements have been made in multi-agent reinforcement learning (MARL). The DeepMind team's AlphaStar reached the human professional-level in StarCraft II [1]. OpenAI Five designed by Berner et al. [2] was able to defeat world champions in Dota2. Ye et al. [3] improved a deep reinforcement learning algorithm to realize complex game control in MOBA mobile games such as Honor of Kings. Smit et al. [4] trained agents in a 2D football simulator with 22 players learning to defeat a variety of handcrafted strategies. In addition, both multi-robot cooperation and multi-robot navigation have been successfully applied in military, industrial, medical and other fields with the key technology of multi-agent cooperative control.

Despite these advancements, challenges still exist when migrating the RL from single-agent to multi-agent environments. With an increasing number of agents, the complexity of the environment escalates [5, 6], rendering many RL algorithms designed for single-agent scenarios less effective. Several key challenges emerge: (1) explosive increase in dimension. As the number of agents increases, the dimensionality of the action-state space expands exponentially, greatly complicating the precision of model fitting. (2) Low generalization of the model. The data of RL is generated through the interactions between agents and environments, which limits the agents to learn effective strategies only in specific environments. Transitioning from one specific environment to another, especially in complex tasks, poses difficulties. This challenge often results in poor model generalization, which may be further deteriorated in complex multi-agent environments. (3) Non-stationary state problem. The non-stationary state of a multi-agent environment poses significant challenges in MARL. When multiple agents interact with the environment simultaneously, each agent's actions cause significant interference to other agents' policies, making the environment keep staying on a non-stationary state.

Multi-agent reinforcement learning can be divided into three categories. (1) Fully Centralized: each agent sends all parameters obtained by itself, including its observations, actions and rewards, to the central controller without taking any policy, and executes commands according to the feedback of the central controller. (2) Fully Decentralized: in contrast to the fully centralized MARL algorithm, agents in fully decentralized MARL algorithms operate independently without communicating with each other, each devising its policies. (3) Centralized Learning & Decentralized Execution: each agent maintains its independent actor network, and the central controller contains value networks that correspond to each agent. The central controller obtains observations, actions and rewards from the agents and learns policies, and agents execute their actions according to their own partial observation spaces.

From an optimization standpoint, MARL methods can be divided into value-based and policy-based methods. Value-based algorithms such as VDN [7], QMIX [8] and QTRAN [9] employ value-decomposition operations to handle non-stationarity in MARL. However, they struggle to learn optimal policies with large action spaces of agents. On the other hand, policy-based algorithms excel in efficiently handling large action spaces. For example, MADDPG [10] extends the deep deterministic policy gradient algorithm [11] to the multi-agent tasks, while COMA [12] utilizes a hypothetical counterfactual baseline to address the non-stationary state problem in MARL.

Proximal policy optimization [13], a popular on-policy algorithm in single agent reinforcement learning, encounters challenges when applied to multi-agent environments. Reference [14] comprehensively analyzes factors influencing the practical performance of PPO and offers insights into extending PPO to multi-agent tasks. Although PPO algorithm has made significant progress in various kinds of complex tasks, it grapples with instability in policy updating and data sampling inefficiency. This instability stems from the quality and characteristics of the loss function, limiting policy training stability. In order to better solve this problem, improved PPO variants, such as PPO with rollback (PPO-RB), PPO smoothed algorithm (PPOS), have been proposed. PPO-RB [15] adopts a rollback mechanism in the clipping function of PPO to restrict and benefit the policy learning. PPOS [16] ensures the convergence of the clipping function by refining the rollback mechanism with a functional clipping approach. Both PPO-RB and PPOS highlight the role of the clipping function in policy learning, enhancing its capacity to restrict policies. However, the instability during policy optimization still exists, leaving room for improvement in policy research efficiency, particularly when the likelihood-ratio approaches the clipping range.

In response to these challenges, we propose a novel enhancement to the PPO algorithm, hereby referred to as PPO with an activated likelihood-ratio (PPO-ALR). The likelihood-ratio is activated to enhance the PPO clipping function and objective function, improving the smoothness and steepness of the objective curve. This enhancement is driven by the overarching goal of enhancing policy research stability and efficiency in multi-agent environments.

In essence, our research is motivated to overcome the inherent challenges of PPO in multi-agent settings and to offer a more robust and effective solution for policy optimization in complex, interactive environments. The main contributions of this paper are as follows:

- We introduce ALR, an activated likelihood-ratio, to enhance the PPO clipping function and objective function, resulting in PPO-ALR. ALR enhances the smoothness and steepness of the objective curve, leading to improved policy research stability.
- We analyze the advantages of PPO-ALR, providing mathematical proof to validate its theoretical feasibility.
- We evaluate the PPO-ALR algorithm in the cooperative multi-agent game Pistonball, demonstrating its ability to learn more stable policies.

In Sect. 2, the factors that affect the efficiency of PPO in MARL are analyzed. In Sect. 3, a series of on-policy RL algorithms are introduced, including PPO-RB and PPOS. Adopting these methods to the MARL environment is one of our objectives. In Sect. 4, we describe the proposed PPO-ALR method, and provide a mathematical proof to verify the theoretical feasibility of it. In Sect. 5, PPO-ALR is tested on the Pistonball cooperative multi-agent game. The number of the agents is adjusted to control the environment's complexity, and the hyperparameter for clipping is tuned to verify the robustness of the improvements.

2 Related Work

In a cooperative multi-agent task, agents share a common goal. Therefore, the interactions between agents should be taken into account, and optimal behaviors of agents should be selected to guarantee a global convergence. However, because of the incompleteness of the agents' observations and the dynamics of environments, nonstationary state occurs. Moreover, many difficulties exist when PPO series algorithms are applied to multi-agent environments. In this part, the lazy agent dilemma, which is a cause and effect of the non-stationary

predicament, along with the factors influencing the performance of PPO in multi-agent tasks (multi-agent PPO, MAPPO), are emphasized.

2.1 Lazy Agent

Sunehag et al. [10] proposes the concepts of "spurious rewards" and "lazy agents". In a centralized multi-agent environment, each agent can only obtain a partial observation space from itself, and the dynamic changes of its local environment make agents face a nonstationary predicament. Then some agents become lazy because they may receive spurious reward signals from other agents, hindering the effective learning of their strategies. With the increasing number of agents in the environment, the nonstationary state becomes even more pronounced. Therefore, simply employing single-agent RL algorithms in multi-agent environments does not work, necessitating the need for specific adjustments.

2.2 Influential Factors to PPO's Performance

In early researches, on-policy RL algorithms are proved to have lower learning efficiency than off-policy RL algorithms in multi-agent tasks. Reference [17] reports that the performance of policy gradient (PG) algorithms such as COMA are significantly overtaken by MADDPG and QMix. Nevertheless, [18] shows that independent PPO still can achieve high success rates in hard StarCraft Multi-Agent Challenge [19]. Moreover, it has been analyzed that PPO can achieve high performance by adopting some measures, such as parameters sharing, value normalization, data sampling and PPO clipping [14].

2.2.1 Parameters Sharing

Parameters sharing, that is, training a single network in which all agents share the same parameters. This approach is particularly applicable when dealing with homogeneous agents, which implies that these agents share identical observation and action spaces. In this work, we only focus on scenarios with homogeneous agents. Literature [20] has demonstrated that a MARL task which contains homogeneous agents sharing parameters could have high learning efficiency. In MARL works, parameter sharing has been widely adopted [21] and [22].

2.2.2 Value Normalization

In the training process of MAPPO, wide difference of the overall returns may lead to drastic change of the output of the value networks, which may cause instability in value learning [14]. This instability can, in turn, disrupt the training process. By calculating the generalized advantage estimate (GAE), the mean value and standard deviation of the advantage function can be employed to normalize the value outputs. Then the instability during training can be moderated appropriately.

2.2.3 Data Sampling

In single-agent RL, an important feature of PPO is to split data into several mini-batches when sampling. However, recent study has shown that too many mini-batches may result

in sample reuse frequently and degrading the performance of PPO with multi-agents [14]. Among the suggestions of [14], using fewer epochs can limit changes of agents' policies and improve the stability of policy and value learning.

2.2.4 PPO Clipping

A major feature of PPO is its adoption of a clipping strategy to restrain drastic changes in policies and value functions. The clipping range and strength are controlled by a hyper-parameter ε. It has been verified that policy and value clipping can limit the non-stationarity and complexity during training. Principle and construction of the clipping function are described in more detail in in Sect. 3.3.

3 Preliminaries

An RL framework mainly consists of two parts: agents and environment. An agent starts from the state s_t and performs an action a_t at time t. According to the agent's action, the environment returns a reward r_t and a new state s_{t+1} to the agent. Such process is generally described as a Markov decision process (MDP). A MDP is defined as a tuple $(S,\ A,\ p,\ r,\ p_0,\ \gamma)$, where $S = \{s_t,\ t = 1,\ 2,\ ...\}$ and $A = \{a_t,\ t = 1,\ 2,\ ...\}$ are finite sets of the states and actions. $p(s_{t+1}|s_t,\ a_t)$ is the distribution of the state transition probability. $r(s_t,\ a_t)$ is the reward when the agent executes action a_t in state s_t. p_0 is the initial state distribution of the agent, and $\gamma \in (0,\ 1)$ is the discount factor. A stable policy π is selected by maximizing the long-term expected reward $\eta(\pi) = \mathbb{E}_{\rho_\pi(s)}\pi(a|s)[Q^\pi(s,\ a)]$. $\rho_\pi(s)$ is the state distribution and $Q^\pi(s,\ a)$ is the state-action value function under policy π [23].

3.1 Policy Gradients

Policy gradients methods [24] update the policy by maximizing the following surrogate performance objective,

$$L^{PG}_{\pi_{old}}(\pi) = \mathrm{E}\left[r_{s,a}(\pi)A^{\pi_{old}}_{s,a}\right] + \eta(\pi_{old}) \tag{1}$$

where $r_{s,a}(\pi) = \pi(a|s)\big/\pi_{old}(a|s)$ is the likelihood ratio between the new policy π and the old policy π_{old}. $A^{\pi_{old}}_{s,a} = Q^{\pi_{old}}(s,\ a) - V^{\pi_{old}}(s)$ is the advantage function under the old policy π_{old}, which represents the best action chosen for an agent at a special state. $V^\pi(s)$ is the state value function under policy π.

3.2 Trust Region Policy Optimization

PG algorithm uses the samples of the old policy π_{old} to update the current policy π at each iteration. However, instability and inaccuracy may exist in the update process. Trust region policy optimization (TRPO) algorithm [23] avoids excessive change of policy parameters in each iteration by limiting the KL divergence between π and π_{old}, and thus preventing the update from being unstable and unsuitable. The objective of TRPO is expressed as:

$$\max_{\pi} \mathrm{E}\left[r_{s,a}(\pi)A^{\pi_{old}}_{s,a} - \beta \mathrm{KL}(\pi_{old}(\cdot|s),\ \pi(\cdot|s))\right] \tag{2}$$

where β is the penalty parameter. "$\cdot$" means any action a.

3.3 Proximal Policy Optimization

Although TRPO algorithm does restrict the policy, the KL divergence constraint is computationally inefficient. PPO algorithm [13] adopts clipped likelihood ratio, which not only simplifies the calculation, but also possesses better sample complexity and applicability. The surrogate objective with the clipping function is

$$L^{\mathrm{CLIP}}(\pi) = \mathrm{E}\big[\min\big(r_{s,a}(\pi)A_{s,a}^{\pi_{old}},\ \mathrm{clip}\big(r_{s,a}(\pi),\ 1-\varepsilon,\ 1+\varepsilon\big)A_{s,a}^{\pi_{old}}\big)\big] \tag{3}$$

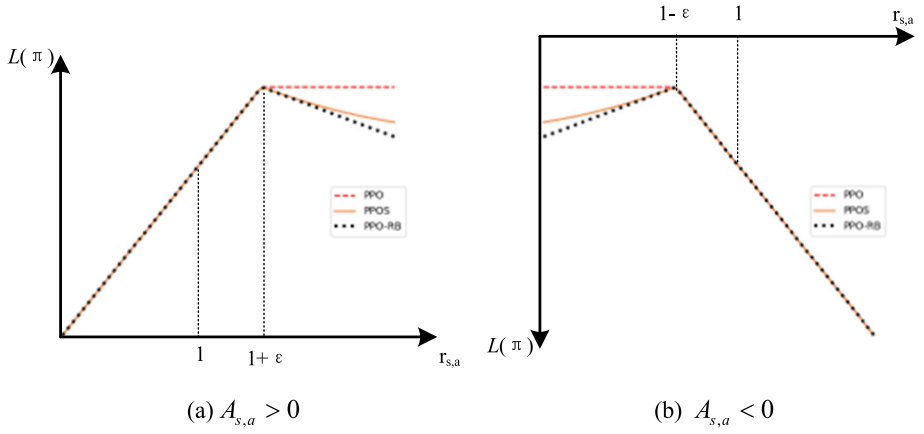
where $\varepsilon \in (0,\ 1)$ is a hyperparameter to control the clipping range $(1-\varepsilon,\ 1+\varepsilon)$. For PPO, the target function can be expressed in a more general form,

$$L^{\mathrm{PPO}}(\pi) = \mathrm{E}\big[\min\big(r_{s,a}(\pi)A_{s,a}^{\pi_{old}},\ F^{\mathrm{PPO}}\big(r_{s,a}(\pi),\ \varepsilon\big)A_{s,a}^{\pi_{old}}\big)\big] \tag{4}$$

where F^{PPO} is a clipping function

$$F^{\mathrm{PPO}}\big(r_{s,a}(\pi),\ \varepsilon\big) = \begin{cases} 1-\varepsilon & r_{s,a}(\pi) \le 1-\varepsilon \\ 1+\varepsilon & r_{s,a}(\pi) \ge 1+\varepsilon \\ r_{s,a}(\pi) & \text{otherwise} \end{cases} \tag{5}$$

For PPO, the policy is updated by the gradient of the surrogate objective shown in Eq. (4). The clipping function indicates that PPO tries to remove the incentive for moving the ratio $r_{s,a}(\pi)$ outside the clipping range $(1-\varepsilon,\ 1+\varepsilon)$, and then restricts the policy update. The minimization operation makes the final objective become a lower bound on the unclipped objective, and keeps the gradients. The relationship curve of the PPO surrogate objective function $L(\pi)$ varied with the likelihood ratio $r_{s,a}(\pi)$ at one term is shown as the red dashed line in Fig. 1. When the ratio goes past the clipping range, the slope of the curve become zero. PPO obtains superior and reliable performances in many high-dimensional single-agent RL tasks [13] and the hyperparameter ε is usually selected by experience [25, 26].



(a) $A_{s,a} > 0$ (b) $A_{s,a} < 0$

Fig. 1 Relationship curves that the objective functions $L(\pi) \hat{=} L^{\mathrm{PPO}}(\pi)$, $L^{\mathrm{PPOS}}(\pi)$ or $L^{\mathrm{PPO\text{-}RB}}(\pi)$ varies with the likelihood ratio $r_{s,a}$ of PPO, PPOS and PPO-RB for positive advantages (left) and negative advantages (right). All plots start from $r = 1$ for optimization. Slopes of the curves outside the clipping range ($r > 1+\varepsilon$, $A > 0$ or $r < 1-\varepsilon$, $A < 0$) show the differences among PPO, PPO-RB and PPOS

3.4 PPO with Rollback

Even though PPO restricts the policy update by setting a clipping mechanism, the likelihood ratio cannot be strictly bounded in the clipping range, and the incentive for pushing the policy away from the old one is not effectively removed [15]. To solve this problem, a rollback mechanism is introduced in the clipping function, and the PPO with rollback (PPO-RB) algorithm is proposed. The objective of PPO-RB is

$$L^{\text{PPO - RB}}(\pi) = \mathrm{E}\big[\min\big(r_{s,a}(\pi)A^{\pi_{old}}_{s,a},\ F^{\text{PPO - RB}}\big(r_{s,a}(\pi),\ \varepsilon,\ \alpha\big)A^{\pi_{old}}_{s,a}\big)\big] \tag{6}$$

The clipping function of PPO-RB is defined as

$$F^{\text{PPO - RB}}\big(r_{s,a}(\pi),\ \varepsilon,\ \alpha\big) = \begin{cases} -\alpha r_{s,a}(\pi) + (1+\alpha)(1-\varepsilon) & r_{s,a}(\pi) \le 1-\varepsilon \\ -\alpha r_{s,a}(\pi) + (1+\alpha)(1+\varepsilon) & r_{s,a}(\pi) \ge 1+\varepsilon \\ r_{s,a}(\pi) & \text{otherwise} \end{cases} \tag{7}$$

where $\alpha > 0$ is a hyperparameter that determines the force of the rollback operation. The black dashed line in Fig. 1 depicts the clipped objective with the roll-back operation. When the likelihood ratio is outside the clipping range, the slope is reversed for providing a negative incentive, and then the restriction on the likelihood ratio is enhanced.

3.5 Proximal Policy Optimization Smoothed Algorithm

Even though PPO-RB provides a distinct improvement over PPO, the rollback operation still has certain drawbacks: the convergence failure with extremely large ratio and the choice of α by experience. To solve these problems, the Proximal Policy Optimization Smoothed method (PPOS) is proposed [16], whose objective is defined with a functional clipping operation.

$$L^{\text{PPOS}}(\pi) = \mathrm{E}\big[\min\big(r_{s,a}(\pi)A^{\pi_{old}}_{s,a},\ F^{\text{PPOS}}\big(r_{s,a}(\pi),\ \varepsilon,\ \alpha\big)A^{\pi_{old}}_{s,a}\big)\big] \tag{8}$$

$$F^{\text{PPOS}}\big(r_{s,a}(\pi),\ \varepsilon,\ \alpha\big) = \begin{cases} -\alpha\tanh\big(r_{s,a}(\pi)-1\big)+1-\varepsilon-\alpha\tanh(\varepsilon) & r_{s,a}(\pi) \le 1-\varepsilon \\ -\alpha\tanh\big(r_{s,a}(\pi)-1\big)+1+\varepsilon+\alpha\tanh(\varepsilon) & r_{s,a}(\pi) \ge 1+\varepsilon \\ r_{s,a}(\pi) & \text{otherwise} \end{cases} \tag{9}$$

$\alpha > 0$ is a hyperparameter. The larger the α, the steeper the objective's slope is. The objective curve of PPOS is shown as the orange dashed line in Fig. 1. The hyperbolic tangent activation function bring PPOS's objective a smoothly changed slope. When the ratio is beyond the clipping range, the slope starts from a larger absolute value and gradually converges to zero. This smoothing mechanism ensures the convergence of the clipping function.

4 Method

4.1 PPO with Activation Likelihood Ratio (PPO-ALR)

The rationale behind introducing the ALR mechanism lies in the recognition of the importance of maintaining a balance between policy stability and update efficiency. The traditional PPO, PPO-RB, and PPOS methods make great efforts to restrict the likelihood ratio between

the old policy and new policy, which prevents the ratio from moving far away from the clipping range and guarantees the convergence property. However, the speed and stability of the policy update within the clipping range still need to be focused on, which determines the incentive for moving the likelihood ratio. This oversight impact the efficiency and effectiveness of policy learning. The ALR mechanism introduced in this paper aims to fill this gap by proactively controlling policy updates before the likelihood ratio exceeding the clipping range. Consequently, we present the PPO-ALR algorithm, which is built upon the principles of PPO but incorporates the ALR mechanism to enhance performance. Unlike PPO-RB and PPOS, which maintain a constant update speed for policy adjustments, PPO-ALR dynamically adjusts the update speed as the likelihood ratio approaches the boundaries of the clipping range. Additionally, the adoption of the hyperbolic tangent activation function in ALR strengthens the efficiency and stability of the policy update when the objective value is worse than the initial one, i.e., $r(\pi)A < r(\pi_{old})A$. In this way, PPO-ALR ensures policy update efficiency and stability. This adaptability allows PPO-ALR to respond more effectively to changes in the environment, ultimately leading to improved performance in complex multi-agent scenarios.

In PPO-ALR, the likelihood ratio is restricted by an activation tanh function when it is in the clipping range, and the ALR based clipping function is defined as

$$F^{\text{PPO - ALR}}\big(r_{s,a}(\pi),\, \varepsilon,\, \alpha\big) = \begin{cases} -\alpha \tanh\big(r_{s,a}(\pi) - 1\big) + 1 - \varepsilon - \alpha \tanh(\varepsilon) & r_{s,a}(\pi) \le 1 - \varepsilon \\ -\alpha \tanh\big(r_{s,a}(\pi) - 1\big) + 1 + \varepsilon + \alpha \tanh(\varepsilon) & r_{s,a}(\pi) \ge 1 + \varepsilon \\ \frac{\varepsilon}{\tanh(\varepsilon)} \tanh\big(r_{s,a}(\pi) - 1\big) + 1 & \text{otherwise} \end{cases} \tag{10}$$

where $\varepsilon \in (0,\, 1)$ and α are hyperparameters. α determines the scale of the clipping. The larger α is, the more precipitous the objective curve is. Within the clipping range, an activated function

$$f\big(r_{s,a}(\pi),\, \varepsilon\big) = \frac{\varepsilon}{\tanh(\varepsilon)} \tanh\big(r_{s,a}(\pi) - 1\big) + 1 \tag{11}$$

is defined to replace $r_{s,a}(\pi)$ to update the policy. The similar functional clipping as PPOS is adopted for preserving the convergence property.

The function $f\big(r_{s,a}(\pi),\, \varepsilon\big)$ is built on the hyperbolic tangent activation function. The hyperbolic tangent activation function is selected because it dynamically adjusts the change of the policy gradient in the process of policy optimization. Its output values falls between -1 and 1, which allows for effective control of gradient fluctuations, and thereby contributing to policy update stability. The hyperparameter ε determines the clipping range and it influences the gradient variation of $f\big(r_{s,a}(\pi),\, \varepsilon\big)$ within the clipping range. In existing literatures, ε is usually selected by experience and 0.2 is a frequently used value.

Moreover, the objective of PPO-ALR is modified with $f\big(r_{s,a}(\pi),\, \varepsilon\big)$ for preserving the curve continuity.

$$L^{\text{PPO - ALR}}(\pi) = \mathrm{E}\big[\min\big(f\big(r_{s,a}(\pi),\, \varepsilon\big)A_{s,a}^{\pi_{old}},\, F^{\text{PPO - ALR}}\big(r_{s,a}(\pi),\, \varepsilon,\, \alpha\big)A_{s,a}^{\pi_{old}}\big)\big] \tag{12}$$

Plot of $L^{\text{PPO - ALR}}(\pi)$ varied with $r_{s,a}(\pi)$ is showing in Fig. 2. When the ratio moves outside the predefined clipping range, the smoothing mechanism works, ensuring convergence of the clipping function, like PPOS. When $r_{s,a}(\pi)$ stays in the clipping range,

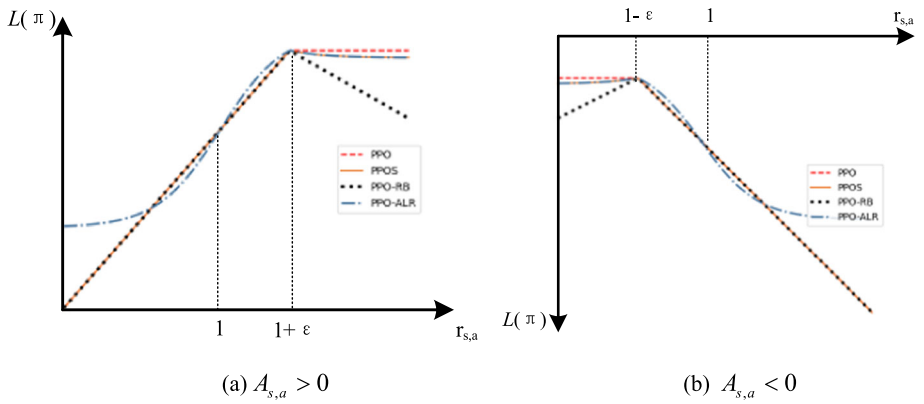


Fig. 2 Relationship curves that the PPO-ALR objective function $L(\pi) \hat{=} L^{\text{PPO - ALR}}(\pi)$ varies with the likelihood ratio $r_{s,a}$ for positive advantages (left) and negative advantages (right). r is restricted by the tanh function to produce a dynamically changed slope before it moves out of the clipping range

$L^{\text{PPO - ALR}}(\pi)$ shifts from the center point ($r_{s,a}(\pi) = 1$) to the clipping boundary. The gradient of $L^{\text{PPO - ALR}}(\pi)$ attains its maximum value at $r_{s,a}(\pi) = 1$, and progressively diminishes as it approaches the boundaries of the clipping range. The variation of the gradient mirrors the variation of the policy update speed. The large gradient at $r_{s,a}(\pi) = 1$ increases the efficiency in policy update, and the small gradients at the borders $r_{s,a}(\pi) = 1 - \varepsilon$ and $r_{s,a}(\pi) = 1 + \varepsilon$ prevent the likelihood ratio from straying outside the clipping range. This, in turn, enhances the stability of policy updates.

The special mechanism of ALR enables the gradient of $L^{\text{PPO - ALR}}(\pi)$ to stay in a dynamic state when the ratio is within the clipping range, and provides PPO–ALR with high policy search and update efficiency. Theoretically, the following properties should be satisfied to ensure the theoretical feasibility and superiority of $L^{\text{PPO - ALR}}(\pi)$ for policy optimization. Moreover, the theoretical feasibility efficiency of the proposed PPO-ALR's target function is proved as follows.

Properties

- $r_{s,a}(\pi) = 1$ *is the starting point of the policy research. When* $F\left(r_{s,a}(\pi), \varepsilon, \alpha\right)$ *is in the interval* $(1 - \varepsilon, 1 + \varepsilon)$*, the objective curve should be symmetric with respect to the center.*
- *Function* $F\left(r_{s,a}(\pi), \varepsilon, \alpha\right)$ *is continuous at* $r_{s,a}(\pi) = 1 - \varepsilon$ *and* $r_{s,a}(\pi) = 1 + \varepsilon$.
- *Function* $F^{PPO\text{ - }ALR}$ *rises or falls faster than* F^{PPO} *when* $r_{s,a}(\pi) = 1$*, and slower than* F^{PPO} *when the ratio moves close to the border of the clipping range.*

Proof

- Because of properties of the hyperbolic tangent activation function, it is easy to find that the function f (as shown in Eq. (11)) is centrosymmetric at $r_{s,a}(\pi) = 1$.
- The continuity of $F\left(r_{s,a}(\pi), \varepsilon, \alpha\right)$ is also obvious that

$$f = -\alpha \tanh(r_{s,a}(\pi) - 1) + 1 - \varepsilon - \alpha \tanh(\varepsilon) \text{ when } r_{s,a}(\pi) = 1 - \varepsilon;$$
$$f = -\alpha \tanh(r_{s,a}(\pi) - 1) + 1 + \varepsilon + \alpha \tanh(\varepsilon) \text{ when } r_{s,a}(\pi) = 1 + \varepsilon.$$

- Then we prove the third property.

We think of $r_{s,a}(\pi)$ as a diagonal line. First, we should prove that the gradient of f is maximum at the starting point, that is,

$$\nabla L^{\text{PPO - ALR}}(\pi) = \nabla \mathcal{F}^{\text{PPO - ALR}}\big(r_{s,a}(\pi),\, \varepsilon,\, \alpha\big) = \nabla f > 1 \tag{13}$$

when $r_{s,a}(\pi) = 1$. Similarly, we should prove that the slope of f turns to small when it is approaching the border of the clipping range, that is,

$$\nabla L^{\text{PPO - ARL}}(\pi) = \nabla \mathcal{F}^{\text{PPO - ARL}}\big(r_{s,a}(\pi),\, \varepsilon,\, \alpha\big) = \nabla f < 1 \tag{14}$$

when $r_{s,a}(\pi) \to 1+\varepsilon$, $A^{\pi_{old}}_{s,a} > 0$ and $r_{s,a}(\pi) \to 1-\varepsilon$, $A^{\pi_{old}}_{s,a} < 0$.

We consider the derivatives of f to prove Eqs. (13) and (14). The first derivative of f with respect to $r_{s,a}(\pi)$ is

$$f' = \frac{\varepsilon}{\tanh(\varepsilon)}\big(1 - \tanh^2\big(r_{s,a}(\pi) - 1\big)\big). \tag{15}$$

When $r_{s,a}(\pi) = 1$ and $\varepsilon \in (0,\, 1)$, $f' = \frac{\varepsilon}{\tanh(\varepsilon)} > 1$. Then $\nabla L^{\text{PPO - ARL}}(\pi) > 1$ which means that $F^{\text{PPO - ARL}}$ changes more rapidly than F^{PPO} when $r_{s,a}(\pi) = 1$.

Next, consider the second derivative of f.

$$f'' = \frac{\varepsilon}{\tanh(\varepsilon)}\big(2\tanh^3\big(r_{s,a}(\pi) - 1\big) - 2\tanh\big(r_{s,a}(\pi) - 1\big)\big) \tag{16}$$

$r_{s,a}(\pi) > 1$, $f'' < 0$; $r_{s,a}(\pi) < 1$,$f'' > 0$. It means that f' is monotonic increasing in interval $(1-\varepsilon,\, 1)$ and monotonic decreasing in interval $(1,\, 1+\epsilon)$. Since f' attains its maximum value when $r_{s,a}(\pi) = 1$, and f' decreases when it deviates from this point in either direction.

Utilizing the Lagrange finite increment theorem, we can establish that,$\exists \varphi \in (0,\, 1)$

$$f\big(1 + \Delta r_{s,a}(\pi)\big) - f(1) = f'\big(1 + \varphi \Delta r_{s,a}(\pi)\big)\Delta r_{s,a}(\pi). \tag{17}$$

When $\Delta r_{s,a}(\pi) = \varepsilon$, $f(1+\varepsilon) - f(1) = f'(1+\varphi\varepsilon)\varepsilon = \varepsilon$. Hence, $f'(1+\varphi\varepsilon) = 1$. Considering the monotone variation of f' in $(1,\, 1+\varepsilon)$, we get $f'(1+\varepsilon) < f'(1+\varphi\varepsilon) = 1$.

Similarly, we get

$$f(1) - f\big(1 - \Delta r_{s,a}(\pi)\big) = f'\big(1 - \varphi \Delta r_{s,a}(\pi)\big)\Delta r_{s,a}(\pi) \tag{18}$$

and $f'(1-\varepsilon) < f'(1-\varphi\varepsilon) = 1$.

To sum up, $\nabla L^{\text{PPO - ARL}}(\pi) = \nabla f < 1$. That is, $F^{\text{PPO - ARL}}$ changes slower than F^{PPO} when the ratio is close to the clipping boundaries.

These findings substantiate the effectiveness of the PPO-ALR algorithm by ensuring that policy updates are efficient and stable, aligning with the goals of the ALR mechanism.

4.2 Implementation of PPO-ALR in Multi-agent RL

To study the performance of PPO series algorithms in multi-agent environments, the decentralized partially observable Markov decision processes (Dec-POMDP) with shared rewards is proposed [27–29]. A Dec-POMDP is defined as a tuple $(S,\, A,\, O,\, P,\, R,\, p_0,\, \gamma)$, where S is the state space. A is the shared action space for each agent i.

$O = \left\{o_t^i,\ i = 1,\ ...,\ I; t = 1,\ 2,\ ...\right\}$ is the set of joint observations. o_t^i is the local observation available to agent i. $P\left(s'|s,\ A_t\right)$ denotes the transition probability from s to s' with the joint action $A_t = (a_t^i,\ ..,\ a_t^I)$ for all I agents. $R(s,\ A_t)$ is the shared reward function. p_0 is the distribution of the initial state. $\gamma \in (0,\ 1)$ is the discount factor [30].

A schematic representation of PPO-ALR in multi-agent scenarios at time t is shown in Fig. 3. During the process of training, PPO-ALR trains two independent neural networks: an actor network and a critic network. We assume that the agents are homogeneous and they share the actor and critic networks with the same parameter θ. For training the actor network π_θ, the following PPO-ALR objective is maximized.

$$L(\pi_\theta) = \mathrm{E}_t\left[\mathrm{E}_i\left[\min\left(f\left(r_t^i(\pi_\theta)\right)A_t^i,\ F^{\text{PPO - ARL}}\left(r_t^i(\pi_\theta),\ \varepsilon,\ \alpha\right)A_t^i\right)\right]\right]. \tag{19}$$

The critic network V_θ is trained by minimizing the following loss function.

$$L(V_\theta) = \mathrm{E}_t\left[\mathrm{E}_i\left[\max\left(\left(V_\theta\left(s_t^i\right) - R_t^i\right)^2,\ \left(\mathrm{clip}\left(V_\theta\left(s_t^i\right),\ V_{\theta_{old}}\left(s_t^i\right) - \varepsilon,\ V_{\theta_{old}}\left(s_t^i\right) + \varepsilon\right) - R_t^i\right)^2\right)\right]\right] \tag{20}$$

$f(\cdot)$ is the activated function defined in Eq. (11). In order to adapt to multi-agent tasks, the ALR based clipping function is defined as

$$F^{\text{PPO - ALR}}\left(r_t^i(\pi),\ \varepsilon,\ \alpha\right) = \begin{cases} -\alpha\tanh\left(r_t^i(\pi) - 1\right) + 1 - \varepsilon - \alpha\tanh(\epsilon) & r_t^i(\pi) \le 1 - \varepsilon \\ -\alpha\tanh\left(r_t^i(\pi) - 1\right) + 1 + \varepsilon + \alpha\tanh(\epsilon) & r_t^i(\pi) \ge 1 + \varepsilon \\ f\left(r_t^i(\pi),\ \varepsilon\right) & \text{otherwise} \end{cases}. \tag{21}$$

$r_t^i \sim r_{s,a}$ is the likelihood ratio for agent i at t. R_t^i is the discounted reward-to-go for agent i at t. $A_t^i \sim A_{s,a}$ is the advantage estimate for agent i at t.

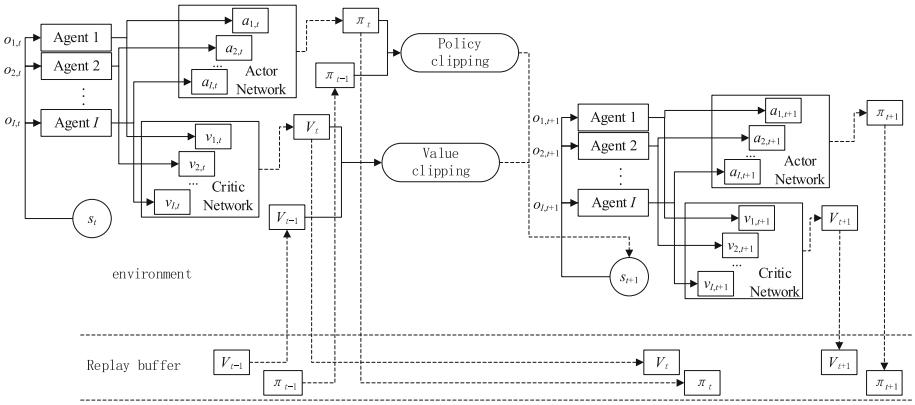


Fig. 3 A schematic representation of PPO-ALR in multi-agent scenarios at time t. At each time, agents in the environment receive their local observations, select actions, and together form the joint action. The joint action is sent to the environment which leads to a state transition

Fig. 4 PettingZoo: Pistonbal

$$R_t^i = \mathcal{A}_t^i + r_t^i \tag{22}$$

$$\mathcal{A}_t^i = -V_\theta\left(s_t^i\right) + r_t^i + \gamma r_{t+1}^i + \cdots + \gamma^{T-t+1} r_{T-1}^i + \gamma^{T-t} V_\theta\left(s_T^i\right) \tag{23}$$

T is the maximum timestep, and r_t^i is the reward of agent i at t.

Since the neural networks share the parameters θ, an entropy bonus is added to combine the policy surrogate and the value function error term [31]. Then we get the following objective which should be maximized at each iteration.

$$L(\theta) = \mathrm{E}[L(\pi_\theta) - c_1 L(V_\theta) + c_2 \mathcal{S}(\pi_\theta)] \tag{24}$$

c_1, c_2 are coefficients. $\mathcal{S}$ is the policy entropy. The training process of PPO-ALR for multi-agents is shown in Algorithm 1.

As shown in Fig. 3, during the process of training, all agents have an interaction with the environment at each time t. Agents receive their individual observations, which encapsulate both individual information and information pertaining to other agents or the overall environment. These observations are processed collectively by the networks with sharing parameters. Through policy networks and value networks, each agent's actions and received rewards are determined. Following the completion of a round of iteration, all agents acquire their respective policies. These policies, collectively forming a joint policy for all agents within the environment, are stored in the replay buffer for subsequent policy updates. The time complexity and space complexity of the process are $O(total\_episodes * T)$ and $O(T * I)$ respectively. T is the maximum timestep. I is the agent quantity.

5 Experiment

In this section, we present experiments to investigate whether PPO-ALR can achieve its goals in multi-agent environments. We set PPO series algorithms in Sect. 3 as baselines for comparison and reveal the competitiveness and superiority of the proposed method in tasks.

Table 1 Arguments in Pistonball environment

Arguments	Descriptions	Values
n pistons	The number of pistons (agents) in the environment	15, 20
Time penalty	Amount of rewards added to each piston at each time step	− 0.1
Continuous	Whether the action space of the agent is continuous or discrete	False
Random drop	Whether a random ball is in the initial position	False
Ball mass	The mass of the ball physics object	0.75
Ball friction	The friction of the ball physics object	0.3
Ball elasticity	The elasticity of the ball physics object	1.5
Max cycles	After how many steps all agents will return done	125

5.1 Environment

Performance of PPO-ALR is tested on the Pistonball game which is a special multi-agent environment in the PettingZoo library [32, 33]. We chose it because PettingZoo as a library of diverse sets of multi-agent environments has a universal, elegant Python API. Pistonball is a cooperative game (Fig. 4) with the chipmunk physics engine. In Pistonball, multiple pistons constitute the cooperative multi-agents. The goal of Pistonball is to push the ball on the right side of the interface to the left side by controlling the vertical movements of the pistons. The observation space of a piston agent is an RGB image composed of adjacent pistons and the space above them. The action space is a discrete tuple (0, 1, 2), where 0 means moving down, 1 staying still, and 2 moving up. The challenge in Pistonball game lies in orchestrating highly coordinated emergency behaviors among the pistons to ensure the ball's smooth traversal from right to left. For each agent, the reward is a combination of how much the ball moved left overall and how much the ball moved left if it is close to the current piston. Generally, a piston is considered to be close to the ball if it is directly below any part of the ball. So the reward for each agent can be expressed by

$$\lambda_{\text{local}} \cdot r_{\text{local}} + (1 - \lambda_{\text{local}}) \cdot r_{\text{global}}$$

where r_{local} is the local reward which is 0.5 times the change in the ball's displacement distance. r_{global} is the global reward which is the change in displacement distance divided by the starting position, times 100, and plus the hyperparameter of time penalty (default − 0.1). The ratio between r_{local} and r_{global} is expressed by λ_{local}. Arguments of the Pistonball environment are shown in Table 1.

Algorithm 1 PPO-ALR for multi-agent environments

Initialize θ, the shared parameters of the actor π and the critic V;
Initialize the replay buffer $D = \{\}$;
for $episode = 1,...,total\_episodes$ **do**
  Initialize $\left\{o_t^i, \forall i\right\}$, i is the number of the agent;
  Initialize the size of the batch T;
  Initialize a parameter $done = 1$ which is returned by the environment and marks the loop terminates;
  for $t = 1,...,T$ **do**
    for $i = 1,...,I$ **do**
    $a_t^i \sim p_t^i(a) = \pi\left(o_t^i;\theta\right)$;
    $v_t^i = V\left(o_t^i;\theta\right)$;
    end for
    Execute actions $\boldsymbol{A}_t = \left\{a_t^i, \forall i\right\}$,
    Observe $\boldsymbol{O}_t = \left\{o_{t+1}^i, \forall i\right\}$, $\boldsymbol{S}_t = \left\{s_{t+1}^i, \forall i\right\}$, $\boldsymbol{R}_t = \left\{r_t^i, \forall i\right\}$ and $done$;
    if $done = 0$ **then**
      $T = t$;
    end if
    $D_t = \left\{\boldsymbol{O}_t, \boldsymbol{S}_t, \boldsymbol{A}_t, \boldsymbol{R}_t\right\}$,
    $D = D \bigcup D_t$;
  end for
    Compute the advantage estimate $\hat{\boldsymbol{\mathcal{A}}} = \left\{\mathcal{A}_t^i, \forall i, t\right\}$ with equation (22),
    Compute the agents' reward-to-go $\hat{\boldsymbol{R}} = \left\{R_t^i, \forall i, t\right\}$ by equation (21),
    $D = D \bigcup \left\{\hat{\boldsymbol{\mathcal{A}}}, \hat{\boldsymbol{R}}\right\}$,
    Random all agent data from D,
    Split data into chunks of fixed length;
  for each data chunk in D **do**
    Update the actor network π_θ and the critic network V_θ;
  end for
  Combine π_θ and V_θ by the surrogate objective $L(\theta)$ in equation (23);
  Adam update θ on $L(\theta)$;
end for

5.2 Baselines

We evaluate the performance of the proposed method in four aspects: the proportion of the likelihood ratio shifting outside the clipping range, the average episode length, the average

episode return and the success rate during training. Moreover, we set the following baselines to test our algorithm.

- MAPPO: the standard PPO algorithm with the clipping range hyperparameter $\varepsilon = 0.2$ [13].
- MAPPO-RB: PPO with rollback operation, $\varepsilon = 0.2$, and the rollback strength parameter $\alpha = 0.3$[15].
- MAPPOS: PPO smoothed (PPOS) algorithm with $\varepsilon = 0.2$ and $\alpha = 0.3$[16].
- MP-MAPPO: PPO with two randomly initialized policy paths [34].
- MAPPO-ALR: the algorithm proposed in Sect. 4, which uses the same rollback operation as PPOS, $\varepsilon = 0.2$ and $\alpha = 0.3$.

In above five methods, 'MA' in front of the methods' names indicates that they are implemented in a multi-agent environment. Moreover, to further illustrate the effectiveness of the proposed method, we select a new algorithm, i.e. the Multi-Path Proximal Policy Optimization (MP-PPO) algorithm [34], to conduct a comparative experiment. MP-PPO is a method that begins with multiple randomly initialized policy paths and optimizes them concurrently during the policy optimization process. This approach aims to enhance policy diversity and mitigate the risk of premature convergence to suboptimal solutions. We test and compare the MP-PPO algorithm in the multi-agent environment and resulting a new baseline method MP-MAPPO. In MP-MAPPO, the number of the randomly initialized policy paths is 2, which is the optimal value selected based on experience.

As our study focuses on the enhancements to the clipping mechanism and policy objective, the MAPPO, MAPPO-RB and MAPPOS methods are particularly relevant for comparison. Their modifications align closely with our objectives, making them suitable benchmarks for evaluating the effectiveness of our proposed method.

In order to further verify the superiority of the activated likelihood ratio, we apply the improved ratio in Eq. (11) to modify the clipping function of PPO and PPO-RB, and compare them with the standard MAPPO and MAPPO-RB methods.

- MAPPO + ALR: the activated ratio in Eq. (11) is used to replace $r_{s,a}(\pi)$ in the clipping function of the standard PPO algorithm.
- MAPPO-RB + ALR: the activated ratio in Eq. (11) is used to replace $r_{s,a}(\pi)$ in the clipping function of the standard PPO algorithm.

5.3 Implementation Details

The Pistonball game in the butterfly environments on the PettingZoo platform version-2(v2) is tested in all experiments. Our network is based on the AC framework. The actor network and the critic network share the same network parameters. Agents process their input data using a convolutional neural network (CNN) consisting of three convolutional layers. These layers are configured with 32, 64, and 128 filters, respectively, each utilizing a 3×3 kernel size and padding of 1. The ReLU activation function is applied to the output of these layers. Following the CNN layers, a fully-connected network with a single hidden layer of size 512 is employed for further processing. Both the actor network and the critic network feature a solitary hidden layer with 512 neurons. The network is optimized by Adam with a learning rate of 0.001 and epsilon 10^{-5}. To facilitate learning, experiences encountered by agents during training are stored in a replay buffer with a batch size of 32. The learning rate γ is 0.99, and the parameters c_1 and c_2 in Eq. (24) are 0.1.

5.4 Training in Multi-agent environment

With the increasing number of agents, the complexity of the environment will increase exponentially. As the number of agents increases, the possible combinations of states in the environment grow exponentially, and interactions among multiple agents lead to a combinatorial explosion of state spaces, making the environment more complex. The actions that each agent takes may increase with the number of agents, resulting in an increase in the dimensionality of the joint action space, thereby increasing the difficulty of learning and decision-making for agents in the environment.

In order to show the performance of our method with different complexities, the experiments are carried out with the different number of agents. As the number of agents increases, the MAPPO, MAPPO-RB and MP-MAPPO algorithms tend to encounter learning failures earlier compared to scenarios with fewer agents. Experiments involving a smaller agent count, such as 10, demonstrate rapid convergence for all baseline methods, making it challenging to discern differences between the proposed method and the others. When the agent count exceeds 20, the effectiveness of all methods diminishes, indicating a noticeable decline in performance. Notably, there is a distinct performance gap between experiments with 15 and 20 agents. Therefore, to effectively showcase the results, we chose 15 agents to represent scenarios of low complexity and 20 agents for high complexity scenarios.

5.4.1 Effect on Policy Restriction

We use the proportion of the likelihood ratio shifting outside the clipping range to analyze the ability of the algorithms in restricting the policy update. The proportion is an average value of all agents in the policy search process at each episode. Figure 5 illustrates the proportions for five algorithms with varying numbers of agents. In the case of 15 agents, we observe that the proportions of MAPPO, MAPPO-RB, MAPPOS and MP-MAPPO rapidly approach zero, indicating swift convergence in their policy updates. However, overquick convergence may inadvertently reduce the diversity in policy update, which may potentially lead to suboptimal solutions (see results in Sec.5.4.2). In contrast, the proportion of MAPPO-ALR settles into

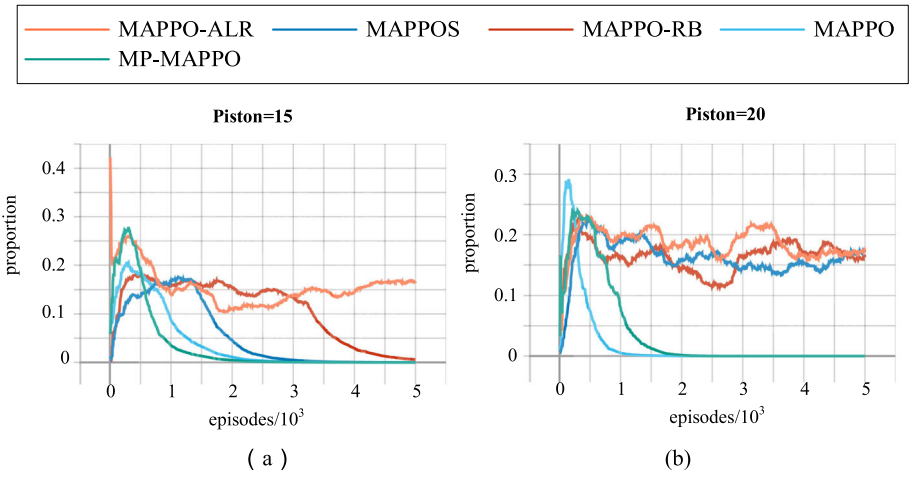


Fig. 5 The proportions of the likelihood ratio shifting outside the clipping range of four algorithms

a relatively stable value. The stable proportion ensures the preservation of the policy update diversity, ultimately contributing to superior results. When the agent count increases to 20, proportions of MAPPO-ALR, MAPPO-RB and MAPPOS turn to a stable value. In contrast, MAPPO and MP-MAPPO continues to rapidly converge to zero. These findings highlight the ability of MAPPO-ALR to strike a balance between the convergence speed and policy diversity, resulting in more efficient policy updates over different numbers of agents.

Our analysis indicates that maintaining a stable clipping ratio can mitigate the risk of converging to sub-optimal outcomes. By ensuring consistent clipping proportions, the algorithm can effectively balance policy exploration and policy optimization, facilitating more stable and efficient learning over time. Thus, maintaining a stable clipping proportion can effectively reduce the likelihood of converging to sub-optimal results.

5.4.2 Effect on Policy Performance

If all agents on one mission cannot make the ball reach the left of the global space before reaching the maximum stride length, the game will end directly. In the context of reinforcement learning, shorter episode lengths often indicate faster convergence. The agent is able to achieve its goal more efficiently within fewer steps. Therefore, we can judge whether the task has been successfully completed by the step size being used in one episode. Furthermore, the relationship between the stability and the success rate is analyzed. A stable algorithm may exhibit stable success rate, and fluctuations indicate instability or inefficiencies in learning. Changes in success rate indicate the learning dynamics of the algorithm, which could be influenced by factors such as exploration strategies, reward structures, or environmental complexities. By qualitatively analyzing the variations in success rate, the algorithm's stability can be inferred.

Figure 6 plots the average episode length, average episode return and success rate during training, respectively. Tables 2 and 3 show the numerical results after 5000 episodes.

In the case of low environmental complexity (i.e. 15 pistons), all methods demonstrate successful policy learning. The decline rates of the average episode length curves can be used to estimate the learning efficiency of algorithms. From Fig. 6a we can see that the proposed method exhibits the most rapid decline, indicating the highest learning efficiency. Figure 6c illustrates the average episode return, showcasing that MAPPO-ALR achieves the highest overall reward among the methods when dealing with 15 agents. Moreover, Fig. 6e highlights that MAPPO-ALR achieves the highest average success rate, nearing 50% after 5000 episodes in the case of 15 agents. But when it comes to a more complex environment (i.e. 20 pistons), Fig. 6b, d, f show that both MP-MAPPO, MAPPO-RB and MAPPO struggle to learn effective policies. After 5000 episodes, MAPPO-ALR's performance is on par with MAPPOS. In general, MAPPO-ALR demonstrates adaptability to complex environments (more agents). For a more detailed examination of the training processes, Fig. 7 presents the loss function curves defined in Eqs. (19), (20), and (23) for all five methods. These curves provide additional insights into the training dynamics.

By effectively balancing policy stability and update efficiency, the MAPPO-ALR method can adapt more efficiently to varying environmental conditions. Additionally, the MAPPO-ALR demonstrates robust adaptability to environmental complexity, especially when confronted with scenarios involving a higher number of agents. This adaptability is attributed to the ALR mechanism's ability to dynamically adjust policy updates based on the likelihood ratio's stability when it is in close proximity to the clipping boundaries, ensuring efficient learning across diverse scenarios.

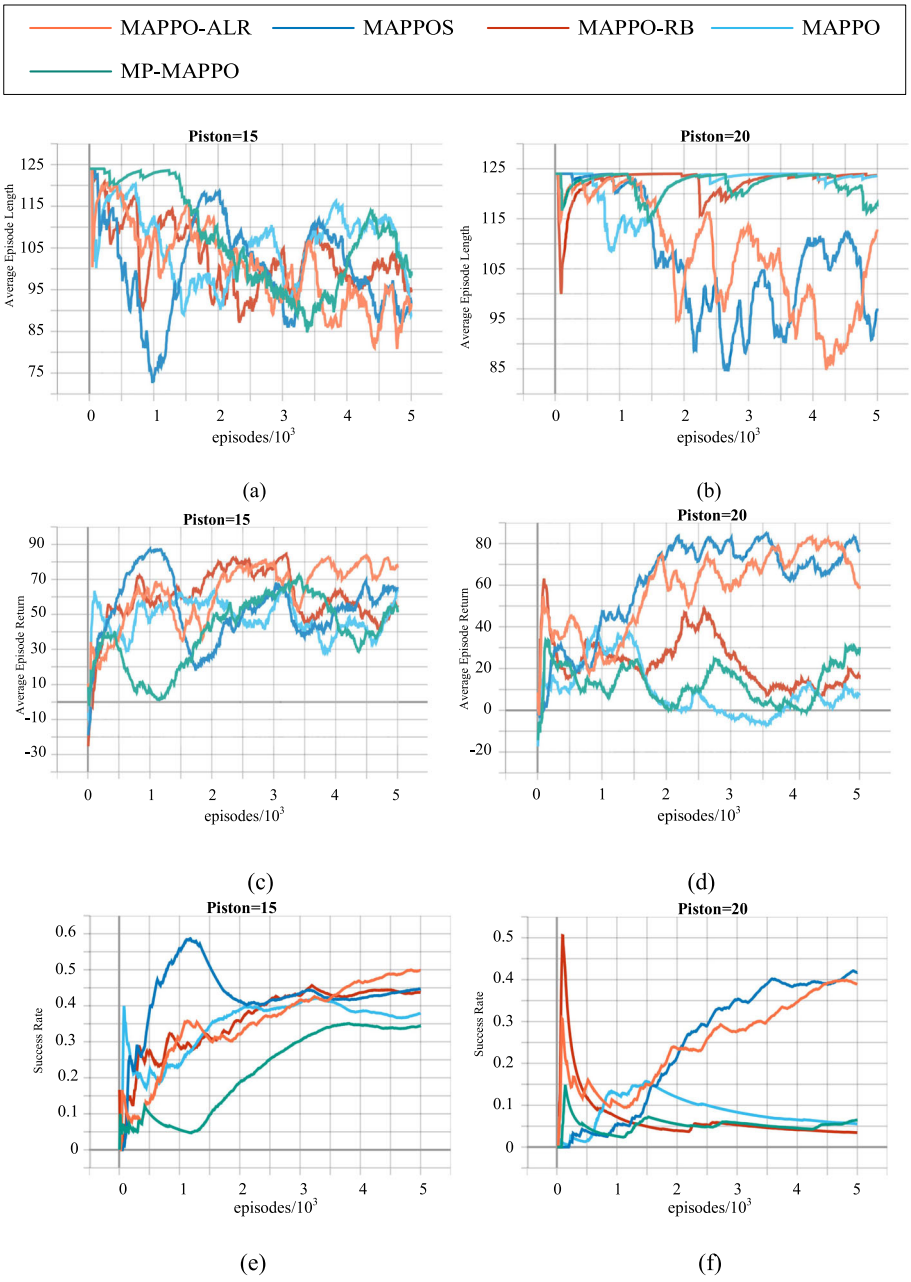


Fig. 6 Performance comparison of MAPPO, MAPPO-RB, MAPPOS, MP-MAPPO and MAPPO-ALR algorithms. Average Episode Length: the curves of the consumption step size per episode at the cases of 15 agents (**a**) and 20 agents (**b**). Average Episode Return: the curves of the overall return value per episode at the cases of 15 agents (**c**) and 20 agents (**d**). Success Rate: the curves of the average probability the missions succeed at the cases of 15 agents (**e**) and 20 agents (**f**)

Table 2 Numerical results of MAPPO, MAPPO-RB, MAPPOS, MP-MAPPO and MAPPO-ALR with 15 agents

	Average episode length	Average episode return	Success rate
MAPPO	103.9	48.26	0.3796
MAPPO-RB	101.2	59.82	0.4378
MAPPOS	99.15	53.35	0.4468
MP-MAPPO	98.34	52.11	0.3443
MAPPO-ALR	97.5	68.54	0.497

Table 3 Numerical results of MAPPO, MAPPO-RB, MAPPOS, MP-MAPPO and MAPPO-ALR with 20 agents

	Average episode length	Average episode return	Success rate
MAPPO	120	15.5	0.1074
MAPPO-RB	122.6	21.52	0.0346
MAPPOS	103.6	67.8	0.4172
MP-MAPPO	117.6	30.22	0.0647
MAPPO-ALR	106	62.71	0.388

5.4.3 Effect of the Activation Likelihood Ratio

In order to further substantiate the pivotal role of the activated likelihood ratio introduced in Eq. (11), we conducted experiments by applying the same activated likelihood ratio to MAPPO and MAPPO-RB, resulting in the MAPPO + ALR and MAPPO-RB + ALR algorithms. The modified clipping functions for MAPPO + ALR and MAPPO-RB + ALR are as follows. The results obtained from comparing MAPPO + ALR and MAPPO-RB + ALR with traditional MAPPO and MAPPO-RB are shown in Tables 4 and 5, as well as Fig. 8.

$$F^{\text{PPO + ALR}}\big(r_{s,a}(\pi),\,\varepsilon\big) = \begin{cases} 1-\varepsilon & r_{s,a}(\pi) \le 1-\varepsilon \\ 1+\varepsilon & r_{s,a}(\pi) \ge 1+\varepsilon \\ \frac{\varepsilon}{\tanh(\varepsilon)}\tanh\big(r_{s,a}(\pi)-1\big)+1 & \text{otherwise} \end{cases} \tag{25}$$

$$F^{\text{PPO - RB + ALR}}\big(r_{s,a}(\pi),\,\varepsilon,\,\alpha\big) = \begin{cases} -\alpha r_{s,a}(\pi) + (1+\alpha)(1-\varepsilon) & r_{s,a}(\pi) \le 1-\varepsilon \\ -\alpha r_{s,a}(\pi) + (1+\alpha)(1+\varepsilon) & r_{s,a}(\pi) \ge 1+\varepsilon \\ \frac{\varepsilon}{\tanh(\varepsilon)}\tanh\big(r_{s,a}(\pi)-1\big)+1 & \text{otherwise} \end{cases} \tag{26}$$

After 5000 episodes, it is evident that both MAPPO + ALR and MAPPO-RB + ALR outperform traditional MAPPO and MAPPO-RB. Specifically, MAPPO + ALR and MAPPO-RB + ALR exhibit shorter episode lengths, higher episode returns, and superior success rates. These findings demonstrate that the activated likelihood ratio plays a crucial role in enhancing the performance of clipping functions. Then the effectiveness and efficiency of the proposed RL algorithm are further confirmed.

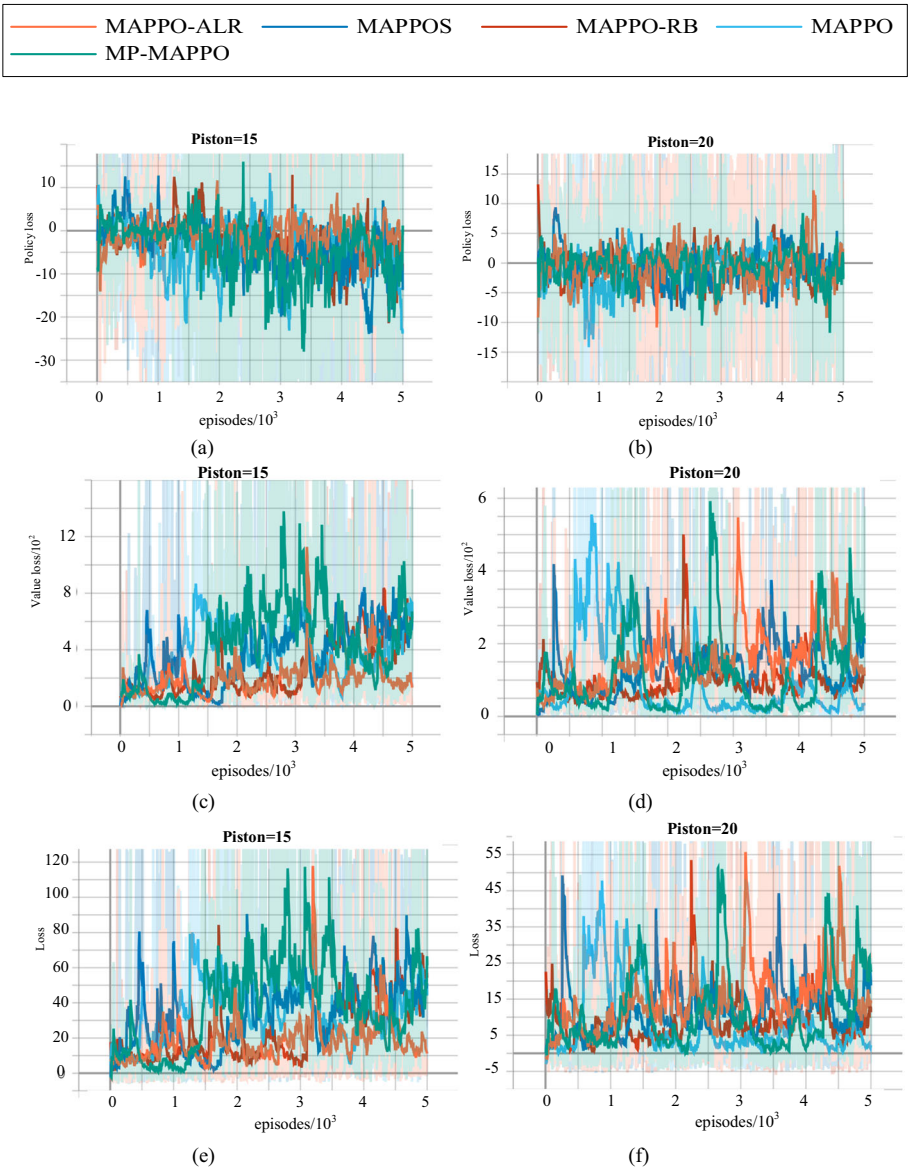


Fig. 7 Curves of loss functions in the training processes of MAPPO, MAPPO-RB, MAPPOS, MP-MAPPO and MAPPO-ALR algorithms. Policy loss: the curves of the loss function defined in Eq. (19) at the cases of 15 agents (**a**) and 20 agents (**b**). Value loss: the curves of the loss function defined in Eq. (20) at the cases of 15 agents (**c**) and 20 agents (**d**). Total loss: the curves of the loss function defined in Eq. (23) at the cases of 15 agents (**e**) and 20 agents (**f**)

Table 4 Numerical results of MAPPO, MAPPO-RB, MAPPO + ALR and MAPPO-RB + ALR with 15 agents

	Average episode length	Average episode return	Success rate
MAPPO	100.5	50.13	0.3796
MAPPO-RB	98.95	52.28	0.4378
MAPPO + ALR	94.63	57.87	0.3764
MAPPO-RB + ALR	96.64	76.66	0.4596

Table 5 Numerical results of MAPPO, MAPPO-RB, MAPPO + ALR and MAPPO-RB + ALR with 20 agents

	Average episode length	Average episode return	Success rate
MAPPO	122.2	4.354	0.0554
MAPPO-RB	122.9	17.18	0.0346
MAPPO + ALR	122.1	2.893	0.0636
MAPPO-RB + ALR	110.7	58.86	0.1968

5.4.4 Effect of the Clipping Parameter

As we have mentioned in Sect. 3.3.4, the clipping range and strength are controlled by a hyperparameter ε. We conduct experiments with varied hyperparameters to analyze their impact on the algorithm's performance. Our parameter selection is based on empirical evidence from prior literature. We do experiments with $\varepsilon = 0.05$, $\varepsilon = 0.2$, and $\varepsilon = 0.5$.

For 15 agents, the optimal parameter choice for ε is 0.05, when the complexity of the environment is at a low level. This parameter produces the fastest decline rates of the average episode length curves and yields the highest overall reward. However, in scenarios with high environmental complexity (20 agents), which is shown in Fig. 9b, d, f, MAPPO-ALR with $\varepsilon = 0.05$ fails to learn an optimal policy within a reasonable timeframe. $\varepsilon = 0.2$ produces a better result. It is because when the agents' policy undergoes less change with each update, learning speed is moderated.

In references, [35] holds the idea that PPO's performance depends heavily on the optimization trick, not on the core clipping mechanism. However, [15] draws a completely different conclusion. Although the clipping mechanism of PPO could not strictly restrict the likelihood ratio within the predefined clipping range, it could somewhat take effect on restricting the policy and benefit the policy learning. To further explore this aspect, we conduct experiments by modifying the clipping parameter and introducing a looser clipping setting. Specifically, when we set ε to a higher value (i.e. $\varepsilon = 0.5$), effectively loosening the clipping range, we observed notable effects on the performance of both MAPPO-ALR and MAPPOS. In these experiments, as ε increased, the performance of both algorithms deteriorates (Fig. 10, Tables 6, 7). However, it's worth noting that MAPPO-ALR exhibits the ability to maintain stability and efficiency even under looser clipping conditions compared to MAPPOS.

From these findings, we can conclude that while a looser clipping range can have a detrimental effect on the performance, MAPPO-ALR showcases resilience and efficiency, further emphasizing the importance of the activation likelihood ratio (ALR) enhancement in managing policy updates across different parameter settings.

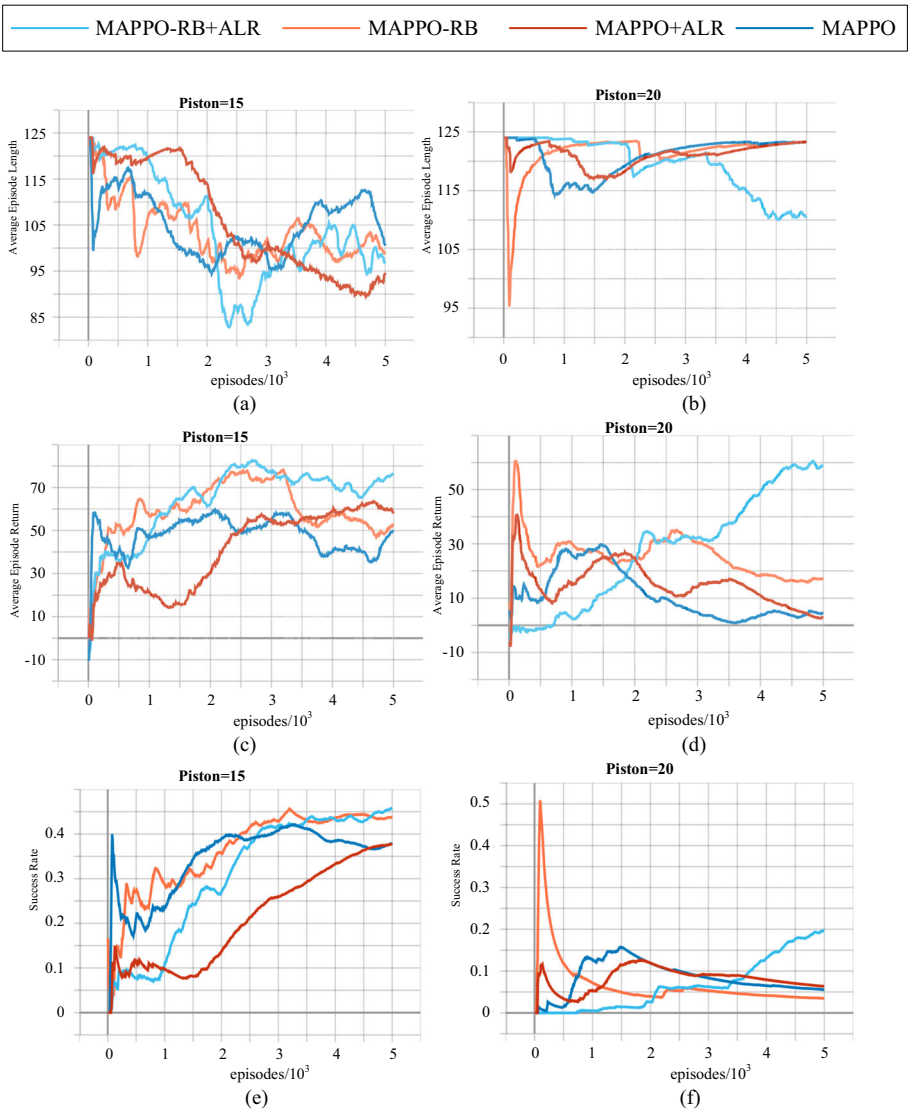


Fig. 8 Performance comparison of MAPPO, MAPPO-RB, MAPPO + ALR and MAPPO-RB + ALR algorithms. Average Episode Length: the curves of the consumption step size per episode at the cases of 15 agents (**a**) and 20 agents (**b**). Average Episode Return: the curves of the overall return value per episode at the cases of 15 agents (**c**) and 20 agents (**d**). Success Rate: the curves of the average probability the missions succeed at the cases of 15 agents (**e**) and 20 agents (**f**)

6 Conclusion

The inherent challenges of instability and inefficiency in policy optimization have long been associated with the PPO algorithm, especially in multi-agent environments. Previous literatures such as PPO-RB and PPOS, highlight the role of the clipping function in policy optimization and propose improved clipping mechanisms to restrict and benefit the policy

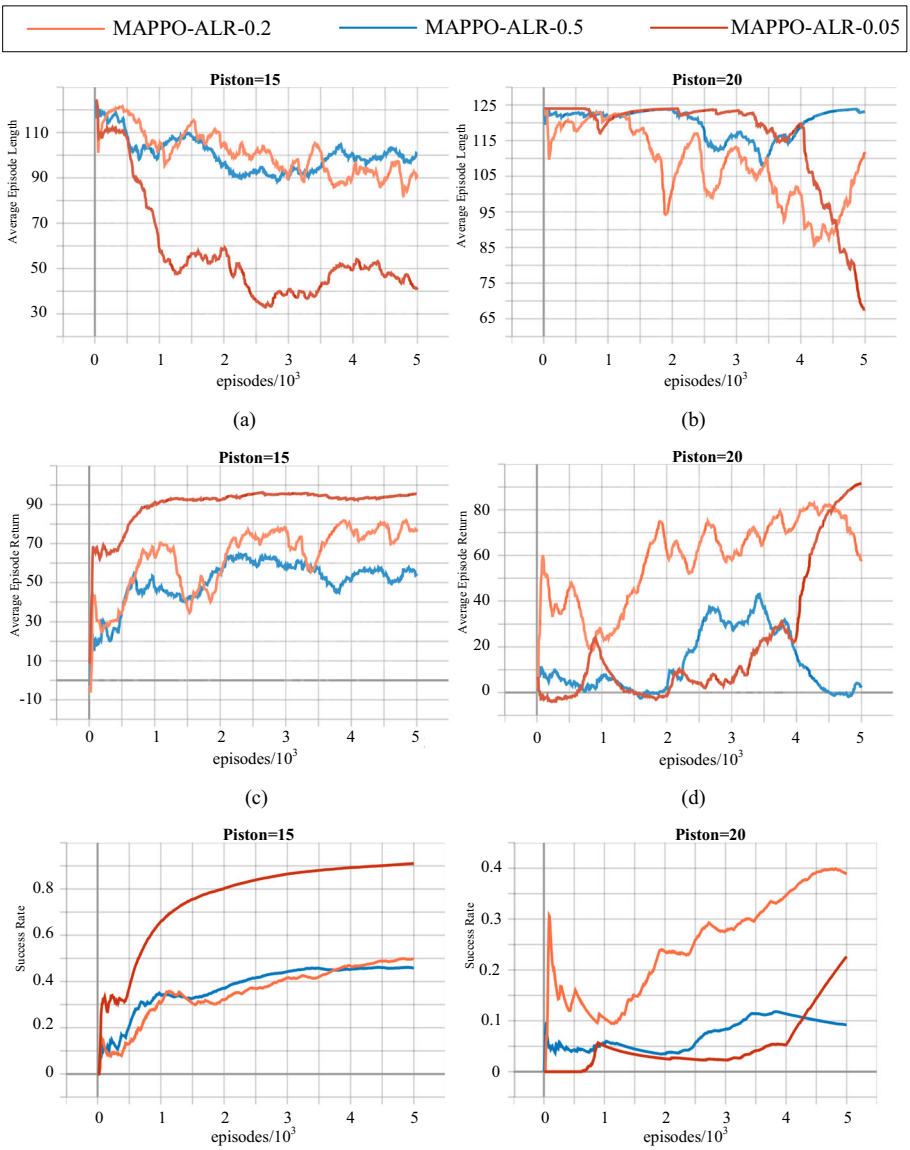


Fig. 9 Performance of MAPPO-ALR with $\varepsilon = 0.05$, $\varepsilon = 0.2$ and $\varepsilon = 0.5$. Average Episode Length: the curves of the consumption step size per episode at the cases of 15 agents (**a**) and 20 agents (**b**). Average Episode Return: the curves of the overall return value per episode at the cases of 15 agents (**c**) and 20 agents (**d**). Success Rate: the curves of the average probability the missions succeed at the cases of 15 agents (**e**) and 20 agents (**f**)

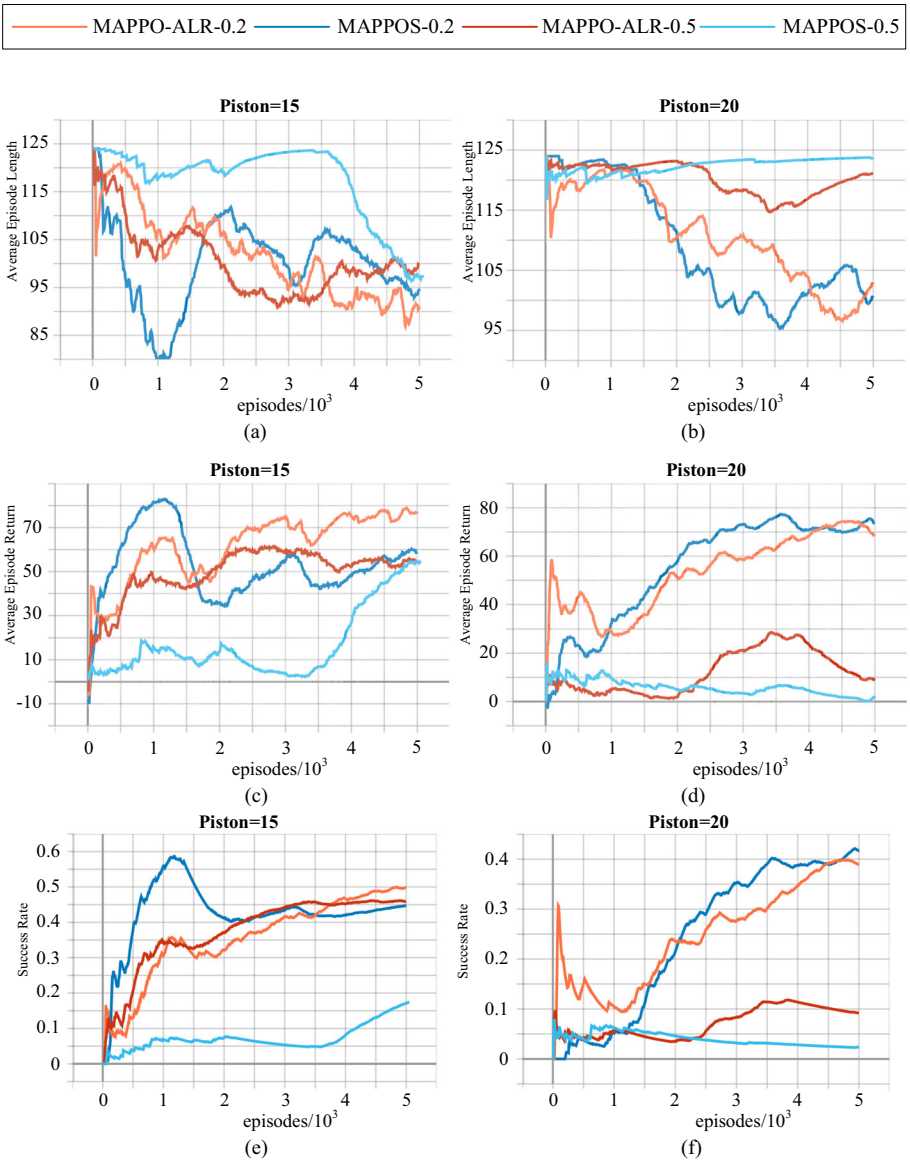


Fig. 10 Performance of MAPPOS and MAPPO-ALR with $\varepsilon = 0.2$ and $\varepsilon = 0.5$. Average Episode Length: the curves of the consumption step size per episode at the cases of 15 agents (**a**) and 20 agents (**b**). Average Episode Return: the curves of the overall return value per episode at the cases of 15 agents (**c**) and 20 agents (**d**). Success Rate: the curves of the average probability the missions succeed at the cases of 15 agents (**e**) and 20 agents (**f**)

Table 6 Numerical results of MAPPOS and MAPPO-ALR with different ε's at the case of 15 agents

	Average episode length	Average episode return	Success rate
MAPPO-ALR-0.05	59	95.47	0.9102
MAPPO-ALR-0.2	91.03	76.28	0.4978
MAPPOS-0.2	94.28	58.85	0.4468
MAPPO-ALR-0.5	99.69	54.48	0.4588
MAPPOS-0.5	97.21	54.32	0.1723

Table 7 Numerical results of MAPPOS and MAPPO-ALR with different ε's at the case of 20 agents

	Average episode length	Average episode return	Success rate
MAPPO-ALR-0.05	67.64	57.42	0.2264
MAPPO-ALR-0.2	103	68.29	0.3886
MAPPOS-0.2	100.7	73.74	0.4172
MAPPO-ALR-0.5	121.1	8.808	0.092
MAPPOS-0.5	123.5	2.223	0.0244

learning. These approaches primarily focus on defining downward-slope objectives to curb policy deviations when the likelihood ratio exceeds the clipping range. They neglect the optimization efficiency within the clipping range and fail to fully address the instability during policy optimization. In response to these challenges, we propose an activation likelihood ratio (ALR) function in the clipping mechanism to solve these problems. The ALR function dynamically adjusts the slope of the surrogate objective before the likelihood ratio extends beyond the clipping range. At the outset, the objective of PPO-ALR exhibits the steepest slope, leading to swift policy updates and high efficiency. As the ratio nears the clipping border, the slope of the PPO-ALR objective gradually diminishes, preventing the ratio from straying outside the clipping range and enhancing policy update stability. The theoretical properties and advantages of PPO-ALR are rigorously demonstrated, and experiments validate its superior performance in various settings. These experiments showcase shorter average episode lengths, increased average episode returns, and higher success rates for the proposed method. In addition, experiments in Sect. 5.4.3 highlight the versatility of ALR, demonstrating its compatibility with various clipping operations and its potential to improve the performance of the broader PPO series algorithms. The PPO-ALR method has demonstrated outstanding performance in the simulations presented in this paper. However, it is primarily proficient in cooperative multi-agent tasks. Its efficacy in other types of multi-agent problems should be validated. Furthermore, PPO-ALR method still exhibits high computational complexity, especially in large-scale environments or during prolonged training periods, necessitating substantial computational resources. In the future, we will consider more effective constraint functions to further improve the clipping strategies of PPO series algorithms. Additionally, we will try to explore their performance in more complex environments, such as the one having continuous action spaces and heterogeneous agents.

Acknowledgements We thank the editors and the reviewers for their help to this work. This work was supported by the National Natural Science Foundation of China under Grant 62101206 and the Provincial Postgraduate Innovation and Entrepreneurship Project under Grant 2022cxcysj020.

Author Contributions LJ and BS contributed mainly to the methodology design, experiment validation and original draft preparation. DX and YW helped on data collection and experimental analysis. JF commented on previous versions of the manuscript. JW provided important assistance in the experimental verification of the MP-MAPPO algorithm. All authors read and approved the final manuscript.

Declarations

Conflict of interest The authors declare no competing interests.

References

1. Vinyals O, Babuschkin I, Czarnecki WM (2019) Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature 575:350–354
2. Berner C, Brockman G, Chan B, Cheung V, Debiak P, Dennison C, Farhi D, Fischer Q, Hashme S, Hesse C, Jozefowicz R, Gray S, Olsson C, Pachocki J, Petrov M, de Oliveira Pinto HP, Raiman J, Salimans T, Schlatter J, Schneider J, Sidor S, Sutskever I, Tang J, Wolski F, Zhang S (2019) Dota 2 with large scale deep reinforcement learning. arXiv preprint arXiv:1912.06680
3. Ye D, Liu Z, Sun M, Shi B, Huang L (2020) Mastering complex control in MOBA games with deep reinforcement learning. In: Proceedings of the AAAI conference on artificial intelligence, vol 34(4), pp 6672–6679
4. Smit A, Engelbrecht HA, Brink W, Pretorius A (2023) Scaling multi-agent reinforcement learning to full 11 versus 11 simulated robotic football. Auton Agents Multi-Agent Syst 37(1):584
5. Wang H, Tang H, Hao J, Hao X, Fu Y, Ma Y (2020) Large scale deep reinforcement learning in war-games. In: 2020 IEEE international conference on bioinformatics and biomedicine (BIBM), Seoul, Korea (South), pp 1693–1699
6. Busoniu L, Babuska R, De Schutter B (2008) A comprehensive survey of multiagent reinforcement learning. IEEE Trans Syst Man Cybern C 38(2):156–172
7. Sunehag P, Lever G, Gruslys A, Czarnecki WM, Zambaldi VF, Jaderberg M, Lanctot M, Sonnerat N, Leibo JZ, Tuyls K, Graepel T (2018) Value-decomposition networks for cooperative multi-agent learning based on team reward. In: 17th international conference on autonomous agents and multiagent systems, vol 3, pp 2085–2087
8. Rashid T, Samvelyan M, de Witt CS, Farquhar G, Foerster JN, Whiteson S (2018) QMIX: monotonic value function factorization for deep multi-agent reinforcement learning. In: Proceedings of the 35th international conference on machine learning, vol 80, pp 4295–4304
9. Son K, Kim D, Kang WJ, Hostallero D, Yi Y (2019) QTRAN: learning to factorize with transformation for cooperative multi-agent reinforcement learning. In: International conference on machine learning, pp 5887–5896
10. Lowe R, Wu Y, Tamar A, Harb J, Abbeel P, Mordatch I (2017) Multiagent actor-critic for mixed cooperative-competitive environments. Adv Neural Inf Process Syst 30:6379–6390
11. Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D (2015) Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971

12. Foerster JN, Farquhar G, Afouras T, Nardelli N, Whiteson S (2018) Counterfactual multiagent policy gradients. In: Proceedings of the AAAI conference on artificial intelligence, vol 32(1)
13. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O (2017) Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347
14. Yu C, Velu A, Vinitsky E, Wang Y, Bayen AM, Wu Y (2022) The surprising effectiveness of MAPPO in cooperative, multi-agent games. arXiv preprint arXiv:2103.01955
15. Wang Y, He H, Tan X (2020) Truly proximal policy optimization. arXiv preprint arXiv:1903.07940
16. Zhu W, Rosendo A (2021) A functional clipping approach for policy optimization algorithms. IEEE Access 9:96056–96063
17. Papoudakis G, Christianos F, Schafer L, Albrecht SV (2021) Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. In: Thirty-fifth conference on neural information processing systems datasets and benchmarks track (round 1)
18. de Witt CS, Gupta T, Makoviichuk D, Makoviychuk V, Torr PHS, Sun M, Whiteson S (2020) Is independent learning all you need in the starcraft multi-agent challenge? arXiv preprint arXiv:2011.09533
19. Samvelyan M, Rashid T, de Witt CS, Farquhar G, Nardelli N, Rudner TGJ, Hung C-M, Torr PHS, Foerster JN, Whiteson S (2019) The starcraft multi-agent challenge. arXiv preprint arXiv:1902.04043
20. Terry JK, Grammel N, Hari A, Santos L, Black B (2020) Revisiting parameter sharing in multi-agent deep reinforcement learning. arXiv preprint arXiv:2005.13625
21. Mao H, Zhang Z, Xiao Z, Gong Z, Ni Y (2020) Learning multi-agent communication with double attentional deep reinforcement learning. Auton Agent Multi-Agent Syst 34:1–34
22. Gronauer S, Diepold K (2022) Multi-agent deep reinforcement learning: a survey. Artif Intell Rev 55:895–943
23. Schulman J, Levine S, Moritz P, Jordan MI, Abbeel P (2015) Trust region policy optimization. In: Proceedings of the 32nd international conference on machine learning, vol 37, pp 1889–1897
24. Peters J, Schaal S (2008) Reinforcement learning of motor skills with policy gradients. Neural Netw 21(4):682–697
25. Marcin A, Anton R, Piotr S, Manu O, Sertan G, Raphael M, Leonard H, Matthieu G, Olivier P, Marcin M, Sylvain G, Olivier B (2021) What matters for on-policy deep actor-critic methods? A large-scale study. In: International conference on learning representations 2021
26. Engstrom L, Ilyas A, Santurkar S, Tsipras D, Madry A (2020) Implementation matters in deep policy gradients: a case study on PPO and TRPO. arXiv preprint arXiv:2005.12729
27. Kaelbling LP, Littman ML, Cassandra AR (1998) Planning and acting in partially observable stochastic domains. Artif Intell 101(1–2):99–134
28. Cassandra AR (1998) Exact and approximate algorithms for partially observable Markov decision processes. Brown University
29. Wiering M, Otterlo MV (2012) Reinforcement learning: state of the art. Springer. https://doi.org/10.1007/978-3-642-27645-3
30. Oliehoek FA, Amato C (2016) A concise introduction to decentralized POMDPs. Springer. https://doi.org/10.1007/978-3-319-28929-8
31. Mnih V, Badia AP, Mirza M, Graves A, Lillicrap TP, Harley T, Silver D, Kavukcuoglu K (2016) Asynchronous methods for deep reinforcement learning. arXiv preprint arXiv:1602.01783
32. Terry JK, Black B, Hari A, Santos LS, Dieffendahl C, Williams NL, Lokesh Y, Horsch C, Ravi P (2021) Pettingzoo: gym for multi-agent reinforcement learning. Adv Neural Inf Process Syst 34:15032–15043
33. Terry JK, Black B, Hari A (2020) Supersuit: simple microwrappers for reinforcement learning environments. arXiv preprint arXiv:2008.08932
34. Pan L, Cai Q, Huang L (2021) Exploration in policy optimization through multiple paths. Auton Agent Multi-Agent Syst 35:33
35. Ilyas A, Engstrom L, Santurkar, S, Tsipras D, Janoos F, Rudolph L, Madry A (2018) Are deep policy gradient algorithms truly policy gradient algorithms? arXiv preprint arXiv:1811.02553