HOLDING MONOTONIC IMPROVEMENT AND GENERALITY FOR MULTI-AGENT PROXIMAL POLICY OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Proximal Policy Optimization (PPO) has achieved empirical successes in the field of single-agent reinforcement learning thanks to guaranteed monotonic improvement. The theoretical support makes its extension in multi-agent systems very attractive. However, existing PPO-based algorithms in cooperative multi-agent reinforcement learning (MARL) either lack the theoretical monotonic improvement guarantee or have inevitably restrictive settings, which greatly limit their applicable scenarios. In this paper, we propose a theoretically-justified and general-purpose multi-agent PPO algorithm for cooperative MARL called Full-Pipeline PPO (FP3O). The core idea of FP3O is to dynamically allocate agents to different optimization pipelines and perform the proposed one-separation trust region optimization for each pipeline. We prove in theory the monotonicity of joint policy improvement when executing the policy iteration procedure of FP3O. In addition, FP3O enjoys high generality since it avoids the restrictive factors that could arise in other existing PPO-based algorithms. In our experiments, FP3O outperforms other strong baselines on Multi-Agent MuJoCo and StarCraftII Multi-Agent Challenge benchmarks and also demonstrates its generality to the common network types (i.e., full parameter sharing, partial parameter sharing, and non-parameter sharing) and various multi-agent tasks.

1 INTRODUCTION

Proximal Policy Optimization (PPO) (Schulman et al., 2017) has been deemed as one of the most effective algorithms in single-agent reinforcement learning (RL) with empirical successes in a wide range of challenging domains, such as game playing (Bellemare et al., 2013), robotic control (Duan et al., 2016) and human-level task (Berner et al., 2019). It was originally developed from *Trust Region Policy Optimization (TRPO)* (Schulman et al., 2015), which enjoys the theoretical monotonic improvement guarantee during the training process by restricting the Kullback-Leibler (KL) divergence between the updated policy and the old one to a trust region. PPO significantly reduces computational complexity by replacing the KL divergence constraint in TRPO with the clipping mechanism. This improvement enables PPO to adopt the first-order optimizer to optimize the objective which greatly simplifies its implementation on deep neural networks.

Unfortunately, in multi-agent reinforcement learning (MARL), independently applying single-agent RL algorithms to each individual agent is poorly suitable. This is because, for each agent, the state transition and reward function will become non-stationary with the changing of other agents' policies. The non-stationary environment makes each agent suffer from the problem of learning stability (Hernandez-Leal et al., 2017; Papoudakis et al., 2019). Recently, the paradigm of *centralized training with decentralized execution (CTDE)* has been widely adopted to combat this challenge. Each agent in CTDE has access to extra information (e.g., global information and policies of other agents) to alleviate the non-stationary issue during the training phase, and only uses the local information during the execution phase. Based on the CTDE framework, varieties of multi-agent algorithms have sprung up (Lowe et al., 2017; Rashid et al., 2018; Mahajan et al., 2019; Wang et al., 2021a;c).

However, existing PPO-based algorithms in cooperative MARL either lack theoretical support or have inevitably restrictive settings, which greatly limit their applicable scenarios. Both Independent

PPO (IPPO) (de Witt et al., 2020a) and Multi-Agent PPO (MAPPO) (Yu et al., 2021) simply applied PPO to the multi-agent domains by optimizing every agent with trust region learning under the full decentralization framework or the CTDE framework. Despite their empirical successes, the direct extension could make the monotonic improvement guarantee no longer hold since an agent's payoff is also affected by other agents' actions. This fact disturbs their performance even in some simple cooperative multi-agent tasks (Kuba et al., 2022). For this, Kuba et al. (2022) proposed the theoretically-justified Heterogeneous-Agent PPO (HAPPO) algorithm with guaranteed monotonic improvement. HAPPO adopts a sequential update scheme: all agents are updated one by one, so that the expected joint advantage can be improved sequentially. Unfortunately, HAPPO is limited to a specific setting (i.e., non-parameter sharing). In other popular types of multi-agent tasks such as training homogeneous agents with full parameter sharing (Yu et al., 2021) or training heterogeneous agents with partial parameter sharing (Chenghao et al., 2021; Christianos et al., 2021), HAPPO will suffer from the problems of uncontrolled KL divergence and poor data efficiency, which greatly limit its generality. Specifically, if the policy networks of agents in HAPPO are associated with each other, the former agents' update will cause the latter agents' policy networks to be changed. These changes can be accumulated as the sequential update scheme progresses. As a result, even though the latter agents have not initiated their update rounds, their policies are very likely to have reached or even exceeded the boundary of the KL divergence trust region. This causes the sampling data of the latter agents to have little positive impact on network optimization (see Appendix A for details).

In this paper, we propose a theoretically-justified multi-agent PPO algorithm called *Full-Pipeline PPO (FP3O)* for cooperative MARL, which holds both theoretical monotonic improvement guarantee and high generality. FP3O describes a joint policy iteration procedure where agents are dynamically allocated to different optimization pipelines to be updated. Each basic optimization pipeline follows the proposed *one-separation trust region optimization*, the key of which is separating an arbitrary agent's contribution from the joint advantage function. We prove in theory that this update scheme guarantees the monotonic joint policy improvement at every iteration. Besides, it enables FP3O to make full use of all sampling data to execute a trust region optimization step without the restrictive factors that could arise in other existing PPO-based algorithms (e.g., the specific network setting in HAPPO). In our experiments, Multi-Agent MuJoCo (MAMuJoCo) (de Witt et al., 2020b) and StarCraftII Multi-Agent Challenge (SMAC) (Samvelyan et al., 2019) benchmarks are used to evaluate our algorithm against several strong baselines. The results demonstrate the superior performances of FP3O and its generality to the common network types (i.e., full parameter sharing, partial parameter sharing and non-parameter sharing) and various multi-agent tasks.

2 Related Work

In this section, we will systematically introduce our related works: trust region learning in MARL. IPPO (de Witt et al., 2020a) directly applied PPO algorithm to each agent in the multi-agent systems and achieved high performances on some SMAC tasks. However, the property of full decentralization makes IPPO suffer from the non-stationary issue. By extending IPPO to the CTDE framework and designing the global agent-specific features for the centralized critic function, Yu et al. (2021) proposed MAPPO algorithm with more stable and stronger performances in SMAC. However, both IPPO and MAPPO ignore the interaction between agents and lack the theoretical monotonic improvement guarantee. Li & He (2020) theoretically extended TRPO to the multi-agent domain and proposed MATRPO, but the KL divergence constraint is associated with the number of agents, which greatly limits its application in scenarios with a large number of agents. This limitation also occurs in the game-theoretical MATRL (Wen et al., 2021), which computes independent trust region learning for each agent and tries to adopt meta-game Nash equilibrium to find the restricted optimization step but can only deal with two-agent cases. Wu et al. (2021) proposed theoretically-justified CoPPO, which directly updates the joint policy during optimization. However, the optimization objective of each agent will constantly change after every mini-batch update leading to the non-stationary problem. Recently, Kuba et al. (2022) proposed theoretically-justified HAPPO/HATRPO algorithm with the sequential update scheme and achieved state-of-the-art performances with non-parameter sharing on heterogeneous-agent tasks. However, the sequential update scheme suffers from the problems of uncontrolled KL divergence and poor data efficiency in some common cases such as training homogeneous agents with full parameter sharing or training heterogeneous agents with partial parameter sharing. In summary, the aforementioned existing trust region learning methods in MARL either lack the theoretical monotonic improvement guarantee or have inevitable restrictions.

3 PRELIMINARIES

Problem formulation. In this paper, we use a multi-agent version of discounted Markov decision process (MDP) (Littman, 1994) to model a fully cooperative multi-agent task, which can be described as a tuple $\mathfrak{M} =< \mathcal{S}, \mathcal{N}, \boldsymbol{\mathcal{A}}, P, r, \gamma, \rho_0 >$. $s \in \mathcal{S}$ is the state of the environment. $i \in \mathcal{N} \equiv \{1, ..., n\}$ denotes the index of agent. $\boldsymbol{\mathcal{A}}$ represents the joint action space, which is defined by $\prod_{i=1}^{n} \mathcal{A}^i$, the product of all agents' action spaces. $\gamma \in [0, 1)$ is the discount factor. $\rho_0 : \mathcal{S} \to \mathbb{R}$ is the distribution of the initial state s_0 of environment. At each timestep $t \in \mathbb{N}$, agent i will choose an action $a_t^i \in \mathcal{A}^i$ according to its policy $\pi^i(\cdot|s_t)$. The joint action $\boldsymbol{a}_t = \left\{a_t^1, ..., a_t^n\right\}$ and the joint policy $\boldsymbol{\pi}(\boldsymbol{a}_t|s_t) = \prod_{i=1}^{n} \pi^i(a_t^i|s_t)$ can be formed by combining all agents' actions and policies. Then, the state of the environment is transferred to a new state according to the transition probability function $P(s_{t+1}|s_t, \boldsymbol{a}_t) : S \times \boldsymbol{\mathcal{A}} \times S \to [0, 1]$, and all agents will receive a shared reward $r(s_t, \boldsymbol{a}_t) : S \times \boldsymbol{\mathcal{A}} \to \mathbb{R}$. We let $\rho_\pi(s) = \sum_{t=0}^{\infty} \gamma^t P(s_t = s)$ denote the (unnormalized) state visitation frequencies. The optimization objective for all agents in the fully cooperative multi-agent task is maximizing the discounted cumulative reward: $\mathcal{J}(\boldsymbol{\pi}) \triangleq \mathbb{E}_{s_{0:\infty} \sim \rho_\pi, \boldsymbol{a}_{0:\infty} \sim \boldsymbol{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, \boldsymbol{a}_t)\right]$. Then, we define the joint state value function as $V_{\boldsymbol{\pi}}(s) \triangleq \mathbb{E}_{s_{1:\infty} \sim \rho_\pi, \boldsymbol{a}_{0:\infty} \sim \boldsymbol{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, \boldsymbol{a}_t) \middle| s_0 = s\right]$, the joint state-action function as $Q_{\boldsymbol{\pi}}(s, \boldsymbol{a}) \triangleq \mathbb{E}_{s_{1:\infty} \sim \rho_\pi, \boldsymbol{a}_{1:\infty} \sim \boldsymbol{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, \boldsymbol{a}_t) \middle| s_0 = s, \boldsymbol{a}_0 = \boldsymbol{a}\right]$ and the joint advantage function as $A_{\boldsymbol{\pi}}(s, \boldsymbol{a}) \triangleq Q_{\boldsymbol{\pi}}(s, \boldsymbol{a}) - V_{\boldsymbol{\pi}}(s)$.

Credit assignment. Quantifying agents' contribution from the whole team's joint reward is a crucial challenge known as the *credit assignment* problem (Chang et al., 2003). Currently, there are many efforts to address this issue from both value-based (Sunehag et al., 2018; Son et al., 2019) and policy-based perspectives (Foerster et al., 2018). In this paper, we follow Kuba et al. (2022) to deduce the contribution of different subsets of agents. We let $i_{1:m}$ be a subset $\{i_1, ..., i_m\}$ of $\mathcal{N}$, and let $-i_{1:m}$ be the complement set of $i_{1:m}$. In particular, $i_{1:n}$ represents the complete set $\mathcal{N}$ of all agents. Then, the multi-agent state-action value function can be defined as

$$Q_{\boldsymbol{\pi}}^{i_{1:m}}(s, \boldsymbol{a}^{i_{1:m}}) \triangleq \mathbb{E}_{\boldsymbol{a}^{-i_{1:m}} \sim \boldsymbol{\pi}^{-i_{1:m}}} \left[Q_{\boldsymbol{\pi}}(s, \boldsymbol{a})\right], \tag{1}$$

which quantifies the contribution of agents $i_{1:m}$ to the joint reward by calculating the average return if agents $i_{1:m}$ execute joint action $\boldsymbol{a}^{i_{1:m}}$. It will degenerate into the joint state value function $V_{\boldsymbol{\pi}}(s)$ in case of $m = 0$, and the joint state-action function $Q_{\boldsymbol{\pi}}(s, \boldsymbol{a})$ in case of $m = n$. Then, for arbitrary disjoint sets $j_{1:k}$ and $i_{1:m}$, the multi-agent advantage function can be defined as

$$A_{\boldsymbol{\pi}}^{i_{1:m}}(s, \boldsymbol{a}^{j_{1:k}}, \boldsymbol{a}^{i_{1:m}}) \triangleq Q_{\boldsymbol{\pi}}^{j_{1:k}, i_{1:m}}(s, \boldsymbol{a}^{j_{1:k}, i_{1:m}}) - Q_{\boldsymbol{\pi}}^{j_{1:k}}(s, \boldsymbol{a}^{j_{1:k}}), \tag{2}$$

which quantifies the additional contribution of agents $i_{1:m}$ to execute joint action $\boldsymbol{a}^{i_{1:m}}$ if agents $j_{1:k}$ have executed joint action $\boldsymbol{a}^{j_{1:k}}$. It is a general setup without extra assumptions for the credit assignment problem in cooperative MARL and helps to establish an interaction between different agents by describing how agents $j_{1:k}$ affect the payoff of agents $i_{1:m}$.

Single-agent trust region learning. Single-agent trust region learning, such as TRPO and PPO, can guarantee the discounted cumulative reward $\mathcal{J}(\pi)$ is non-decreasing by improving its lower bound at each policy iteration process. The lower bound is given by the following theorem.

Theorem 1. *(Schulman et al., 2015) Let π denote the old policy, $\tilde{\pi}$ denote the new policy that is obtained by optimizing π, $L_\pi(\tilde{\pi}) = \mathcal{J}(\pi) + \mathbb{E}_{s \sim \rho_\pi, a \sim \tilde{\pi}}[A_\pi(s, a)]$, $D_{\mathrm{KL}}^{\max}(\pi, \tilde{\pi}) = \max_s D_{\mathrm{KL}}(\pi(\cdot|s), \tilde{\pi}(\cdot|s))$, $C = \frac{4\gamma \max_{s,a} |A_\pi(s,a)|}{(1-\gamma)^2}$. The following policy improvement bound holds:*

$$\mathcal{J}(\tilde{\pi}) \geq L_\pi(\tilde{\pi}) - C D_{\mathrm{KL}}^{\max}(\pi, \tilde{\pi}).$$

Theorem 1 describes an approximate optimization scheme: we can just improve the right-hand lower bound by performing $\mathrm{maximize}_{\tilde{\pi}} \left[L_\pi(\tilde{\pi}) - C D_{\mathrm{KL}}^{\max}(\pi, \tilde{\pi})\right]$ at each policy iteration process, so that the true objective $\mathcal{J}$ can also be improved (see Equation (10) in Schulman et al. (2015)). Thus, the lower bound is also regarded as the surrogate objective for the true objective $\mathcal{J}$.

To derive a practical algorithm, TRPO was proposed with parameterized policies π_θ and the average KL divergence constraint. However, it has expensive computing overhead and cannot adopt first-order optimization for policy networks. To solve this problem, Schulman et al. (2017) replaced the

average KL divergence constraint with the clipping mechanism and proposed PPO algorithm, which aims to maximize the PPO-clip objective as

$$\underset{\theta}{\text{maximize}}\, \mathbb{E}_{s\sim\rho_{\pi_{\theta_{\text{old}}}},a\sim\pi_{\theta_{\text{old}}}}\left[\min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)}A_{\pi_{\theta_{\text{old}}}}(s,a), \text{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)}, 1\pm\epsilon\right)A_{\pi_{\theta_{\text{old}}}}(s,a)\right)\right], \tag{3}$$

where $\pi_{\theta_{\text{old}}}$ is the old policy that we want to improve, $A_{\pi_{\theta_{\text{old}}}}$ is the estimator of the advantage function, and ϵ is a hyperparameter used to restrict the optimization step size to a trust region by clipping the ratio $\pi_\theta(a|s)/\pi_{\theta_{\text{old}}}(a|s)$. This clipping mechanism, together with the solid theoretical support of Theorem 1, enables PPO to achieve stable performance faster than TRPO.

4 THE FP3O ALGORITHM

In this section, we will introduce our theoretically-justified Full-Pipeline PPO (FP3O) algorithm. Subsection 4.1 introduces the basic optimization pipeline of FP3O: one-separation trust region optimization. The key is to separate an arbitrary agent's contribution and then perform the independence-dependence optimization. In Subsection 4.2, we introduce the full-pipeline optimization and then develop it into a practical form. In Subsection 4.3, we adopt the clipping mechanism and parameterized policies, which together yield the F3PO algorithm. In Subsection 4.4, we further analyze the monotonic joint policy improvement of FP3O.

4.1 ONE-SEPARATION TRUST REGION OPTIMIZATION

At the beginning of each policy iteration process, each agent has no chance to know the actions and the optimization directions of other agents. If the optimization objective of one agent is bound by other agents at the beginning of the policy iteration, it will be very thorny. To address this issue, we give the following lemma based on Equations (1) and (2) to separate one expected agent's contribution independent of other agents from the joint advantage function.

Lemma 1. *(One-Separation Advantage Decomposition) Let $\varnothing$ denote the empty set. In cooperative multi-agent tasks, for $\forall i_p \in i_{1:n} = \mathcal{N}$, the joint advantage function $A_{\boldsymbol{\pi}}(s,\boldsymbol{a})$ can be decoupled as*

$$A_{\boldsymbol{\pi}}(s,\boldsymbol{a}) = A^{i_p}_{\boldsymbol{\pi}}(s,a^{\varnothing},a^{i_p}) + A^{-i_p}_{\boldsymbol{\pi}}(s,a^{i_p},\boldsymbol{a}^{-i_p}).$$

The proof is in Appendix B.1. In Lemma 1, the joint advantage function is decoupled into two parts: the contribution $A^{i_p}_{\boldsymbol{\pi}}(s,a^{\varnothing},a^{i_p})$ of agent i_p which is independent of other agents $-i_p$, and the contribution $A^{-i_p}_{\boldsymbol{\pi}}(s,a^{i_p},\boldsymbol{a}^{-i_p})$ of agents $-i_p$ which depends on the action taken by agent i_p. It allows us to separate an arbitrary agent i_p to start a basic policy iteration without considering the influence of other agents, which is critical for full-pipeline optimization in Subsection 4.2.

Theorem 1 in single-agent trust region learning achieves the monotonic improvement of the true objective by optimizing its surrogate objective. Similarly, before extending Theorem 1 to the multi-agent domains, we give the following definition as our surrogate objective.

Definition 1. *(Surrogate Objective) Let $\boldsymbol{\pi}$ denote the old joint policy, $\tilde{\boldsymbol{\pi}}$ denote the new joint policy, $\mathcal{C} = \frac{4\gamma\max_{s,\boldsymbol{a}}|A_{\boldsymbol{\pi}}(s,\boldsymbol{a})|}{(1-\gamma)^2}$. For any subset $i_{1:m}$ of $i_{1:n}$, we take $\forall i_p \in i_{1:m}$ or $i_p = \varnothing$, and then define*

$$\begin{aligned} &M_{\boldsymbol{\pi}}{}^{i_{1:m}-i_p}(\tilde{\pi}^{i_p},\tilde{\boldsymbol{\pi}}^{i_{1:m}-i_p}) \\ &\quad= \mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}},\boldsymbol{a}^{i_{1:m}}\sim\tilde{\boldsymbol{\pi}}^{i_{1:m}}}\left[A^{i_{1:m}-i_p}_{\boldsymbol{\pi}}(s,a^{i_p},\boldsymbol{a}^{i_{1:m}-i_p})\right] - \sum\nolimits_{i_k\in i_{1:m}-i_p}\mathcal{C}D^{\max}_{\text{KL}}(\pi^{i_k},\tilde{\pi}^{i_k}). \end{aligned}$$

Then, we can derive the following theorem in MARL with Definition 1, Lemma 1 and Theorem 1.

Theorem 2. *Let $\boldsymbol{\pi}$ denote the old joint policy, $\tilde{\boldsymbol{\pi}}$ denote the new joint policy that is obtained by optimizing $\boldsymbol{\pi}$. For $\forall i_p \in i_{1:n}$, the following lower bound holds.*

$$\mathcal{J}(\tilde{\boldsymbol{\pi}}) \geq \mathcal{J}(\boldsymbol{\pi}) + M^{i_p}_{\boldsymbol{\pi}}(\tilde{\pi}^{\varnothing},\tilde{\pi}^{i_p}) + M^{-i_p}_{\boldsymbol{\pi}}(\tilde{\pi}^{i_p},\tilde{\boldsymbol{\pi}}^{-i_p}).$$

The proof is in Appendix B.2. This theorem describes the joint policy improvement bound of true objective $\mathcal{J}$ in multi-agent domains based on one-separation advantage decomposition. We can improve the lower bound by maximizing the surrogate objectives $M^{i_p}_{\boldsymbol{\pi}}(\tilde{\pi}^{\varnothing},\tilde{\pi}^{i_p})$ and $M^{-i_p}_{\boldsymbol{\pi}}(\tilde{\pi}^{i_p},\tilde{\boldsymbol{\pi}}^{-i_p})$

at each policy iteration to guarantee the monotonically improving true objective $\mathcal{J}$. To this end, we propose *one-separation trust region optimization* and describe the policy iteration process in Algorithm 1. At each policy iteration k, we perform *independence-dependence optimization* to make the lower bound raised and get the updated joint policy $\boldsymbol{\pi}_{k+1}$. Specifically, we do $\text{maximize}_{\pi^{i_p}}[M^{i_p}_{\boldsymbol{\pi}_k}(\pi^{\varnothing}_{k+1}, \pi^{i_p})]$ for agent i_p independently of agents $-i_p$ at independence step, and do $\text{maximize}_{\boldsymbol{\pi}^{-i_p}}[M^{-i_p}_{\boldsymbol{\pi}_k}(\pi^{i_p}_{k+1}, \boldsymbol{\pi}^{-i_p})]$ for agents $-i_p$ dependently of agent i_p at dependence step. Then we can get non-decreasing true objective $\mathcal{J}(\boldsymbol{\pi}_{k+1}) \geq \mathcal{J}(\boldsymbol{\pi}_k)$. See Appendix B.3 for the proof of independence-dependence optimization enabling improved lower bound and non-decreasing $\mathcal{J}$.

Algorithm 1 Policy iteration of one-separation trust region optimization

1: Initialize the joint policy $\boldsymbol{\pi}_0$.
2: **for** $k = 1, 2, ...$ until convergence **do**
3: Separate one arbitrary agent i_p from $i_{1:n}$.
4: Compute advantage values $A^{i_p}_{\pi_k}(s, a^{\varnothing}, a^{i_p})$ and $A^{-i_p}_{\boldsymbol{\pi}_k}(s, a^{i_p}, \boldsymbol{a}^{-i_p})$.
5: Solve the independence-dependence optimization:
6: Independence step: $\pi^{i_p}_{k+1} = \arg\max_{\pi^{i_p}}[M^{i_p}_{\boldsymbol{\pi}_k}(\pi^{\varnothing}_{k+1}, \pi^{i_p})]$
7: Dependence step: $\boldsymbol{\pi}^{-i_p}_{k+1} = \arg\max_{\boldsymbol{\pi}^{-i_p}}[M^{-i_p}_{\boldsymbol{\pi}_k}(\pi^{i_p}_{k+1}, \boldsymbol{\pi}^{-i_p})]$
8: **end for**

4.2 Practical Full-Pipeline Optimization

In the previous subsection, we have discussed that one-separation advantage decomposition provides a critical insight that an *arbitrary* agent i_p can be separated to start a policy iteration described by Algorithm 1 without considering the influence of other agents. With this property, we can take one-separation trust region optimization as the basic optimization pipeline, and then let $i_p = i_1, ..., i_n$ sequentially so that every agent can be the separated agent to start a one-separation trust region optimization. As a result, we get n equivalent policy improvement lower bounds on n parallel pipelines with p denoting the index of the pipeline:

$$\left\{\mathcal{J}(\boldsymbol{\pi}) + M^{i_p}_{\boldsymbol{\pi}}(\tilde{\pi}^{\varnothing}, \tilde{\pi}^{i_p}) + M^{-i_p}_{\boldsymbol{\pi}}(\tilde{\pi}^{i_p}, \tilde{\boldsymbol{\pi}}^{-i_p})\right\}^{n}_{p=1}. \tag{4}$$

We notice that agents $i_{1:n}$ are assigned to n different optimization pipelines, which means that these pipelines can exactly cover all agents' updates when they execute the basic one-separation trust region optimization in parallel, so we call this process *full-pipeline optimization*. Different from the sequential update scheme in HAPPO, full-pipeline optimization enables all agents to be updated simultaneously with all collected data in one episode. However, the implementation of full-pipeline optimization in Equation (4) is not practical. We next develop a practical full-pipeline optimization through non-overlapping selection, importance sampling, and pipeline interaction.

Non-overlapping selection. For the one-separation trust region optimization of pipeline p, we aim to optimize the joint policy $\tilde{\boldsymbol{\pi}}^{-i_p}$ at the dependence step, which involves the policy update of agents $-i_p$. Unfortunately, for each agent in $-i_p$, the change of other agents' policies will cause its optimization objective $M^{-i_p}_{\boldsymbol{\pi}}(\tilde{\pi}^{i_p}, \tilde{\boldsymbol{\pi}}^{-i_p})$ to be unstable. In our solution, we select one specific agent's policy from $\tilde{\boldsymbol{\pi}}^{-i_p}$ to optimize and take the remaining part as a constant instead of directly optimizing the joint policy. The selection rule satisfies the below non-overlapping selection:

$$f : i_{1:n} \to j_{1:n}, \quad \text{where } j_p \in -i_p,\ j_{1:n} = \mathcal{N}. \tag{5}$$

Specifically, it describes an optimization rule for one-separation trust region optimization of pipeline p: policy $\tilde{\pi}^{i_p}$ is optimized at independence step and only $\tilde{\pi}^{j_p}$ is selected from $\tilde{\boldsymbol{\pi}}^{-i_p}$ to be optimized at dependence step. Although the remaining agents' policies $\tilde{\boldsymbol{\pi}}^{-i_p-j_p}$ are fixed to enhance the optimization stability of pipeline p, it does not mean that they will not be updated. Thanks to full-pipeline optimization and non-overlapping selection, they can be optimized by other pipelines. This is because $j_{1:n} = \mathcal{N}$ ensures that n agents can be exactly selected by n different pipelines to be optimized without overlapping at dependence step, which also retains the property of simultaneous update of all agents in full-pipeline optimization.

Importance sampling. For the one-separation trust region optimization of pipeline p in Algorithm 1, each agent has to maintain advantage estimators for $A_{\pi}^{i_p}(s, a^{\varnothing}, a^{i_p})$ and $A_{\boldsymbol{\pi}}^{-i_p}(s, a^{i_p}, \boldsymbol{a}^{-i_p})$ with the dynamic action inputs due to $\mathbb{E}_{a^{i_p}\sim\tilde{\pi}^{i_p}}[\cdot]$ and $\mathbb{E}_{\boldsymbol{a}\sim\tilde{\boldsymbol{\pi}}}[\cdot]$ in surrogate objectives $M_{\boldsymbol{\pi}}^{i_p}(\tilde{\pi}^{\varnothing}, \tilde{\pi}^{i_p})$ and $M_{\boldsymbol{\pi}}^{-i_p}(\tilde{\pi}^{i_p}, \tilde{\boldsymbol{\pi}}^{-i_p})$. After considering all pipelines, it will be more complicated and computationally expensive. To combat this challenge, we apply the importance sampling method for the advantage functions in $M_{\boldsymbol{\pi}}^{i_p}(\tilde{\pi}^{\varnothing}, \tilde{\pi}^{i_p})$ and $M_{\boldsymbol{\pi}}^{-i_p}(\tilde{\pi}^{i_p}, \tilde{\boldsymbol{\pi}}^{-i_p})$ with $\tilde{\pi}^*$ denoting the candidate update policy and get:

$$\mathbb{E}_{a^{i_p}\sim\tilde{\pi}^{i_p}}\left[A_{\boldsymbol{\pi}}^{i_p}(s, a^{\varnothing}, a^{i_p})\right] = \mathbb{E}_{\boldsymbol{a}\sim\boldsymbol{\pi}}\left[\frac{\tilde{\pi}^{*i_p}(a^{i_p}|s)}{\pi^{i_p}(a^{i_p}|s)} A_{\boldsymbol{\pi}}(s, \boldsymbol{a})\right], \tag{6}$$

$$\mathbb{E}_{\boldsymbol{a}\sim\tilde{\boldsymbol{\pi}}}\left[A_{\boldsymbol{\pi}}^{-i_p}(s, a^{i_p}, \boldsymbol{a}^{-i_p})\right] = \mathbb{E}_{\boldsymbol{a}\sim\boldsymbol{\pi}}\left[\left(\frac{\tilde{\pi}^{*j_p}(a^{j_p}|s)}{\pi^{j_p}(a^{j_p}|s)} \frac{\tilde{\boldsymbol{\pi}}^{-j_p}(\boldsymbol{a}^{-j_p}|s)}{\boldsymbol{\pi}^{-i_p}(\boldsymbol{a}^{-j_p}|s)} - \frac{\tilde{\pi}^{i_p}(a^{i_p}|s)}{\pi^{i_p}(a^{i_p}|s)}\right) A_{\boldsymbol{\pi}}(s, \boldsymbol{a})\right], \tag{7}$$

where the sampling of actions $\{a^1, ..., a^n\}$ is changed from $\{\tilde{\pi}^1, ..., \tilde{\pi}^n\}$ to $\{\pi^1, ..., \pi^n\}$. The proof is in Appendix B.4. The above two equations enable all agents at all pipelines to maintain a joint advantage estimator for $A_{\boldsymbol{\pi}}(s, \boldsymbol{a})$ like GAE (Schulman et al., 2016), which can be easily implemented under the CTDE framework with a shared centralized critic network and non-dynamic action input.

Pipeline interaction. For pipeline p, we cannot obtain $\tilde{\boldsymbol{\pi}}^{-j_p}(\boldsymbol{a}^{-j_p}|s)$ and $\tilde{\pi}^{i_p}(a^{i_p}|s)$ in Equations (6), (7) since they will be updated by other pipelines at dependence step and can only be determined after all pipelines finish the independence-dependence optimization. Fortunately, the *middle* update policies $\{\hat{\pi}^1, ..., \hat{\pi}^n\}$ can be generated from different pipelines at the end of independence step. Thus, by replacing $\tilde{\boldsymbol{\pi}}^{-j_p}(\boldsymbol{a}^{-j_p}|s)$ and $\tilde{\pi}^{i_p}(a^{i_p}|s)$ with $\hat{\boldsymbol{\pi}}^{-j_p}(\boldsymbol{a}^{-j_p}|s)$ and $\hat{\pi}^{i_p}(a^{i_p}|s)$, we can get the approximations (i.e., $\mathbb{E}_{a^{i_p}\sim\hat{\pi}^{i_p}}[A_{\boldsymbol{\pi}}^{i_p}(s, a^{\varnothing}, a^{i_p})]$ and $\mathbb{E}_{a^{j_p}\sim\tilde{\pi}^{j_p}, \boldsymbol{a}^{-j_p}\sim\hat{\boldsymbol{\pi}}^{-j_p}}[A_{\boldsymbol{\pi}}^{-i_p}(s, a^{i_p}, \boldsymbol{a}^{-i_p})]$) of Equations (6), (7). This approximation establishes the interaction between different pipelines through mutual utilization of the middle policies, which is an effective solution to the above problem. More importantly, it does not break the monotonic improvement, which will be discussed in Subsection 4.4. Algorithm 2 describes the policy iteration of practical full-pipeline optimization.

Algorithm 2 Policy iteration of practical full-pipeline optimization

1: Initialize the joint policy $\boldsymbol{\pi}_0$.
2: **for** $k = 1, 2, ...$ until convergence **do**
3: Compute the joint advantage estimator $A_{\boldsymbol{\pi}}(s, \boldsymbol{a})$.
4: Solve the independence-dependence optimization for all pipelines in parallel:
5: Independence step: $\left\{\hat{\pi}_{k+1}^{i_p} = \arg\max_{\pi^{i_p}}[M_{\boldsymbol{\pi}_k}^{i_p}(\pi^{\varnothing}, \pi^{i_p})]\right\}_{p=1}^{n}$
6: Select the policies to be optimized according to non-overlapping selection.
7: Replace the remaining policies with the middle policies $\hat{\pi}_{k+1}^{i_1}, ..., \hat{\pi}_{k+1}^{i_n}$.
8: Dependence step: $\left\{\pi_{k+1}^{j_p} = \arg\max_{\pi^{j_p}}[M_{\boldsymbol{\pi}_k}^{-i_p}(\hat{\pi}_{k+1}^{i_p}, \boldsymbol{\pi}^{-i_p})]\right\}_{p=1}^{n}$
9: **end for**

Independence step enables every agent to start an optimization pipeline simultaneously without considering the influence of other agents. It is the basis to achieve a general-purpose parallel update scheme, which will be discussed in Subsection 4.3. However, independence step could cause deviation from the joint optimization objective (i.e., $A_{\boldsymbol{\pi}}(s, \boldsymbol{a})$) since the objective is just the first part of it. Dependence step helps to correct the deviation from the joint optimization objective and also establish an important interaction between different agents so that their updates can be coordinated.

4.3 The Ultimate Algorithm

Finally, we parameterize the policies, and let $\pi_{\theta^{i_p}}^{i_p}$ be the policy network of agent i_p with parameter vector θ^{i_p}, $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ be the joint policy with parameter vector $\boldsymbol{\theta} = \{\theta^1, ..., \theta^n\}$. Then, we overload the previous notation $\{\pi^1, ..., \pi^n\}$ with parameterized policy $\{\pi_{\theta^1}^1, ..., \pi_{\theta^n}^n\}$. θ_{old} denotes the old policy parameter set that we want to improve, θ_{mid} denotes the middle policy parameter set generated by independence step, and θ_{new} denotes the new policy parameter set generated by dependence step. For $M_{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\text{old}}}}^{-i_p}(\pi_{\theta_{\text{mid}}^{i_p}}^{i_p}, \boldsymbol{\pi}_{\boldsymbol{\theta}^{-i_p}}^{-i_p})$ of pipeline p, the terms $-\pi_{\theta^{i_p}}^{i_p}(a^{i_p}|s)/\pi_{\theta_{\text{old}}^{i_p}}^{i_p}(a^{i_p}|s)$ in Equation (7),

$\sum_{i_k \in -i_p - j_p} \mathcal{C} D_{\mathrm{KL}}^{\max}(\pi_{\theta_{\mathrm{old}}^{i_k}}^{i_k}, \pi_{\theta^{i_k}}^{i_k})$ can be omitted since they, as constant terms, have no contribution to the gradients of θ^{j_p} (see Appendix B.5). Thus, $\underset{\theta^{j_p}}{\text{maximize}}\, M_{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\mathrm{old}}}}^{-i_p}(\pi_{\theta_{\mathrm{mid}}^{i_p}}^{i_p}, \boldsymbol{\pi}_{\boldsymbol{\theta}^{-i_p}}^{-i_p})$ can be simplified to

$$\underset{\theta^{j_p}}{\text{maximize}} \left[\mathbb{E}_{s,\boldsymbol{a}} \left[\frac{\pi_{\theta^{j_p}}^{j_p}(a^{j_p}|s)}{\pi_{\theta_{\mathrm{old}}^{j_p}}^{j_p}(a^{j_p}|s)} \cdot \frac{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\mathrm{mid}}^{-j_p}}^{-j_p}(\boldsymbol{a}^{-j_p}|s)}{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\mathrm{old}}^{-j_p}}^{-j_p}(\boldsymbol{a}^{-j_p}|s)} A_{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\mathrm{old}}}}(s, \boldsymbol{a}) \right] - \mathcal{C} D_{\mathrm{KL}}^{\max}(\pi_{\theta_{\mathrm{old}}^{j_p}}^{j_p}, \pi_{\theta^{j_p}}^{j_p}) \right], \tag{8}$$

where $s \sim \rho_{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\mathrm{old}}}}, \boldsymbol{a} \sim \boldsymbol{\pi}_{\boldsymbol{\theta}_{\mathrm{old}}}$. We replace KL divergence term with the clipping mechanism following Schulman et al. (2017), and propose our *Full-Pipeline PPO* (FP3O), an efficient implementation of Algorithm 2, as follows: for pipelines $p = 1, ..., n$, we obtain the middle policy parameters (i.e., $\theta_{\mathrm{mid}}^{i_1}, ..., \theta_{\mathrm{mid}}^{i_n}$) at independence step by

$$\theta_{\mathrm{mid}}^{i_p} = \arg\max_{\theta^{i_p}} \mathbb{E}_{s,\boldsymbol{a}} \left[\min \left(\frac{\pi_{\theta^{i_p}}^{i_p}(a^{i_p}|s)}{\pi_{\theta_{\mathrm{old}}^{i_p}}^{i_p}(a^{i_p}|s)} A_{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\mathrm{old}}}}(s, \boldsymbol{a}), \mathrm{clip}\Big(\frac{\pi_{\theta^{i_p}}^{i_p}(a^{i_p}|s)}{\pi_{\theta_{\mathrm{old}}^{i_p}}^{i_p}(a^{i_p}|s)}, 1 \pm \epsilon \Big) A_{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\mathrm{old}}}}(s, \boldsymbol{a}) \right) \right], \tag{9}$$

and we obtains the final new policy parameters (i.e., $\theta_{\mathrm{new}}^{j_1}, ..., \theta_{\mathrm{new}}^{j_n}$) at dependence step by

$$\begin{aligned} \theta_{\mathrm{new}}^{j_p} = \arg\max_{\theta^{j_p}} \mathbb{E}_{s,\boldsymbol{a}} \Bigg[\min \Bigg(& \frac{\pi_{\theta^{j_p}}^{j_p}(a^{j_p}|s)}{\pi_{\theta_{\mathrm{old}}^{j_p}}^{j_p}(a^{j_p}|s)} \cdot \frac{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\mathrm{mid}}^{-j_p}}^{-j_p}(\boldsymbol{a}^{-j_p}|s)}{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\mathrm{old}}^{-j_p}}^{-j_p}(\boldsymbol{a}^{-j_p}|s)} A_{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\mathrm{old}}}}(s, \boldsymbol{a}), \\ & \mathrm{clip}\Big(\frac{\pi_{\theta^{j_p}}^{j_p}(a^{j_p}|s)}{\pi_{\theta_{\mathrm{old}}^{j_p}}^{j_p}(a^{j_p}|s)}, 1 \pm \epsilon \Big) \cdot \frac{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\mathrm{mid}}^{-j_p}}^{-j_p}(\boldsymbol{a}^{-j_p}|s)}{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\mathrm{old}}^{-j_p}}^{-j_p}(\boldsymbol{a}^{-j_p}|s)} A_{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\mathrm{old}}}}(s, \boldsymbol{a}) \Big) \Bigg]. \end{aligned} \tag{10}$$

FP3O enables all policy network parameters (i.e., $\theta^1, ..., \theta^n$) to be optimized in parallel with all sampling data in one epsilon, which is a highly general update scheme. To elaborate, let's consider that there are associated parameters θ^* between $\theta^1, ..., \theta^n$. In the sequential update scheme of HAPPO, θ^* will be cumulatively changed as agents are updated one by one. The changed θ^* can easily cause the ratio $\pi_{\theta^{j_p}}^{j_p}(a^{j_p}|s) / \pi_{\theta_{\mathrm{old}}^{j_p}}^{j_p}(a^{j_p}|s)$ of the latter agents to reach or even exceed the clipping boundary (i.e., $1 - \epsilon$ or $1 + \epsilon$), thus resulting in the problems of uncontrolled KL divergence and poor data efficiency. In FP3O, any iterative optimization of θ^* can utilize all sampling data to update all agents simultaneously thus avoiding the accumulated policy change and improving the data efficiency.

4.4 Additional Analysis on The Monotonic Improvement of FP3O

In the previous subsection, we carried out a series of operations (non-overlapping selection, importance sampling, pipeline interaction) to develop full-pipeline optimization from the theoretical level to the practical level, then derived our FP3O algorithm. To prove that these practical operations and the mutual influence between different pipelines in FP3O will not break the monotonic improvement guarantee, we first convert FP3O to the KL penalty version similar to TRPO (Schulman et al., 2017):

$$\left\{ \hat{\pi}^{i_p} = \arg\max_{\hat{\pi}^{i_p}} \left[\mathbb{E}_{s \sim \rho_{\boldsymbol{\pi}}, a^{i_p} \sim \hat{\pi}^{i_p}} \left[A_{\boldsymbol{\pi}}^{i_p}(s, a^{\varnothing}, a^{i_p}) \right] - \beta D_{\mathrm{KL}}^{\max}(\pi^{i_p}, \hat{\pi}^{i_p}) \right] \right\}_{p=1}^{n}, \tag{11}$$

$$\left\{ \tilde{\pi}^{j_p} = \arg\max_{\tilde{\pi}^{j_p}} \left[\mathbb{E}_{s \sim \rho_{\boldsymbol{\pi}}, a^{j_p} \sim \tilde{\pi}^{j_p}, \boldsymbol{a}^{-j_p} \sim \hat{\boldsymbol{\pi}}^{-j_p}} \left[A_{\boldsymbol{\pi}}^{-i_p}(s, a^{i_p}, \boldsymbol{a}^{-i_p}) \right] - \beta D_{\mathrm{KL}}^{\max}(\pi^{j_p}, \tilde{\pi}^{j_p}) \right] \right\}_{p=1}^{n}, \tag{12}$$

where β is the penalty coefficient corresponding to ϵ of the clipping mechanism. Note that for the theoretical analysis, we do not consider parameterized policies, and use the more rigorous max KL divergence instead of the mean KL divergence used in TRPO. Then we give the following theorem:

Theorem 3. *We let $\pi^1, ..., \pi^n$ denote the old policies. $\hat{\pi}^1, ..., \hat{\pi}^n$ and $\tilde{\pi}^1, ..., \tilde{\pi}^n$ are the middle policies and new policies, which are obtained from all pipelines (i.e., $p = 1, ..., n$) by Equation (11) and Equation (12) respectively. $\mathcal{L}_{\boldsymbol{\pi}}(\tilde{\pi}^{i_p}, \tilde{\boldsymbol{\pi}}^{-i_p}) = \mathcal{J}(\boldsymbol{\pi}) + M_{\boldsymbol{\pi}}^{i_p}(\tilde{\pi}^{\varnothing}, \tilde{\pi}^{i_p}) + M_{\boldsymbol{\pi}}^{-i_p}(\tilde{\pi}^{i_p}, \tilde{\boldsymbol{\pi}}^{-i_p})$ is the theoretical lower bound of pipeline p defined in Theorem 2. Given the assumption described in Appendix B.6, the following inequalities hold for all pipeline $p = 1, ..., n$:*

$$\left\{ \mathcal{L}_{\boldsymbol{\pi}}(\tilde{\pi}^{i_p}, \tilde{\boldsymbol{\pi}}^{-i_p}) \geq \mathcal{L}_{\boldsymbol{\pi}}(\hat{\pi}^{i_p}, \hat{\boldsymbol{\pi}}^{-i_p}) \geq \mathcal{L}_{\boldsymbol{\pi}}(\pi^{i_p}, \boldsymbol{\pi}^{-i_p}) \right\}_{p=1}^{n}.$$

The proof is in Appendix B.6. When jointly updating all pipelines, Theorem 3 guarantees that the lower bounds can be monotonically improved in FP3O after considering non-overlapping selection, pipeline interaction and the mutual influence between different pipelines on the joint policy change.

5 EXPERIMENTS

Environments and baselines. We aim to evaluate our algorithm on various types of cooperative multi-agent challenges, thus choosing 1) Multi-Agent MuJoCo (MAMuJoCo) (de Witt et al., 2020b) which involves heterogeneous agents with continuous actions; 2) StarCraftII Multi-Agent Challenge (SMAC) (Samvelyan et al., 2019) which involves both homogeneous and heterogeneous agents with discrete actions. More details of MAMuJoCo and SMAC are in Appendix C. We compare FP3O to existing state-of-the-art PPO-based algorithms in cooperative MARL including HAPPO (Kuba et al., 2022), MAPPO (Yu et al., 2021) and IPPO (de Witt et al., 2020a). For a fair comparison, the settings of all algorithms are identical including all hyperparameters and network inputs.

Network types. We will adopt the common network types including full parameter sharing (FuPS), partial parameter sharing (PaPS) and non-parameter sharing (NoPS) to evaluate the generality of our algorithm. FuPS means that different agents' policy networks share the same set of parameters, which is widely used to improve learning efficiency and reduce parameters, especially in the case of large numbers of agents. For NoPS, each agent has an individual policy network, which enables agents to make personalized decisions, especially for heterogeneous agents. PaPS combines the properties of full parameter sharing and non-parameter sharing, and its implementation is diverse (Chenghao et al., 2021; Christianos et al., 2021). In our experiments, we let the last layers of different agents' policy networks have individual parameters enabling agents to make personalized actions, while other layers share the same set of parameters to improve learning efficiency.

Performances on MAMuJoCo We select six representative tasks (i.e., 2-Agent Reacher [2×1], 2-Agent Ant [2×4], 2-Agent Walker [2×3], 3-Agent Hopper [3×1], 6-Agent HalfCheetah [6×1], Manyagent Swimmer [8×2]), which efficiently cover most types of robotic control in MAMuJoCo. Then, we evaluate our algorithm on these tasks with FuPS, PaPS and NoPS network types. More details and hyperparameter settings are in Appendix D.1. The results are shown in Table 1. It is worth noting that the best performances of each algorithm mostly occur in the PaPS and NoPS network types, which demonstrates the heterogeneity of MAMuJoCo agents and the necessity of PaPS and the NoPS for such tasks. IPPO fails to learn effective policies in several tasks (e.g., 2-agent Walker and Manyagent Swimmer) while MAPPO can achieve more stable performances benefitting from the CTDE framework. However, both of them cannot perform well on these complex robotic control tasks due to the lack of theoretical support. HAPPO with NoPS establishes the strongest baseline on these heterogeneous-agent tasks in MAMuJoCo as indicated by the underlined values of NoPS case in Table 1. Unfortunately, it is not general enough and cannot maintain its advantages under

Table 1: The average evaluation rewards and standard deviations on MAMuJoCo. We bold the best performances and underline the second best performances. HE denotes heterogeneous agents.

Network	Task	FP3O	HAPPO	MAPPO	IPPO	Agent
FuPS	2-Agent Reacher [2×1]	**-29.3**±4.2	-33.2±2.2	-45.4±11.3	-51.5±4.5	HE
	2-Agent Ant [2×4]	**2536.3**±97.8	2296.2±181.3	1939.3±52.1	1584.8±192.3	HE
	2-Agent Walker [2×3]	**2410.5**±459.9	907.0±280.8	1684.3±858.7	523.4±208.4	HE
	3-Agent Hopper [3×1]	3545.9±148.2	**3554.7**±118.3	3373.3±238.3	2203.7±855.0	HE
	6-Agent HalfCheetah [6×1]	**4142.0**±362.9	4058.8±129.7	3170.5±116.6	2683.5±431.3	HE
	Manyagent Swimmer [8×2]	**414.8**±39.7	404.9±38.9	409.2±48.3	63.6±21.8	HE
PaPS	2-Agent Reacher [2×1]	-35.6±3.8	-36.7±4.1	**-33.4**±4.8	-36.1±1.9	HE
	2-Agent Ant [2×4]	**3129.4**±186.4	2809.5±61.9	2306.7±255.4	2026.6±207.2	HE
	2-Agent Walker [2×3]	**3161.1**±790.9	2717.4±581.7	2748.7±734.1	651.7±266.0	HE
	3-Agent Hopper [3×1]	**2981.1**±817.7	1547.0±321.7	2678.1±997.8	2486.8±712.3	HE
	6-Agent HalfCheetah [6×1]	**4593.8**±759.5	4413.7±840.0	4432.6±196.1	4336.1±768.7	HE
	Manyagent Swimmer [8×2]	**391.7**±30.5	353.9±15.4	366.0±27.0	76.1±2.8	HE
NoPS	2-Agent Reacher [2×1]	**-28.2**±2.0	-34.9±4.3	-31.4±6.4	-31.7±5.4	HE
	2-Agent Ant [2×4]	**2824.8**±353.6	2043.0±187.7	1825.4±157.0	1469.9±36.1	HE
	2-Agent Walker [2×3]	**3678.0**±238.6	2461.8±368.3	2303.2±815.9	405.6±107.9	HE
	3-Agent Hopper [3×1]	**3565.1**±109.1	3422.8±207.9	3390.1±225.9	1851.4±683.1	HE
	6-Agent HalfCheetah [6×1]	**4795.6**±322.3	4408.7±170.2	4103.8±233.9	3831.3±131.5	HE
	Manyagent Swimmer [8×2]	**428.2**±14.0	317.0±12.2	303.3±44.0	97.2±42.0	HE

FuPS and PaPS settings. We will demonstrate in SMAC that this problem will be more prominent on homogeneous-agent tasks. Our FP3O significantly outperforms the baselines across all scenarios, which demonstrates the advantages in theoretically-supported performance and generality.

Performances on SMAC We evaluate our algorithm on various types of maps in SMAC with FuPS, PaPS and NoPS. We select six different maps (i.e., bane vs. bane, 2c vs. 64zg, 5m vs. 6m, 3s5z, corridor, 6h vs. 8z), which involve three difficulties, seven types of units, as well as both homogeneous and heterogeneous tasks. More details and hyperparameter settings are in Appendix D.2. The results are shown in Table 2. We can notice that MAPPO with FuPS achieves the strongest baseline on SMAC, even for heterogeneous-agent maps. This shows the homogeneity of SMAC agents and the high learning efficiency of FuPS while dealing with such tasks. However, MAPPO cannot maintain its superiority in other scenarios, which further verifies its lack of theoretical performance guarantee. Moreover, due to the problems of uncontrolled KL divergence and poor data efficiency brought by the sequential update scheme, we can see that HAPPO with FuPS and PaPS will suffer from serious performance degradation in most maps, especially in a large number of agents cases (e.g., HAAPO with PaPS has only 65.6% win rate in *easy* map bane vs. bane, and 0.0% win rate in map 3s5z). This further demonstrates the limitation of HAPPO's application. Our FP3O again gets the superior performances on all task types and network types, clearly demonstrating its advantages in theoretically-supported performance and generality.

Table 2: The median evaluation win rates and standard deviations on SMAC. We bold the best performances and underline the second best performances. HO and HE denote homogeneous and heterogeneous agents respectively.

Network	Scenario	Difficulty	FP3O	HAPPO	MAPPO	IPPO	Agent
FuPS	bane vs. bane	Easy	**100.0**$\pm$1.3	**100.0**$\pm$1.9	**100.0**$\pm$3.5	**100.0**$\pm$2.3	HE
	2c vs. 64zg	Hard	**100.0**$\pm$3.4	96.9$\pm$2.4	**100.0**$\pm$2.5	96.9$\pm$2.3	HO
	3s5z	Hard	**100.0**$\pm$1.2	87.5$\pm$24.7	<u>90.6</u>$\pm$18.5	87.5$\pm$41.6	HE
	5m vs. 6m	Hard	**93.8**$\pm$7.3	18.8$\pm$12.8	<u>81.3</u>$\pm$19.0	62.5$\pm$19.1	HO
	corridor	Super Hard	**100.0**$\pm$1.5	96.9$\pm$35.3	**100.0**$\pm$2.9	96.9$\pm$4.3	HO
	6h vs. 8z	Super Hard	**93.8**$\pm$4.2	53.1$\pm$20.2	<u>87.5</u>$\pm$12.9	<u>87.5</u>$\pm$10.4	HO
PaPS	bane vs. bane	Easy	**100.0**$\pm$0.0	65.6$\pm$26.4	**100.0**$\pm$0.0	**100.0**$\pm$0.0	HE
	2c vs. 64zg	Hard	**100.0**$\pm$3.1	96.9$\pm$3.3	**100.0**$\pm$3.1	**100.0**$\pm$4.3	HO
	3s5z	Hard	**96.9**$\pm$4.8	0.0$\pm$31.0	<u>93.8</u>$\pm$4.6	90.6$\pm$6.5	HE
	5m vs. 6m	Hard	**93.8**$\pm$8.2	31.3$\pm$15.4	87.5$\pm$7.7	<u>90.6</u>$\pm$11.1	HO
	corridor	Super Hard	<u>90.6</u>$\pm$9.2	78.1$\pm$22.7	**93.8**$\pm$10.9	<u>90.6</u>$\pm$3.6	HO
	6h vs. 8z	Super Hard	**75.0**$\pm$13.9	9.4$\pm$18.8	15.6$\pm$21.1	<u>46.9</u>$\pm$23.0	HO
NoPS	bane vs. bane	Easy	**100.0**$\pm$0.0	**100.0**$\pm$0.0	**100.0**$\pm$0.0	**100.0**$\pm$0.0	HE
	2c vs. 64zg	Hard	**100.0**$\pm$2.7	**100.0**$\pm$1.9	**100.0**$\pm$1.9	**100.0**$\pm$1.3	HO
	3s5z	Hard	**100.0**$\pm$3.1	90.6$\pm$4.8	93.8$\pm$5.3	<u>96.9</u>$\pm$8.1	HE
	5m vs. 6m	Hard	**90.6**$\pm$5.2	<u>68.8</u>$\pm$16.8	62.5$\pm$10.0	62.5$\pm$11.3	HO
	corridor	Super Hard	**96.9**$\pm$6.1	**96.9**$\pm$3.0	93.8$\pm$7.2	93.8$\pm$27.2	HO
	6h vs. 8z	Super Hard	**84.4**$\pm$14.7	43.8$\pm$10.2	43.8$\pm$15.8	<u>46.9</u>$\pm$20.5	HO

6 Conclusion

Although PPO algorithm has got remarkable successes in single-agent RL, its existing extensions in cooperative MARL fail to achieve the compatibility between the theoretical monotonic improvement and generality. In this paper, we propose the theoretically-justified and general-purpose FP3O algorithm, the key of which is to dynamically allocate agents to different one-separation trust region optimization pipelines to be updated. FP3O enjoys the proved monotonic policy improvement guarantee and is general enough since it avoids the restrictive factors that could arise in other existing multi-agent PPO algorithms. Experiments on MAMuJoCo and SMAC demonstrate the superior performances of FP3O on continuous and discrete, homogeneous and heterogeneous-agent tasks with three different network types. This ability of FP3O provides us with great opportunities to explore its potential in more multi-agent fields such as offline settings (Levine et al., 2020) and its application on elaborately designed networks (Vaswani et al., 2017; Fu et al., 2022; Wen et al., 2022).

7 REPRODUCIBILITY STATEMENT

By non-overlapping selection, importance sampling, pipeline interaction in Subsection 4.2 and the parameterized policy networks and the clipping mechanism in Subsection 4.3, we develop a programming-friendly FP3O algorithm. We provide the source code in the supplementary material. After the review process, the code will also be released online. We specify all training details (e.g. hyperparameter settings), repetitive experiments (e.g., the number of random seeds), and the type of computing infrastructure (e.g. the type of GPUs) in Appendix D.

REFERENCES

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47: 253–279, 2013.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Yu-Han Chang, Tracey Ho, and Leslie Kaelbling. All learning is local: Multi-agent learning in global reward games. *Advances in Neural Information Processing Systems*, 16, 2003.

Li Chenghao, Tonghan Wang, Chengjie Wu, Qianchuan Zhao, Jun Yang, and Chongjie Zhang. Celebrating diversity in shared multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34:3991–4002, 2021.

Filippos Christianos, Georgios Papoudakis, Muhammad A Rahman, and Stefano V Albrecht. Scaling multi-agent reinforcement learning with selective parameter sharing. In *International Conference on Machine Learning*, pp. 1989–1998. PMLR, 2021.

Christian Schroeder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020a.

Christian Schroeder de Witt, Bei Peng, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. Deep multi-agent reinforcement learning for decentralized continuous cooperative control. *arXiv preprint arXiv:2003.06709*, 2020b.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pp. 1329–1338. PMLR, 2016.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on Artificial Intelligence*, volume 32, 2018.

Wei Fu, Chao Yu, Zelai Xu, Jiaqi Yang, and Yi Wu. Revisiting some common practices in cooperative multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 6863–6877. PMLR, 2022.

Pablo Hernandez-Leal, Michael Kaisers, Tim Baarslag, and Enrique Munoz de Cote. A survey of learning in multiagent environments: Dealing with non-stationarity. *arXiv preprint arXiv:1707.09183*, 2017.

Jakub Grudzien Kuba, Ruiqing Chen, Muning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. Trust region policy optimisation in multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2022.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Hepeng Li and Haibo He. Multi-agent trust region policy optimization. *arXiv preprint arXiv:2010.07916*, 2020.

Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning Proceedings 1994*, pp. 157–163. Elsevier, 1994.

Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in Neural Information Processing Systems*, 30, 2017.

Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. Maven: Multi-agent variational exploration. *Advances in Neural Information Processing Systems*, 32, 2019.

Georgios Papoudakis, Filippos Christianos, Arrasy Rahman, and Stefano V Albrecht. Dealing with non-stationarity in multi-agent deep reinforcement learning. *arXiv preprint arXiv:1906.04737*, 2019.

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*, pp. 4295–4304. PMLR, 2018.

Mikayel Samvelyan, Tabish Rashid, Christian Schröder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob N. Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2186–2188, 2019.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897. PMLR, 2015.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *4th International Conference on Learning Representations*, 2016.

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

Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 5887–5896. PMLR, 2019.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinícius Flores Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2085–2087, 2018.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. QPLEX: duplex dueling multi-agent q-learning. In *9th International Conference on Learning Representations*. OpenReview.net, 2021a.

Tonghan Wang, Tarun Gupta, Anuj Mahajan, Bei Peng, Shimon Whiteson, and Chongjie Zhang. RODE: learning roles to decompose multi-agent tasks. In *9th International Conference on Learning Representations*. OpenReview.net, 2021b.

Yihan Wang, Beining Han, Tonghan Wang, Heng Dong, and Chongjie Zhang. DOP: off-policy multi-agent decomposed policy gradients. In *9th International Conference on Learning Representations*. OpenReview.net, 2021c.

Muning Wen, Jakub Grudzien Kuba, Runji Lin, Weinan Zhang, Ying Wen, Jun Wang, and Yaodong Yang. Multi-agent reinforcement learning is a sequence modeling problem. *arXiv preprint arXiv:2205.14953*, 2022.

Ying Wen, Hui Chen, Yaodong Yang, Zheng Tian, Minne Li, Xu Chen, and Jun Wang. A game-theoretic approach to multi-agent trust region optimization. *arXiv preprint arXiv:2106.06828*, 2021.

Zifan Wu, Chao Yu, Deheng Ye, Junge Zhang, Hankz Hankui Zhuo, et al. Coordinated proximal policy optimization. *Advances in Neural Information Processing Systems*, 34:26437–26448, 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.

A THE LIMITATION OF SEQUENTIAL UPDATE SCHEME

If the policy networks of agents in HAPPO are associated with each other (e.g., full parameter sharing, partial parameter sharing or other associative design), a problem will arise: the former agents' update will cause the latter agents' policy networks to be changed. These changes will be accumulated as the sequential update scheme progresses. As a result, even though the latter agents have not initiated their update rounds, their policies are very likely to have reached or even exceeded the boundary of the KL divergence trust region causing the sampling data of the latter agents to have little positive impact on the network optimization. To verify this, we train HAPPO with full parameter sharing network type on 3s5z map of StarCraftII Multi-Agent Challenge (Samvelyan et al., 2019). When each agent starts the update round at one policy iteration, we record the distribution of ratio $\pi^{i_m}_{\theta^{i_m}}(\boldsymbol{a}^i|s)/\pi^{i_m}_{\theta^{i_m}_k}(\boldsymbol{a}^i|s)$ (see Equation (11) in Kuba et al. (2022)), which is demonstrated in Figure 1. PPO-clip operator $\text{clip}(\pi^{i_m}_{\theta^{i_m}}(\boldsymbol{a}^i|s)/\pi^{i_m}_{\theta^{i_m}_k}(\boldsymbol{a}^i|s), 1 \pm \epsilon)$ is used to restrict the optimization step

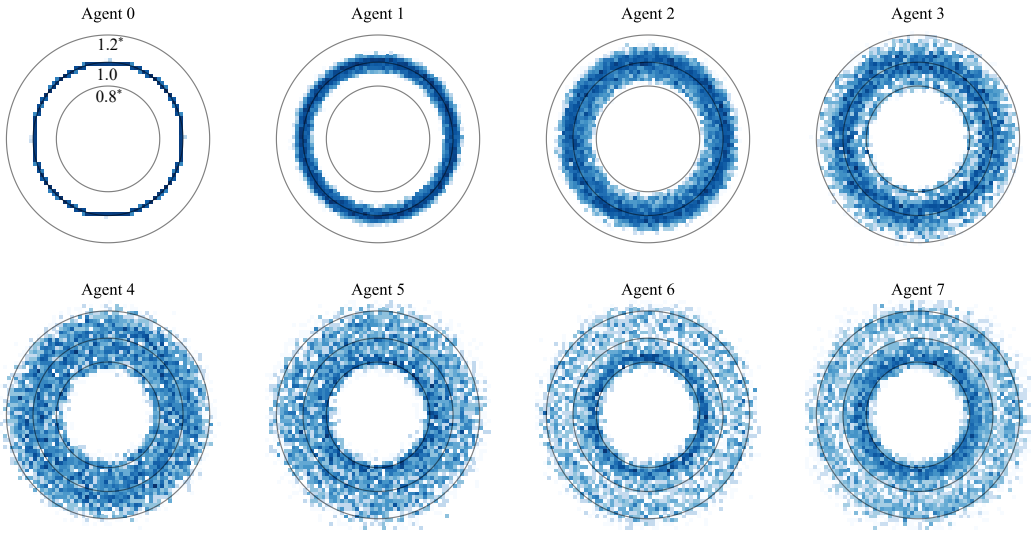


Figure 1: The distribution of ratio $\pi^{i_m}_{\theta^{i_m}}(\boldsymbol{a}^i|s)/\pi^{i_m}_{\theta^{i_m}_k}(\boldsymbol{a}^i|s)$ of each agent. The places where the ratio equals 0.8, 1.0 and 1.2 are indicated by the black circles. * denotes the clipping boundary $1 \pm \epsilon$, where $\epsilon = 0.2$.

size to a trust region by clipping the ratio $\pi^{i_m}_{\theta^{i_m}}(\boldsymbol{a}^i|s)/\pi^{i_m}_{\theta^{i_m}_k}(\boldsymbol{a}^i|s)$, and we set $\epsilon = 0.2$. Since the former agents' update will cause the latter agents' policies to be changed, we can see in Figure 1 that there will be a gap (ratio not equal to 1) between the updated policies and the old policies before the optimization for all agents except agent 0. Clearly, this policy change will be accumulated as agents are updated one by one. For agents 4, 5, 6 and 7, a great proportion of ratios have reached or even exceeded the clipping boundary 0.8 and 1.2 before their optimization, which will make the sampling data have little positive impact on the network optimization.

In addition, when each agent finishes its update round at one policy iteration, we record the distribution of KL divergence between its updated policy and the old one, which is demonstrated in Figure 2. We can notice that due to the accumulated policy changes, the later the agent is updated, the greater the KL divergence between its updated policy and the old policy. Especially, for agents 6 and 7, the KL divergence between the new policy and the old policy has been too large, and this uncontrolled KL divergence is very harmful to the trust region optimization.

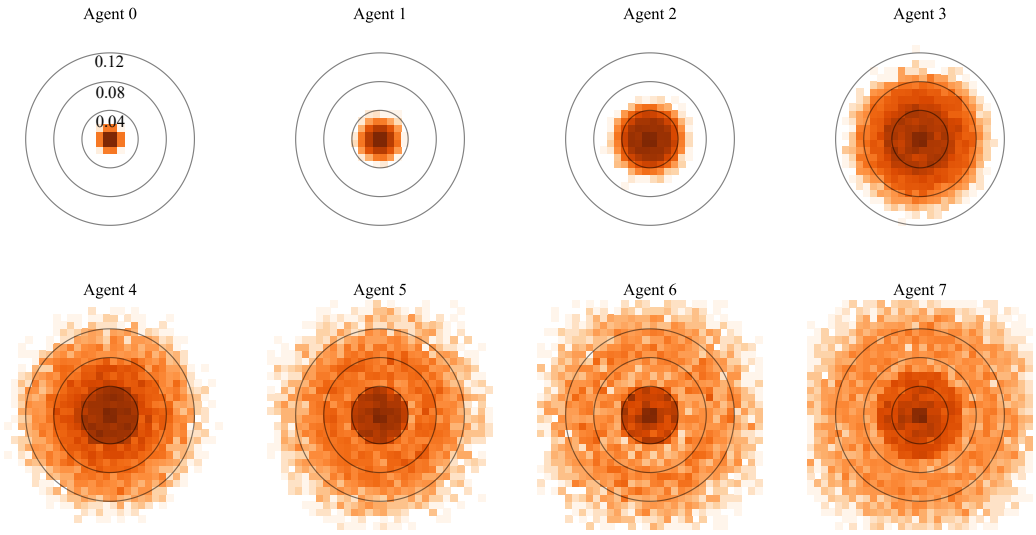


Figure 2: The distribution of KL divergence between the updated policy and the old policy of each agent in HAPPO. The darker the color is, the more data is distributed. The center represents that KL divergence is equal to 0, and the outward direction represents that KL divergence gets greater. The places where KL divergence equals 0.04, 0.08, and 0.12 are indicated by the black circles.

B PROOFS

B.1 PROOF OF ONE-SEPARATION ADVANTAGE DECOMPOSITION

We separate one expected agent's contribution independently of other agents from the team through Lemma 1. Now we give its derivation.

Lemma 1. *(One-Separation Advantage Decomposition) Let $\varnothing$ denotes empty set. In the cooperative multi-agent tasks, for $\forall i_p \in i_{1:n} = \mathcal{N}$, the joint advantage funtion $A_{\boldsymbol{\pi}}(s, \boldsymbol{a})$ can be decomposed as*

$$A_{\boldsymbol{\pi}}(s, \boldsymbol{a}) = A^{i_p}_{\boldsymbol{\pi}}(s, a^{\varnothing}, a^{i_p}) + A^{-i_p}_{\boldsymbol{\pi}}(s, a^{i_p}, \boldsymbol{a}^{-i_p}).$$

Proof. Based on Equations (1), (2), we have

$$\begin{aligned}
&A^{i_p}_{\boldsymbol{\pi}}(s, a^{\varnothing}, a^{i_p}) + A^{-i_p}_{\boldsymbol{\pi}}(s, a^{i_p}, \boldsymbol{a}^{-i_p}), \\
&= \left(Q^{i_p}_{\boldsymbol{\pi}}(s, \boldsymbol{a}^{i_p}) - V_{\boldsymbol{\pi}}(s)\right) - \left(Q^{i_p, -i_p}_{\boldsymbol{\pi}}(s, \boldsymbol{a}^{i_p, -i_p}) - Q^{i_p}_{\boldsymbol{\pi}}(s, \boldsymbol{a}^{i_p})\right) \\
&= Q^{i_p, -i_p}_{\boldsymbol{\pi}}(s, \boldsymbol{a}^{i_p, -i_p}) - V_{\boldsymbol{\pi}}(s) \\
&= Q_{\boldsymbol{\pi}}(s, \boldsymbol{a}) - V_{\boldsymbol{\pi}}(s) \\
&= A_{\boldsymbol{\pi}}(s, \boldsymbol{a}).
\end{aligned}$$

□

B.2 PROOF OF THEOREM 2

Theorem 1. *(Schulman et al., 2015) Let π denote the old policy, $\tilde{\pi}$ denote the new policy that is obtained by optimizing π, $L_{\pi}(\tilde{\pi}) = \mathcal{J}(\pi) + \mathbb{E}_{s\sim\rho_{\pi}, a\sim\tilde{\pi}}[A_{\pi}(s, a)]$, $D^{\max}_{\mathrm{KL}}(\pi, \tilde{\pi}) = \max_s D_{\mathrm{KL}}(\pi(\cdot|s), \tilde{\pi}(\cdot|s))$, $C = \frac{4\gamma \max_{s,a} |A_{\pi}(s,a)|}{(1-\gamma)^2}$. The following policy improvement bound holds:*

$$\mathcal{J}(\tilde{\pi}) \geq L_{\pi}(\tilde{\pi}) - C D^{\max}_{\mathrm{KL}}(\pi, \tilde{\pi}).$$

Proof. See Appendix A in Schulman et al. (2015). □

Theorem 2 describes the lower bound of the true objective in cooperative multi-agent domains based on one-separation advantage decomposition. Before proving this theorem, we first give the following lemma to describe the relationship between the maximum KL divergence of joint policy and the maximum KL divergence of an individual policy.

Lemma 2. *(Kuba et al., 2022) For the old joint policy $\boldsymbol{\pi} = \prod_{k=1}^{n} \pi^k$ and the new joint policy $\tilde{\boldsymbol{\pi}} = \prod_{k=1}^{n} \tilde{\pi}^k$, the following inequality holds:*

$$D_{\mathrm{KL}}^{\max}(\boldsymbol{\pi}, \tilde{\boldsymbol{\pi}}) \leq \sum_{k=1}^{n} D_{\mathrm{KL}}^{\max}(\pi^k, \tilde{\pi}^k).$$

Proof. See Lemma 8 in Kuba et al. (2022). □

Then, we give the derivation of Theorem 2 as follows.

Theorem 2. *Let $\boldsymbol{\pi}$ denote the old joint policy, $\tilde{\boldsymbol{\pi}}$ denote the new joint policy that is obtained by optimizing $\boldsymbol{\pi}$. For $\forall i_p \in i_{1:n}$, the following lower bound holds.*

$$\mathcal{J}(\tilde{\boldsymbol{\pi}}) \geq \mathcal{J}(\boldsymbol{\pi}) + M_{\boldsymbol{\pi}}^{i_p}(\tilde{\pi}^{\varnothing}, \tilde{\pi}^{i_p}) + M_{\boldsymbol{\pi}}^{-i_p}(\tilde{\pi}^{i_p}, \tilde{\boldsymbol{\pi}}^{-i_p}).$$

Proof. we give the equivalent form of Theorem 1 in multi-agent domains:

$$\begin{aligned}
&\mathcal{J}(\tilde{\boldsymbol{\pi}}) \geq L_{\boldsymbol{\pi}}(\tilde{\boldsymbol{\pi}}) - \mathcal{C} D_{\mathrm{KL}}^{\max}(\boldsymbol{\pi}, \tilde{\boldsymbol{\pi}}) \\
&= \mathcal{J}(\boldsymbol{\pi}) + \mathbb{E}_{s \sim \rho_{\boldsymbol{\pi}}, \boldsymbol{a} \sim \tilde{\boldsymbol{\pi}}}[A_{\boldsymbol{\pi}}(s, \boldsymbol{a})] - \mathcal{C} D_{\mathrm{KL}}^{\max}(\boldsymbol{\pi}, \tilde{\boldsymbol{\pi}}),
\end{aligned}$$

which, by decomposing the joint advantage function $A_{\boldsymbol{\pi}}(s, \boldsymbol{a})$ according to Lemma 1, equals

$$= \mathcal{J}(\boldsymbol{\pi}) + \mathbb{E}_{s \sim \rho_{\boldsymbol{\pi}}, \boldsymbol{a} \sim \tilde{\boldsymbol{\pi}}}[A_{\boldsymbol{\pi}}^{i_p}(s, a^{i_p}) + A_{\boldsymbol{\pi}}^{-i_p}(s, a^{i_p}, \boldsymbol{a}^{-i_p})] - \mathcal{C} D_{\mathrm{KL}}^{\max}(\boldsymbol{\pi}, \tilde{\boldsymbol{\pi}}),$$

then, introducing Lemma 2, the above equation satisfies

$$\begin{aligned}
&\geq \mathcal{J}(\boldsymbol{\pi}) + \mathbb{E}_{s \sim \rho_{\boldsymbol{\pi}}, \boldsymbol{a} \sim \tilde{\boldsymbol{\pi}}}[A_{\boldsymbol{\pi}}^{i_p}(s, a^{i_p}) + A_{\boldsymbol{\pi}}^{-i_p}(s, a^{i_p}, \boldsymbol{a}^{-i_p})] - \sum_{k=1}^{n} \mathcal{C} D_{\mathrm{KL}}^{\max}(\pi^{i_k}, \tilde{\pi}^{i_k}) \\
&= \mathcal{J}(\boldsymbol{\pi}) + \mathbb{E}_{s \sim \rho_{\boldsymbol{\pi}}, \boldsymbol{a} \sim \tilde{\boldsymbol{\pi}}}[A_{\boldsymbol{\pi}}^{i_p}(s, a^{i_p})] - \mathcal{C} D_{\mathrm{KL}}^{\max}(\pi^{i_p}, \tilde{\pi}^{i_p}) \\
&\qquad + \mathbb{E}_{s \sim \rho_{\boldsymbol{\pi}}, \boldsymbol{a} \sim \tilde{\boldsymbol{\pi}}}[A_{\boldsymbol{\pi}}^{-i_p}(s, a^{i_p}, \boldsymbol{a}^{-i_p})] - \sum_{i_k \in -i_p} \mathcal{C} D_{\mathrm{KL}}^{\max}(\pi^{i_k}, \tilde{\pi}^{i_k}),
\end{aligned}$$

and by Definition 1, this is

$$= \mathcal{J}(\boldsymbol{\pi}) + M_{\boldsymbol{\pi}}^{i_p}(\tilde{\pi}^{\varnothing}, \tilde{\pi}^{i_p}) + M_{\boldsymbol{\pi}}^{-i_p}(\tilde{\pi}^{i_p}, \tilde{\boldsymbol{\pi}}^{-i_p}).$$

□

B.3 Proof of independence-dependence optimization enabling monotonic improvement

Property 1. *For any $\tilde{\pi}^{i_p}$, the surrogate objective defined in Definition 1 satisfies*

$$M_{\boldsymbol{\pi}}^{i_{1:m}-i_p}(\tilde{\pi}^{i_p}, \boldsymbol{\pi}^{i_{1:m}-i_p}) = 0.$$

Proof. According to Definition 1, we have

$$\begin{aligned}
&M_{\boldsymbol{\pi}}^{i_{1:m}-i_p}(\tilde{\pi}^{i_p}, \boldsymbol{\pi}^{i_{1:m}-i_p}) \\
&= \mathbb{E}_{s \sim \rho_{\boldsymbol{\pi}}, a^{i_p} \sim \tilde{\pi}^{i_p}, \boldsymbol{a}^{i_{1:m}-i_p} \sim \boldsymbol{\pi}^{i_{1:m}-i_p}} \left[A_{\boldsymbol{\pi}}^{i_{1:m}-i_p}(s, a^{i_p}, \boldsymbol{a}^{i_{1:m}-i_p}) \right] \\
&\qquad\qquad - \sum_{i_k \in i_{1:m}-i_p} C D_{\mathrm{KL}}^{\max}(\pi^{i_k}, \pi^{i_k}),
\end{aligned}$$

which, by Equations (1) and (2), equals

$$
\begin{aligned}
&= \mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}},a^{i_p}\sim\tilde{\pi}^{i_p},\boldsymbol{a}^{i_{1:m}-i_p}\sim\boldsymbol{\pi}^{i_{1:m}-i_p}}\left[Q_{\boldsymbol{\pi}}^{i_{1:m}}(s,\boldsymbol{a}^{i_{1:m}}) - Q_{\boldsymbol{\pi}}^{i_p}(s,\boldsymbol{a}^{i_p})\right] - 0,\\
&= \mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}},a^{i_p}\sim\tilde{\pi}^{i_p},\boldsymbol{a}^{i_{1:m}-i_p}\sim\boldsymbol{\pi}^{i_{1:m}-i_p}}\left[\mathbb{E}_{\boldsymbol{a}^{-i_{1:m}}\sim\boldsymbol{\pi}^{-i_{1:m}}}\left[Q_{\boldsymbol{\pi}}(s,\boldsymbol{a})\right] - \mathbb{E}_{\boldsymbol{a}^{-i_p}\sim\boldsymbol{\pi}^{-i_p}}\left[Q_{\boldsymbol{\pi}}(s,\boldsymbol{a})\right]\right]\\
&= \mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}},a^{i_p}\sim\tilde{\pi}^{i_p}}\Big[\mathbb{E}_{\boldsymbol{a}^{-i_{1:m}},i_{1:m}-i_p\sim\boldsymbol{\pi}^{-i_{1:m}},i_{1:m}-i_p}\left[Q_{\boldsymbol{\pi}}(s,\boldsymbol{a})\right]\\
&\qquad\qquad - \mathbb{E}_{\boldsymbol{a}^{-i_p},i_{1:m}-i_p\sim\boldsymbol{\pi}^{-i_p},i_{1:m}-i_p}\left[Q_{\boldsymbol{\pi}}(s,\boldsymbol{a})\right]\Big]\\
&= \mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}},a^{i_p}\sim\tilde{\pi}^{i_p}}\left[\mathbb{E}_{\boldsymbol{a}^{-i_p}\sim\boldsymbol{\pi}^{-i_p}}\left[Q_{\boldsymbol{\pi}}(s,\boldsymbol{a})\right] - \mathbb{E}_{\boldsymbol{a}^{-i_p}\sim\boldsymbol{\pi}^{-i_p}}\left[Q_{\boldsymbol{\pi}}(s,\boldsymbol{a})\right]\right]\\
&= 0.
\end{aligned}
$$

□

Property 1 describes that if the policy $\boldsymbol{\pi}^{i_{1:m}-i_p}$ has not been updated, the corresponding surrogate objective will be 0, which means that it will have no affect on the lower bound.

Then, we let $\boldsymbol{\pi}$ be the old joint policy, $\tilde{\boldsymbol{\pi}}$ be the updated joint policy, $\mathcal{L}_{\boldsymbol{\pi}}(\tilde{\pi}^{i_p},\tilde{\boldsymbol{\pi}}^{-i_p}) = \mathcal{J}(\boldsymbol{\pi}) + M_{\boldsymbol{\pi}}^{i_p}(\tilde{\pi}^{\varnothing},\tilde{\pi}^{i_p}) + M_{\boldsymbol{\pi}}^{-i_p}(\tilde{\pi}^{i_p},\tilde{\boldsymbol{\pi}}^{-i_p})$ represents the right-hand lower bound in Theorem 2. Then, we first prove that independence-dependence optimization can improve $\mathcal{L}_{\boldsymbol{\pi}}$.

At the beginning of the independence-dependence optimization ($\tilde{\pi}^1 = \pi^1, ..., \tilde{\pi}^n = \pi^n$), we have

$$
\begin{aligned}
&\mathcal{L}_{\boldsymbol{\pi}}(\pi^{i_p},\boldsymbol{\pi}^{-i_p})\\
&= \mathcal{J}(\boldsymbol{\pi}) + M_{\boldsymbol{\pi}}^{i_p}(\tilde{\pi}^{\varnothing},\pi^{i_p}) + M_{\boldsymbol{\pi}}^{-i_p}(\pi^{i_p},\boldsymbol{\pi}^{-i_p})
\end{aligned}
$$

then with Property 1, this can be written as

$$
\begin{aligned}
&= \mathcal{J}(\boldsymbol{\pi}) + 0 + 0\\
&= \mathcal{J}(\boldsymbol{\pi}).
\end{aligned}
$$

Therefore, we obtain

$$
\mathcal{L}_{\boldsymbol{\pi}}(\pi^{i_p},\boldsymbol{\pi}^{-i_p}) = \mathcal{J}(\boldsymbol{\pi}). \tag{13}
$$

At independence step, we do $\text{maximize}_{\tilde{\pi}^{i_p}} M_{\boldsymbol{\pi}}^{i_p}(\tilde{\pi}^{\varnothing},\tilde{\pi}^{i_p})$ and obtain updated policy $\tilde{\pi}^{i_p}$. Thus, we have $M_{\boldsymbol{\pi}}^{i_p}(\tilde{\pi}^{\varnothing},\tilde{\pi}^{i_p}) \geq M_{\boldsymbol{\pi}}^{i_p}(\tilde{\pi}^{\varnothing},\pi^{i_p})$. Then, the lower bound of independence step satisfies

$$
\begin{aligned}
&\mathcal{L}_{\boldsymbol{\pi}}(\tilde{\pi}^{i_p},\boldsymbol{\pi}^{-i_p})\\
&= \mathcal{J}(\boldsymbol{\pi}) + M_{\boldsymbol{\pi}}^{i_p}(\tilde{\pi}^{\varnothing},\tilde{\pi}^{i_p}) + M_{\boldsymbol{\pi}}^{-i_p}(\tilde{\pi}^{i_p},\boldsymbol{\pi}^{-i_p})
\end{aligned}
$$

then with Property 1, we get

$$
\begin{aligned}
&= \mathcal{J}(\boldsymbol{\pi}) + M_{\boldsymbol{\pi}}^{i_p}(\tilde{\pi}^{\varnothing},\tilde{\pi}^{i_p}) + 0 \qquad (14)\\
&\geq \mathcal{J}(\boldsymbol{\pi}) + M_{\boldsymbol{\pi}}^{i_p}(\tilde{\pi}^{\varnothing},\pi^{i_p})\\
&= \mathcal{J}(\boldsymbol{\pi}) + 0\\
&= \mathcal{L}_{\boldsymbol{\pi}}(\pi^{i_p},\boldsymbol{\pi}^{-i_p}).
\end{aligned}
$$

Therefore, we obtain

$$
\mathcal{L}_{\boldsymbol{\pi}}(\tilde{\pi}^{i_p},\boldsymbol{\pi}^{-i_p}) \geq \mathcal{L}_{\boldsymbol{\pi}}(\pi^{i_p},\boldsymbol{\pi}^{-i_p}). \tag{15}
$$

At dependence step, we do $\text{maximize}_{\tilde{\boldsymbol{\pi}}^{-i_p}} M_{\boldsymbol{\pi}}^{-i_p}(\tilde{\pi}^{i_p},\tilde{\boldsymbol{\pi}}^{-i_p})$ and obtain updated policies $\tilde{\boldsymbol{\pi}}^{-i_p}$. Thus, $M_{\boldsymbol{\pi}}^{-i_p}(\tilde{\pi}^{i_p},\tilde{\boldsymbol{\pi}}^{-i_p}) \geq M_{\boldsymbol{\pi}}^{-i_p}(\tilde{\pi}^{i_p},\boldsymbol{\pi}^{-i_p})$, and then the lower bound of dependence step satisfies

$$
\begin{aligned}
&\mathcal{L}_{\boldsymbol{\pi}}(\tilde{\pi}^{i_p},\tilde{\boldsymbol{\pi}}^{-i_p})\\
&= \mathcal{J}(\boldsymbol{\pi}) + M_{\boldsymbol{\pi}}^{i_p}(\tilde{\pi}^{\varnothing},\tilde{\pi}^{i_p}) + M_{\boldsymbol{\pi}}^{-i_p}(\tilde{\pi}^{i_p},\tilde{\boldsymbol{\pi}}^{-i_p})\\
&\geq \mathcal{J}(\boldsymbol{\pi}) + M_{\boldsymbol{\pi}}^{i_p}(\tilde{\pi}^{\varnothing},\tilde{\pi}^{i_p}) + M_{\boldsymbol{\pi}}^{-i_p}(\tilde{\pi}^{i_p},\boldsymbol{\pi}^{-i_p}),
\end{aligned}
$$

then according to Property 1, we have

$$
= \mathcal{J}(\boldsymbol{\pi}) + M_{\boldsymbol{\pi}}^{i_p}(\tilde{\pi}^{\varnothing},\tilde{\pi}^{i_p}) + 0,
$$

and with Equation (14), this is

$$= \mathcal{L}_{\boldsymbol{\pi}}(\tilde{\pi}^{i_p}, \boldsymbol{\pi}^{-i_p}).$$

Thus, we get

$$\mathcal{L}_{\boldsymbol{\pi}}(\tilde{\pi}^{i_p}, \tilde{\boldsymbol{\pi}}^{-i_p}) \geq \mathcal{L}_{\boldsymbol{\pi}}(\tilde{\pi}^{i_p}, \boldsymbol{\pi}^{-i_p}). \tag{16}$$

Summarily, according to Equations (15), (16), we prove that $\mathcal{L}_{\boldsymbol{\pi}}(\tilde{\pi}^{i_p}, \tilde{\boldsymbol{\pi}}^{-i_p}) \geq \mathcal{L}_{\boldsymbol{\pi}}(\tilde{\pi}^{i_p}, \boldsymbol{\pi}^{-i_p}) \geq \mathcal{L}_{\boldsymbol{\pi}}(\pi^{i_p}, \boldsymbol{\pi}^{-i_p})$, which means independence-dependence optimization can incrementally improve the lower bound.

Then, we prove that independence-dependence optimization can improve the true object $\mathcal{J}$. At each policy iteration k in Algorithm 1, we have

$$\begin{aligned}
&\mathcal{J}(\boldsymbol{\pi}_{k+1}) \geq \mathcal{L}_{\boldsymbol{\pi}_k}(\pi^{i_p}_{k+1}, \boldsymbol{\pi}^{-i_p}_{k+1}) && \text{// Theorem 2} \\
&\geq \mathcal{L}_{\boldsymbol{\pi}_k}(\pi^{i_p}_{k}, \boldsymbol{\pi}^{-i_p}_{k}) && \text{// Equation (15), (16)} \\
&= \mathcal{J}(\boldsymbol{\pi}_k) && \text{// Equation (13)}
\end{aligned}$$

Therefore, we prove that maximizing the surrogate objectives $M^{i_p}_{\boldsymbol{\pi}}(\tilde{\pi}^{\varnothing}, \tilde{\pi}^{i_p})$ and $M^{-i_p}_{\boldsymbol{\pi}}(\tilde{\pi}^{i_p}, \tilde{\boldsymbol{\pi}}^{-i_p})$ through independence-dependence optimization method can improve the lower bound and the true objective $\mathcal{J}$.

B.4 Proof of Equation (6) and Equation (7)

Dynamic action input of $A^{i_p}_{\pi}(s, a^{\varnothing}, a^{i_p})$ and $A^{-i_p}_{\boldsymbol{\pi}}(s, a^{i_p}, \boldsymbol{a}^{-i_p})$ leads to high cost for agents to maintain the advantage functions. Thus, we introduce Equations (6), (7) to address this issue. The proof is as follows.

For the advantage function term $\mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}}, a^{i_p}\sim\tilde{\pi}^{i_p}}\left[A^{i_p}_{\boldsymbol{\pi}}(s, a^{\varnothing}, a^{i_p})\right]$ in surrogate objective $M^{i_p}_{\boldsymbol{\pi}}(\tilde{\pi}^{\varnothing}, \tilde{\pi}^{i_p})$, we have:

$$\begin{aligned}
&\mathbb{E}_{a^{i_p}\sim\tilde{\pi}^{i_p}}\left[A^{i_p}_{\boldsymbol{\pi}}(s, a^{\varnothing}, a^{i_p})\right] \\
&\quad \text{which, by Equations (1), (2), equals} \\
&= \mathbb{E}_{a^{i_p}\sim\tilde{\pi}^{i_p}}\left[Q^{i_p}_{\boldsymbol{\pi}}(s, a^{i_p}) - V_{\boldsymbol{\pi}}(s)\right] \\
&= \mathbb{E}_{a^{i_p}\sim\tilde{\pi}^{i_p}, \boldsymbol{a}^{-i_p}\sim\boldsymbol{\pi}^{-i_p}}\left[\mathbb{E}_{\boldsymbol{a}^{-i_p}\sim\boldsymbol{\pi}^{-i_p}}[Q_{\boldsymbol{\pi}}(s, \boldsymbol{a})] - V_{\boldsymbol{\pi}}(s)\right] \\
&= \mathbb{E}_{a^{i_p}\sim\tilde{\pi}^{i_p}, \boldsymbol{a}^{-i_p}\sim\boldsymbol{\pi}^{-i_p}}\left[Q_{\boldsymbol{\pi}}(s, \boldsymbol{a}) - V_{\boldsymbol{\pi}}(s)\right] \\
&= \mathbb{E}_{a^{i_p}\sim\tilde{\pi}^{i_p}, \boldsymbol{a}^{-i_p}\sim\boldsymbol{\pi}^{-i_p}}\left[A_{\boldsymbol{\pi}}(s, \boldsymbol{a})\right], \\
&\quad \text{then with importance sampling, this is} \\
&= \mathbb{E}_{\boldsymbol{a}\sim\boldsymbol{\pi}}\left[\frac{\tilde{\pi}^{i_p}(a^{i_p}|s)}{\pi^{i_p}(a^{i_p}|s)} A_{\boldsymbol{\pi}}(s, \boldsymbol{a})\right].
\end{aligned}$$

For the advantage function term $\mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}},\boldsymbol{a}\sim\tilde{\boldsymbol{\pi}}}\left[A_{\boldsymbol{\pi}}^{-i_p}(s,a^{i_p},\boldsymbol{a}^{-i_p})\right]$ in surrogate objective $M_{\boldsymbol{\pi}}^{-i_p}(\tilde{\pi}^{i_p},\tilde{\boldsymbol{\pi}}^{-i_p})$, we have

$$
\begin{aligned}
&\mathbb{E}_{\boldsymbol{a}\sim\tilde{\boldsymbol{\pi}}}\left[A_{\boldsymbol{\pi}}^{-i_p}(s,a^{i_p},\boldsymbol{a}^{-i_p})\right] \\
&\quad \text{which, by Equations (1), (2), equals} \\
&= \mathbb{E}_{\boldsymbol{a}\sim\tilde{\boldsymbol{\pi}}}\left[Q_{\boldsymbol{\pi}}^{i_p,-i_p}(s,\boldsymbol{a}^{i_p,-i_p}) - Q_{\boldsymbol{\pi}}^{i_p}(s,\boldsymbol{a}^{i_p})\right] \\
&= \mathbb{E}_{\boldsymbol{a}\sim\tilde{\boldsymbol{\pi}}}\left[Q_{\boldsymbol{\pi}}(s,\boldsymbol{a}) - V_{\boldsymbol{\pi}}(s)\right] - \mathbb{E}_{\boldsymbol{a}\sim\tilde{\boldsymbol{\pi}}}\left[Q_{\boldsymbol{\pi}}^{i_p}(s,\boldsymbol{a}^{i_p}) - V_{\boldsymbol{\pi}}(s)\right] \\
&= \mathbb{E}_{\boldsymbol{a}\sim\tilde{\boldsymbol{\pi}}}\left[A_{\boldsymbol{\pi}}(s,\boldsymbol{a})\right] - \mathbb{E}_{\boldsymbol{a}\sim\tilde{\boldsymbol{\pi}}}\left[\mathbb{E}_{\boldsymbol{a}^{-i_p}\sim\boldsymbol{\pi}^{-i_p}}[Q_{\boldsymbol{\pi}}(s,\boldsymbol{a})] - V_{\boldsymbol{\pi}}(s)\right] \\
&= \mathbb{E}_{\boldsymbol{a}\sim\tilde{\boldsymbol{\pi}}}\left[A_{\boldsymbol{\pi}}(s,\boldsymbol{a})\right] - \mathbb{E}_{a^{i_p}\sim\tilde{\pi}^{i_p},\boldsymbol{a}^{-i_p}\sim\boldsymbol{\pi}^{-i_p}}\left[Q_{\boldsymbol{\pi}}(s,\boldsymbol{a}) - V_{\boldsymbol{\pi}}(s)\right] \\
&= \mathbb{E}_{\boldsymbol{a}\sim\tilde{\boldsymbol{\pi}}}\left[A_{\boldsymbol{\pi}}(s,\boldsymbol{a})\right] - \mathbb{E}_{a^{i_p}\sim\tilde{\pi}^{i_p},\boldsymbol{a}^{-i_p}\sim\boldsymbol{\pi}^{-i_p}}\left[A_{\boldsymbol{\pi}}(s,\boldsymbol{a})\right] \\
&= \mathbb{E}_{\boldsymbol{a}\sim\boldsymbol{\pi}}\left[\frac{\tilde{\boldsymbol{\pi}}(\boldsymbol{a}|s)}{\boldsymbol{\pi}(\boldsymbol{a}|s)}A_{\boldsymbol{\pi}}(s,\boldsymbol{a})\right] - \mathbb{E}_{\boldsymbol{a}\sim\boldsymbol{\pi}}\left[\frac{\tilde{\pi}^{i_p}(a^{i_p}|s)}{\pi^{i_p}(a^{i_p}|s)}A_{\boldsymbol{\pi}}(s,\boldsymbol{a})\right] \\
&= \mathbb{E}_{\boldsymbol{a}\sim\boldsymbol{\pi}}\left[\left(\frac{\tilde{\pi}^{j_p}(a^{j_p}|s)}{\pi^{j_p}(a^{j_p}|s)}\cdot\frac{\tilde{\boldsymbol{\pi}}^{-j_p}(\boldsymbol{a}^{-j_p}|s)}{\boldsymbol{\pi}^{-i_p}(\boldsymbol{a}^{-j_p}|s)} - \frac{\tilde{\pi}^{i_p}(a^{i_p}|s)}{\pi^{i_p}(a^{i_p}|s)}\right)A_{\boldsymbol{\pi}}(s,\boldsymbol{a})\right].
\end{aligned}
\tag{17}
$$

End of the proof. We achieve the change of the sampling of actions $\{a^1,...,a^n\}$ from $\{\tilde{\pi}^1,...,\tilde{\pi}^n\}$ to $\{\pi^1,...,\pi^n\}$ and avoid the dynamic action input problem. Moreover, all agents only need to maintain a joint advantage estimator for $A_{\boldsymbol{\pi}}(s,\boldsymbol{a})$.

B.5 Proof of Equation 8

$$
\begin{aligned}
&M_{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\text{old}}}}^{-i_p}(\pi_{\theta_{\text{mid}}^{i_p}}^{i_p},\boldsymbol{\pi}_{\boldsymbol{\theta}^{-i_p}}^{-i_p}) \\
&= \mathbb{E}_{s,\boldsymbol{a}}\left[\left(\frac{\pi_{\theta^{j_p}}^{j_p}(a^{j_p}|s)}{\pi_{\theta_{\text{old}}^{j_p}}^{j_p}(a^{j_p}|s)}\cdot\frac{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\text{mid}}^{-j_p}}^{-j_p}(\boldsymbol{a}^{-j_p}|s)}{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\text{old}}^{-j_p}}^{-j_p}(\boldsymbol{a}^{-j_p}|s)} - \frac{\pi_{\theta^{i_p}}^{i_p}(a^{i_p}|s)}{\pi_{\theta_{\text{old}}^{i_p}}^{i_p}(a^{i_p}|s)}\right)A_{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\text{old}}}}(s,\boldsymbol{a})\right] - \sum\nolimits_{i_k\in-i_p}\mathcal{C}D_{\text{KL}}^{\max}(\pi_{\theta_{\text{old}}^{i_k}}^{i_k},\pi_{\theta^{i_k}}^{i_k}), \\
&= \mathbb{E}_{s,\boldsymbol{a}}\left[\frac{\pi_{\theta^{j_p}}^{j_p}(a^{j_p}|s)}{\pi_{\theta_{\text{old}}^{j_p}}^{j_p}(a^{j_p}|s)}\cdot\frac{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\text{mid}}^{-j_p}}^{-j_p}(\boldsymbol{a}^{-j_p}|s)}{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\text{old}}^{-j_p}}^{-j_p}(\boldsymbol{a}^{-j_p}|s)}A_{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\text{old}}}}(s,\boldsymbol{a})\right] - \mathcal{C}D_{\text{KL}}^{\max}(\pi_{\theta_{\text{old}}^{j_p}}^{j_p},\pi_{\theta^{j_p}}^{j_p}) \\
&\qquad - \mathbb{E}_{s,\boldsymbol{a}}\left[\frac{\pi_{\theta^{i_p}}^{i_p}(a^{i_p}|s)}{\pi_{\theta_{\text{old}}^{i_p}}^{i_p}(a^{i_p}|s)}A_{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\text{old}}}}(s,\boldsymbol{a})\right] - \sum\nolimits_{i_k\in-i_p-j_p}\mathcal{C}D_{\text{KL}}^{\max}(\pi_{\theta_{\text{old}}^{i_k}}^{i_k},\pi_{\theta^{i_k}}^{i_k}),
\end{aligned}
$$

where $s\sim\rho_{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\text{old}}}}$, $\boldsymbol{a}\sim\boldsymbol{\pi}_{\boldsymbol{\theta}_{\text{old}}}$. To execute $\underset{\theta^{j_p}}{\text{maximize}}\, M_{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\text{old}}}}^{-i_p}(\pi_{\theta_{\text{mid}}^{i_p}}^{i_p},\boldsymbol{\pi}_{\boldsymbol{\theta}^{-i_p}}^{-i_p})$ in Pytorch or Tensorflow framework, we take the derivative of θ^{j_p} and then adopt the gradient boosting method to make

$M^{-i_p}_{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\text{old}}}}(\pi^{i_p}_{\theta^{i_p}_{\text{mid}}}, \boldsymbol{\pi}^{-i_p}_{\boldsymbol{\theta}^{-i_p}})$ improved. Therefore, we have

$$
\begin{aligned}
&\frac{\partial M^{-i_p}_{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\text{old}}}}(\pi^{i_p}_{\theta^{i_p}_{\text{mid}}}, \boldsymbol{\pi}^{-i_p}_{\boldsymbol{\theta}^{-i_p}})}{\partial \theta^{j_p}} \\
&= \frac{\partial \mathbb{E}_{s,\boldsymbol{a}}\left[\frac{\pi^{j_p}_{\theta^{j_p}}(a^{j_p}|s)}{\pi^{j_p}_{\theta^{j_p}_{\text{old}}}(a^{j_p}|s)} \cdot \frac{\boldsymbol{\pi}^{-j_p}_{\boldsymbol{\theta}^{-j_p}_{\text{mid}}}(\boldsymbol{a}^{-j_p}|s)}{\boldsymbol{\pi}^{-j_p}_{\boldsymbol{\theta}^{-j_p}_{\text{old}}}(\boldsymbol{a}^{-j_p}|s)} A_{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\text{old}}}}(s,\boldsymbol{a})\right] - \mathcal{C}D^{\max}_{\text{KL}}(\pi^{j_p}_{\theta^{j_p}_{\text{old}}}, \pi^{j_p}_{\theta^{j_p}})}{\partial \theta^{j_p}} \\
&\quad - \frac{\partial \mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\text{old}}}}, \boldsymbol{a}\sim\boldsymbol{\pi}_{\boldsymbol{\theta}_{\text{old}}}}\left[\frac{\pi^{i_p}_{\theta^{i_p}}(a^{i_p}|s)}{\pi^{i_p}_{\theta^{i_p}_{\text{old}}}(a^{i_p}|s)} A_{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\text{old}}}}(s,\boldsymbol{a})\right] + \sum_{i_k\in -i_p - j_p} \mathcal{C}D^{\max}_{\text{KL}}(\pi^{i_k}_{\theta^{i_k}_{\text{old}}}, \pi^{i_k}_{\theta^{i_k}})}{\partial \theta^{j_p}}, \\
&= \frac{\partial \mathbb{E}_{s,\boldsymbol{a}}\left[\frac{\pi^{j_p}_{\theta^{j_p}}(a^{j_p}|s)}{\pi^{j_p}_{\theta^{j_p}_{\text{old}}}(a^{j_p}|s)} \cdot \frac{\boldsymbol{\pi}^{-j_p}_{\boldsymbol{\theta}^{-j_p}_{\text{mid}}}(\boldsymbol{a}^{-j_p}|s)}{\boldsymbol{\pi}^{-j_p}_{\boldsymbol{\theta}^{-j_p}_{\text{old}}}(\boldsymbol{a}^{-j_p}|s)} A_{\boldsymbol{\pi}_{\boldsymbol{\theta}_{\text{old}}}}(s,\boldsymbol{a})\right] - \mathcal{C}D^{\max}_{\text{KL}}(\pi^{j_p}_{\theta^{j_p}_{\text{old}}}, \pi^{j_p}_{\theta^{j_p}})}{\partial \theta^{j_p}} - 0
\end{aligned} \tag{18}
$$

Note that we aim to optimize the policy network of agent j_p, i.e., θ^{j_p} with the policies of other agents as constants (see non-overlapping selection). Therefore, as indicated in Equation (18), $-\pi^{i_p}_{\theta^{i_p}}(a^{i_p}|s)/\pi^{i_p}_{\theta^{i_p}_{\text{old}}}(a^{i_p}|s)$ and $\sum_{i_k\in -i_p - j_p} \mathcal{C}D^{\max}_{\text{KL}}(\pi^{i_k}_{\theta^{i_k}_{\text{old}}}, \pi^{i_k}_{\theta^{i_k}})$ can be omitted since they, as constant terms, have no contribution to the gradients of θ^{j_p}.

B.6 Proof of Theorem 3

Theorem 3 describes a process of monotonically improved lower bound in FP3O: $\mathcal{L}_{\boldsymbol{\pi}}(\tilde{\pi}^{i_p}, \tilde{\boldsymbol{\pi}}^{-i_p}) \geq \mathcal{L}_{\boldsymbol{\pi}}(\hat{\pi}^{i_p}, \hat{\boldsymbol{\pi}}^{-i_p}) \geq \mathcal{L}_{\boldsymbol{\pi}}(\pi^{i_p}, \boldsymbol{\pi}^{-i_p})$. To prove this, we first expand the lower bound into the form of importance sampling:

$$
\begin{aligned}
&\mathcal{L}_{\boldsymbol{\pi}}(\tilde{\pi}^{i_p}, \tilde{\boldsymbol{\pi}}^{-i_p}) \\
&= \mathcal{J}(\boldsymbol{\pi}) + M^{i_p}_{\boldsymbol{\pi}}(\tilde{\pi}^{\varnothing}, \tilde{\pi}^{i_p}) + M^{-i_p}_{\boldsymbol{\pi}}(\tilde{\pi}^{i_p}, \tilde{\boldsymbol{\pi}}^{-i_p}) \\
&\text{and by Definition 1, this is} \\
&= \mathcal{J}(\boldsymbol{\pi}) + \mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}}, \boldsymbol{a}\sim\tilde{\boldsymbol{\pi}}}[A^{i_p}_{\boldsymbol{\pi}}(s, a^{i_p})] - \mathcal{C}D^{\max}_{\text{KL}}(\pi^{i_p}, \tilde{\pi}^{i_p}) \\
&\qquad + \mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}}, \boldsymbol{a}\sim\tilde{\boldsymbol{\pi}}}[A^{-i_p}_{\boldsymbol{\pi}}(s, a^{i_p}, \boldsymbol{a}^{-i_p})] - \sum_{i_k\in -i_p} \mathcal{C}D^{\max}_{\text{KL}}(\pi^{i_k}, \tilde{\pi}^{i_k}), \\
&\text{which, by Equations (6), (7), equals} \\
&= \mathcal{J}(\boldsymbol{\pi}) + \mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}}, \boldsymbol{a}\sim\boldsymbol{\pi}}\left[\frac{\tilde{\pi}^{i_p}(a^{i_p}|s)}{\pi^{i_p}(a^{i_p}|s)} A_{\boldsymbol{\pi}}(s,\boldsymbol{a})\right] \qquad \text{// Equation (6)} \\
&\qquad + \mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}}, \boldsymbol{a}\sim\boldsymbol{\pi}}\left[\left(\frac{\tilde{\pi}^{j_p}(a^{j_p}|s)}{\pi^{j_p}(a^{j_p}|s)} \cdot \frac{\tilde{\boldsymbol{\pi}}^{-j_p}(\boldsymbol{a}^{-j_p}|s)}{\boldsymbol{\pi}^{-i_p}(\boldsymbol{a}^{-j_p}|s)} - \frac{\tilde{\pi}^{i_p}(a^{i_p}|s)}{\pi^{i_p}(a^{i_p}|s)}\right) A_{\boldsymbol{\pi}}(s,\boldsymbol{a})\right] \qquad \text{// Equation (7)} \\
&\qquad\qquad - \sum_{k=1}^{n} \mathcal{C}D^{\max}_{\text{KL}}(\pi^{i_k}, \tilde{\pi}^{i_k}) \\
&= \mathcal{J}(\boldsymbol{\pi}) + \mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}}, \boldsymbol{a}\sim\boldsymbol{\pi}}\left[\prod_{k=1}^{n}\left(\frac{\tilde{\pi}^{i_k}(a^{i_k}|s)}{\pi^{i_k}(a^{i_k}|s)}\right) A_{\boldsymbol{\pi}}(s,\boldsymbol{a})\right] - \sum_{k=1}^{n} \mathcal{C}D^{\max}_{\text{KL}}(\pi^{i_k}, \tilde{\pi}^{i_k}).
\end{aligned} \tag{19}
$$

Note that we will analyze the monotonic improvement property of FP3O at the theoretical level. Thus, we do not consider the parameterized policy π_θ but theoretical policy π.

We first discuss the optimization direction for Equation (9), (10) in FP3O without parameterized policy. At the beginning of one iteration, the initial optimization objective is

$$\begin{aligned}
&\mathbb{E}_{s,\boldsymbol{a}}\left[\min\left(\frac{\pi^{i_p}(a^{i_p}|s)}{\pi^{i_p}(a^{i_p}|s)}A_{\boldsymbol{\pi}}(s,\boldsymbol{a}),\text{clip}\left(\frac{\pi^{i_p}(a^{i_p}|s)}{\pi^{i_p}(a^{i_p}|s)},1\pm\epsilon\right)A_{\boldsymbol{\pi}}(s,\boldsymbol{a})\right)\right] \\
&=\mathbb{E}_{s,\boldsymbol{a}}\left[A_{\boldsymbol{\pi}}(s,\boldsymbol{a})\right]=\sum_i P_{\boldsymbol{\pi}}(s_i,\boldsymbol{a}_i)A_{\boldsymbol{\pi}}(s_i,\boldsymbol{a}_i) \\
&=\sum_i P_{\boldsymbol{\pi}}(s_i)\cdot\pi^1(a_i^1|s_i)\cdot\ldots\cdot\pi^n(a_i^n|s_i)\cdot A_{\boldsymbol{\pi}}(s_i,\boldsymbol{a}_i)
\end{aligned}\tag{20}$$

where $(s_i,\boldsymbol{a}_i)$ is the state-action pair and $P_{\boldsymbol{\pi}}(s_i,\boldsymbol{a}_i)$ is the probability of pair $(s_i,\boldsymbol{a}_i)$. How can we make Equation (20) improved in these state-action pairs $(s_i,\boldsymbol{a}_i)$? We know that every state-action pair $(s_i,\boldsymbol{a}_i)$ corresponds to one $A_{\boldsymbol{\pi}}(s_i,\boldsymbol{a}_i)$. Therefore, if we increase the probability (i.e., $\boldsymbol{\pi}=\prod_i\pi^i$) of those state-action pairs with positive $A_{\boldsymbol{\pi}}$ and decrease the probability of those state-action pairs with negative $A_{\boldsymbol{\pi}}$, then the initial optimization objective, i.e., Equation (20) can be effectively improved. However, the increase/decrease of joint policy probability $\boldsymbol{\pi}=\prod_i\pi^i$ does not mean the increase/decrease of single policy probability π^i. Now, we analyze the relationship between single policy and joint policy in FP3O. At the end of independence step, the optimization objective is

$$\begin{aligned}
&\mathbb{E}_{s,\boldsymbol{a}}\left[\min\left(\frac{\hat{\pi}^{i_p}(a^{i_p}|s)}{\pi^{i_p}(a^{i_p}|s)}A_{\boldsymbol{\pi}}(s,\boldsymbol{a}),\text{clip}\left(\frac{\hat{\pi}^{i_p}(a^{i_p}|s)}{\pi^{i_p}(a^{i_p}|s)},1\pm\epsilon\right)A_{\boldsymbol{\pi}}(s,\boldsymbol{a})\right)\right] \\
&=\sum_i P_{\boldsymbol{\pi}}(s_i)\pi^1(a_i^1|s_i)\ldots\pi^n(a_i^n|s_i)\min\left(\frac{\hat{\pi}^{i_p}(a_i^{i_p}|s_i)}{\pi^{i_p}(a_i^{i_p}|s_i)}A_{\boldsymbol{\pi}}(s_i,\boldsymbol{a}_i),\text{clip}\left(\frac{\hat{\pi}^{i_p}(a_i^{i_p}|s_i)}{\pi^{i_p}(a_i^{i_p}|s_i)},1\pm\epsilon\right)A_{\boldsymbol{\pi}}(s_i,\boldsymbol{a}_i)\right) \\
&=\begin{cases}\sum_i P_{\boldsymbol{\pi}}(s_i)\prod_{k\in-i_p}\pi^k(a_i^k|s_i)\cdot\hat{\pi}^{i_p}(a_i^{i_p}|s_i)A_{\boldsymbol{\pi}}(s_i,\boldsymbol{a}_i), \\ \qquad\qquad\text{or} \\ \sum_i P_{\boldsymbol{\pi}}(s_i)\prod_{k\in-i_p}\pi^k(a_i^k|s_i)\cdot\text{clip}\left(\hat{\pi}^{i_p}(a_i^{i_p}|s_i),\pi^{i_p}(a_i^{i_p}|s_i)\pm\epsilon\pi^{i_p}(a_i^{i_p}|s_i)\right)A_{\boldsymbol{\pi}}(s_i,\boldsymbol{a}_i)\end{cases}
\end{aligned}\tag{21}$$

Comparing Equation (21) with the initial objective, i.e., Equation (20), we can see that probability $P_{\boldsymbol{\pi}}(s_i)\prod_{k\in-i_p}\pi^k(a_i^k|s_i)$ is exactly the same. This is because that we have taken $\prod_{k\in-i_p}\pi^k(a_i^k|s_i)$ as a constant at independence step of pipeline p. In this case, the increase/decrease of joint policy probability $\boldsymbol{\pi}$ is completely equivalent to the increase/decrease of single policy probability π^{i_p}. Thus, at independence step in Equation (21), we can increase probability (i.e., $\pi^{i_p}/\boldsymbol{\pi}$) of those state-action pairs with positive $A_{\boldsymbol{\pi}}$ and decrease the probability (i.e., $\pi^{i_p}/\boldsymbol{\pi}$) of those state-action pairs with negative $A_{\boldsymbol{\pi}}$ to make Equation (20) effectively improved. We have discussed an optimization direction at the independence step, and dependence step is similar to independence step. The initial objective:

$$\sum_i P_{\boldsymbol{\pi}}(s_i)\cdot\hat{\pi}^1(a_i^1|s_i)\cdot\ldots\cdot\hat{\pi}^n(a_i^n|s_i)\cdot A_{\boldsymbol{\pi}}(s_i,\boldsymbol{a}_i)\tag{22}$$

At the end of dependence step (note that we take policies $\boldsymbol{\pi}^{-j_p}$ as constants according to non-overlapping selection), the optimization objective is

$$\begin{aligned}
&\mathbb{E}_{s,\boldsymbol{a}}\left[\min\left(\frac{\tilde{\pi}^{j_p}(a^{j_p}|s)}{\pi^{j_p}(a^{j_p}|s)}\frac{\hat{\boldsymbol{\pi}}^{-j_p}(\boldsymbol{a}^{-j_p}|s)}{\boldsymbol{\pi}^{-i_p}(\boldsymbol{a}^{-j_p}|s)}A_{\boldsymbol{\pi}}(s,\boldsymbol{a}),\text{clip}\left(\frac{\tilde{\pi}^{j_p}(a^{j_p}|s)}{\pi^{j_p}(a^{j_p}|s)},1\pm\epsilon\right)\frac{\hat{\boldsymbol{\pi}}^{-j_p}(\boldsymbol{a}^{-j_p}|s)}{\boldsymbol{\pi}^{-i_p}(\boldsymbol{a}^{-j_p}|s)}A_{\boldsymbol{\pi}}(s,\boldsymbol{a})\right)\right] \\
&=\begin{cases}\sum_i P_{\boldsymbol{\pi}}(s_i)\prod_{k\in-j_p}\hat{\pi}^k(a_i^k|s_i)\cdot\tilde{\pi}^{j_p}(a_i^{j_p}|s_i)A_{\boldsymbol{\pi}}(s_i,\boldsymbol{a}_i), \\ \qquad\qquad\text{or} \\ \sum_i P_{\boldsymbol{\pi}}(s_i)\prod_{k\in-j_p}\hat{\pi}^k(a_i^k|s_i)\cdot\text{clip}\left(\tilde{\pi}^{j_p}(a_i^{j_p}|s_i),\pi^{j_p}(a_i^{j_p}|s_i)\pm\epsilon\pi^{j_p}(a_i^{j_p}|s_i)\right)A_{\boldsymbol{\pi}}(s_i,\boldsymbol{a}_i)\end{cases}
\end{aligned}\tag{23}$$

In this case, due to the equal item $\sum_i P_{\boldsymbol{\pi}}(s_i)\prod_{k\in-j_p}\hat{\pi}^k(a_i^k|s_i)$, the increase/decrease of joint policy probability $\boldsymbol{\pi}$ is completely equivalent to the increase/decrease of single policy probability π^{j_p}. Thus, based on the above discussion, we give the following assumption as a reasonable optimization direction for theoretical policy π.

Assumption 1. *For pipeline p in FP3O, in case of $A_{\boldsymbol{\pi}}(s,\boldsymbol{a}) > 0$, $\pi^{i_p}(a^{i_p}|s)$ and $\pi^{j_p}(a^{j_p}|s)$ tends to increase at independence step and dependence step respectively to make optimization objective improved; In case of $A_{\boldsymbol{\pi}}(s,\boldsymbol{a}) < 0$, $\pi^{i_p}(a^{i_p}|s)$ and $\pi^{j_p}(a^{j_p}|s)$ tends to decrease at independence step and dependence step respectively to make optimization objective improved;*

Lemma 3. *For $n \in \mathbb{N}$ positive numbers $\{l_k\}_{k=1}^n = \{l_1, l_2, ..., l_n\}$, if $l_k \geq 1$, we have $\prod_{k=1}^n l_k \geq \frac{1}{n}\sum_{k=1}^n l_k$. If $l_k \leq 1$, we have $\prod_{k=1}^n l_k \leq \frac{1}{n}\sum_{k=1}^n l_k$.*

Proof. If $l_k \geq 1$, we take the maximum value $l_{k_{\max}} = \max\{l_1, l_2, ..., l_n\}$, then we have

$$\prod_{k=1}^{n} l_k = l_{k_{\max}} \cdot \left(\prod_{k=1}^{k_{\max}-1} l_k \cdot \prod_{k=k_{\max}+1}^{n} l_k \right) \geq l_{k_{\max}},$$

$$\frac{1}{n}\sum_{k=1}^{n} l_k = \frac{1}{n}\left(l_{k_{\max}} + \sum_{k=1}^{k_{\max}-1} \sum_{k=k_{\max}+1}^{n} l_k \right) \leq \frac{1}{n}\left(\sum_{k=1}^{n} l_{k_{\max}} \right) = l_{k_{\max}},$$

$$\text{so we have } \prod_{k=1}^{n} l_k \geq \frac{1}{n}\sum_{k=1}^{n} l_k.$$

If $l_k \leq 1$, we take the minimum value $l_{k_{\min}} = \min\{l_1, l_2, ..., l_n\}$, then we have

$$\prod_{k=1}^{n} l_k = l_{k_{\min}} \cdot \left(\prod_{k=1}^{k_{\min}-1} l_k \cdot \prod_{k=k_{\min}+1}^{n} l_k \right) \leq l_{k_{\min}},$$

$$\frac{1}{n}\sum_{k=1}^{n} l_k = \frac{1}{n}\left(l_{k_{\min}} + \sum_{k=1}^{k_{\min}-1} \sum_{k=k_{\min}+1}^{n} l_k \right) \geq \frac{1}{n}\left(\sum_{k=1}^{n} l_{k_{\min}} \right) = l_{k_{\min}},$$

$$\text{so we have } \prod_{k=1}^{n} l_k \leq \frac{1}{n}\sum_{k=1}^{n} l_k.$$

□

Theorem 3. *We let $\pi^1, ..., \pi^n$ denote the old policies. $\hat{\pi}^1, ..., \hat{\pi}^n$ and $\tilde{\pi}^1, ..., \tilde{\pi}^n$ are the middle policies and new policies, which are obtained from all pipelines (i.e., $p = 1, ..., n$) by Equation (11) and Equation (12) respectively. $\mathcal{L}_{\boldsymbol{\pi}}(\tilde{\pi}^{i_p}, \tilde{\boldsymbol{\pi}}^{-i_p}) = \mathcal{J}(\boldsymbol{\pi}) + M_{\boldsymbol{\pi}}^{i_p}(\tilde{\pi}^{\varnothing}, \tilde{\pi}^{i_p}) + M_{\boldsymbol{\pi}}^{-i_p}(\tilde{\pi}^{i_p}, \tilde{\boldsymbol{\pi}}^{-i_p})$ is the theoretical lower bound of pipeline p defined in Theorem 2. Given the assumption described in Appendix B.6, the following inequalities hold for all pipeline $p = 1, ..., n$:*

$$\left\{ \mathcal{L}_{\boldsymbol{\pi}}(\tilde{\pi}^{i_p}, \tilde{\boldsymbol{\pi}}^{-i_p}) \geq \mathcal{L}_{\boldsymbol{\pi}}(\hat{\pi}^{i_p}, \hat{\boldsymbol{\pi}}^{-i_p}) \geq \mathcal{L}_{\boldsymbol{\pi}}(\pi^{i_p}, \boldsymbol{\pi}^{-i_p}) \right\}_{p=1}^{n}.$$

Proof. We have converted FP3O to KL penalty version by Equations (11) and (12).

At independence step of FP3O, we do

$$\begin{aligned}
\hat{\pi}^{i_p} &= \arg\max_{\hat{\pi}^{i_p}} \left[\mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}}, a^{i_p}\sim\hat{\pi}^{i_p}} \left[A_{\boldsymbol{\pi}}^{i_p}(s, a^{\varnothing}, a^{i_p}) \right] - \beta D_{\mathrm{KL}}^{\max}(\pi^{i_p}, \hat{\pi}^{i_p}) \right] \\
&= \arg\max_{\hat{\pi}^{i_p}} \left[\mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}}, \boldsymbol{a}\sim\boldsymbol{\pi}} \left[\frac{\hat{\pi}^{i_p}(a^{i_p}|s)}{\pi^{i_p}(a^{i_p}|s)} A_{\boldsymbol{\pi}}(s,\boldsymbol{a}) \right] - \beta D_{\mathrm{KL}}^{\max}(\pi^{i_p}, \hat{\pi}^{i_p}) \right].
\end{aligned}$$

Thus, we can get

$$\begin{aligned}
&\mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}}, \boldsymbol{a}\sim\boldsymbol{\pi}} \left[\frac{\hat{\pi}^{i_p}(a^{i_p}|s)}{\pi^{i_p}(a^{i_p}|s)} A_{\boldsymbol{\pi}}(s,\boldsymbol{a}) \right] - \beta D_{\mathrm{KL}}^{\max}(\pi^{i_p}, \hat{\pi}^{i_p}) \\
&\qquad \geq \mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}}, \boldsymbol{a}\sim\boldsymbol{\pi}} \left[\frac{\pi^{i_p}(a^{i_p}|s)}{\pi^{i_p}(a^{i_p}|s)} A_{\boldsymbol{\pi}}(s,\boldsymbol{a}) \right] - \beta D_{\mathrm{KL}}^{\max}(\pi^{i_p}, \pi^{i_p}). \\
\Leftrightarrow \quad &\mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}}, \boldsymbol{a}\sim\boldsymbol{\pi}} \left[\frac{\hat{\pi}^{i_p}(a^{i_p}|s)}{\pi^{i_p}(a^{i_p}|s)} A_{\boldsymbol{\pi}}(s,\boldsymbol{a}) \right] - \beta D_{\mathrm{KL}}^{\max}(\pi^{i_p}, \hat{\pi}^{i_p}) \geq 0 \\
\Leftrightarrow \quad &D_{\mathrm{KL}}^{\max}(\pi^{i_p}, \hat{\pi}^{i_p}) \leq \frac{1}{\beta} \mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}}, \boldsymbol{a}\sim\boldsymbol{\pi}} \left[\frac{\hat{\pi}^{i_p}(a^{i_p}|s)}{\pi^{i_p}(a^{i_p}|s)} A_{\boldsymbol{\pi}}(s,\boldsymbol{a}) \right].
\end{aligned}$$

When jointly updating all pipelines $p = 1, .., n$, we have

$$\left\{ D_{\mathrm{KL}}^{\max}(\pi^{i_p}, \hat{\pi}^{i_p}) \leq \frac{1}{\beta} \mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}}, \boldsymbol{a}\sim\boldsymbol{\pi}} \left[\frac{\hat{\pi}^{i_p}(a^{i_p}|s)}{\pi^{i_p}(a^{i_p}|s)} A_{\boldsymbol{\pi}}(s, \boldsymbol{a}) \right] \right\}_{p=1}^{n}. \tag{24}$$

This means that, by executing independence step of FP3O, we can get the above inequality relations. Then we prove that executing independence step of FP3O can improve the theoretical lower bound $\mathcal{L}_\pi$. For any pipeline p, we compare the lower bound at the end of independence step of FP3O with the lower bound at the beginning:

$$\begin{aligned}
&\mathcal{L}_{\boldsymbol{\pi}}(\hat{\pi}^{i_p}, \hat{\boldsymbol{\pi}}^{-i_p}) - \mathcal{L}_{\boldsymbol{\pi}}(\pi^{i_p}, \boldsymbol{\pi}^{-i_p}) \\
&= \mathcal{L}_{\boldsymbol{\pi}}(\hat{\pi}^{i_p}, \hat{\boldsymbol{\pi}}^{-i_p}) - \mathcal{J}(\boldsymbol{\pi}) \qquad \text{// Equation (13)} \\
&\text{and, by Equation (19), this is} \\
&= \mathcal{J}(\boldsymbol{\pi}) + \mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}}, \boldsymbol{a}\sim\boldsymbol{\pi}} \left[\prod_{k=1}^{n} \left(\frac{\hat{\pi}^{i_k}(a^{i_k}|s)}{\pi^{i_k}(a^{i_k}|s)} \right) A_{\boldsymbol{\pi}}(s, \boldsymbol{a}) \right] - \sum_{k=1}^{n} \mathcal{C} D_{\mathrm{KL}}^{\max}(\pi^{i_k}, \hat{\pi}^{i_k}) - \mathcal{J}(\boldsymbol{\pi}) \qquad (25) \\
&\geq \mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}}, \boldsymbol{a}\sim\boldsymbol{\pi}} \left[\prod_{k=1}^{n} \left(\frac{\hat{\pi}^{i_k}(a^{i_k}|s)}{\pi^{i_k}(a^{i_k}|s)} \right) A_{\boldsymbol{\pi}}(s, \boldsymbol{a}) \right] \\
&\qquad\qquad - \sum_{k=1}^{n} \frac{\mathcal{C}}{\beta} \mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}}, \boldsymbol{a}\sim\boldsymbol{\pi}} \left[\frac{\hat{\pi}^{i_k}(a^{i_k}|s)}{\pi^{i_k}(a^{i_k}|s)} A_{\boldsymbol{\pi}}(s, \boldsymbol{a}) \right] \qquad \text{// Equation (24)} \\
&= \mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}}, \boldsymbol{a}\sim\boldsymbol{\pi}} \left[\prod_{k=1}^{n} \left(\frac{\hat{\pi}^{i_k}(a^{i_k}|s)}{\pi^{i_k}(a^{i_k}|s)} \right) A_{\boldsymbol{\pi}}(s, \boldsymbol{a}) \right] \\
&\qquad\qquad - \mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}}, \boldsymbol{a}\sim\boldsymbol{\pi}} \left[\frac{\mathcal{C}}{\beta} \sum_{k=1}^{n} \frac{\hat{\pi}^{i_k}(a^{i_k}|s)}{\pi^{i_k}(a^{i_k}|s)} A_{\boldsymbol{\pi}}(s, \boldsymbol{a}) \right] \\
&= \mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}}, \boldsymbol{a}\sim\boldsymbol{\pi}} \left[\left(\prod_{k=1}^{n} \frac{\hat{\pi}^{i_k}(a^{i_k}|s)}{\pi^{i_k}(a^{i_k}|s)} - \frac{\mathcal{C}}{\beta} \sum_{k=1}^{n} \frac{\hat{\pi}^{i_k}(a^{i_k}|s)}{\pi^{i_k}(a^{i_k}|s)} \right) A_{\boldsymbol{\pi}}(s, \boldsymbol{a}) \right] \\
&\text{if we take } \frac{\mathcal{C}}{\beta} = \frac{1}{n}, \text{ then this is} \\
&\geq 0 \qquad \text{// Assumption 1 and Lemma 3} \qquad (26)
\end{aligned}$$

Note that we have considered that all policies are changed when jointly updating all pipelines (see the superscript). So far, we have proved that independence step of FP3O can improve the theoretical lower bound: $\left\{ \mathcal{L}_{\boldsymbol{\pi}}(\hat{\pi}^{i_p}, \hat{\boldsymbol{\pi}}^{-i_p}) \geq \mathcal{L}_{\boldsymbol{\pi}}(\pi^{i_p}, \boldsymbol{\pi}^{-i_p}) \right\}_{p=1}^{n}$. Then, we prove that dependence step of FP3O can further improve the theoretical lower bound.

At dependence step, we do

$$\begin{aligned}
\tilde{\pi}^{j_p} &= \arg\max_{\tilde{\pi}^{j_p}} \left[\mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}}, a^{j_p}\sim\tilde{\pi}^{j_p}, \boldsymbol{a}^{-j_p}\sim\hat{\boldsymbol{\pi}}^{-j_p}} \left[A_{\boldsymbol{\pi}}^{-i_p}(s, a^{i_p}, \boldsymbol{a}^{-i_p}) \right] - \beta D_{\mathrm{KL}}^{\max}(\pi^{j_p}, \tilde{\pi}^{j_p}) \right] \\
&= \arg\max_{\tilde{\pi}^{j_p}} \left[\mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}}, \boldsymbol{a}\sim\boldsymbol{\pi}} \left[\left(\frac{\tilde{\pi}^{j_p}(a^{j_p}|s)}{\pi^{j_p}(a^{j_p}|s)} \frac{\hat{\boldsymbol{\pi}}^{-j_p}(\boldsymbol{a}^{-j_p}|s)}{\boldsymbol{\pi}^{-i_p}(\boldsymbol{a}^{-j_p}|s)} - \frac{\hat{\pi}^{i_p}(a^{i_p}|s)}{\pi^{i_p}(a^{i_p}|s)} \right) A_{\boldsymbol{\pi}}(s, \boldsymbol{a}) \right] - \beta D_{\mathrm{KL}}^{\max}(\pi^{j_p}, \tilde{\pi}^{j_p}) \right].
\end{aligned}$$

Thus, we have

$$\mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}},\boldsymbol{a}\sim\boldsymbol{\pi}}\left[\left(\frac{\tilde{\pi}^{j_p}(a^{j_p}|s)}{\pi^{j_p}(a^{j_p}|s)}\frac{\hat{\boldsymbol{\pi}}^{-j_p}(\boldsymbol{a}^{-j_p}|s)}{\boldsymbol{\pi}^{-i_p}(\boldsymbol{a}^{-j_p}|s)}-\frac{\hat{\pi}^{i_p}(a^{i_p}|s)}{\pi^{i_p}(a^{i_p}|s)}\right)A_{\boldsymbol{\pi}}(s,\boldsymbol{a})\right]-\beta D_{\mathrm{KL}}^{\max}(\pi^{j_p},\tilde{\pi}^{j_p})$$
$$\geq \mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}},\boldsymbol{a}\sim\boldsymbol{\pi}}\left[\left(\frac{\hat{\pi}^{j_p}(a^{j_p}|s)}{\pi^{j_p}(a^{j_p}|s)}\frac{\hat{\boldsymbol{\pi}}^{-j_p}(\boldsymbol{a}^{-j_p}|s)}{\boldsymbol{\pi}^{-i_p}(\boldsymbol{a}^{-j_p}|s)}-\frac{\hat{\pi}^{i_p}(a^{i_p}|s)}{\pi^{i_p}(a^{i_p}|s)}\right)A_{\boldsymbol{\pi}}(s,\boldsymbol{a})\right]-\beta D_{\mathrm{KL}}^{\max}(\pi^{j_p},\hat{\pi}^{j_p}) \tag{27}$$

$$\Leftrightarrow\quad D_{\mathrm{KL}}^{\max}(\pi^{j_p},\tilde{\pi}^{j_p})-D_{\mathrm{KL}}^{\max}(\pi^{j_p},\hat{\pi}^{j_p})$$
$$\leq\frac{1}{\beta}\left(\mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}},\boldsymbol{a}\sim\boldsymbol{\pi}}\left[\left(\frac{\tilde{\pi}^{j_p}(a^{j_p}|s)}{\pi^{j_p}(a^{j_p}|s)}\frac{\hat{\boldsymbol{\pi}}^{-j_p}(\boldsymbol{a}^{-j_p}|s)}{\boldsymbol{\pi}^{-i_p}(\boldsymbol{a}^{-j_p}|s)}\right)A_{\boldsymbol{\pi}}(s,\boldsymbol{a})\right]\right.$$
$$\left.-\mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}},\boldsymbol{a}\sim\boldsymbol{\pi}}\left[\left(\frac{\hat{\pi}^{j_p}(a^{j_p}|s)}{\pi^{j_p}(a^{j_p}|s)}\frac{\hat{\boldsymbol{\pi}}^{-j_p}(\boldsymbol{a}^{-j_p}|s)}{\boldsymbol{\pi}^{-i_p}(\boldsymbol{a}^{-j_p}|s)}\right)A_{\boldsymbol{\pi}}(s,\boldsymbol{a})\right]\right)$$

$$\Leftrightarrow\quad D_{\mathrm{KL}}^{\max}(\pi^{j_p},\tilde{\pi}^{j_p})-D_{\mathrm{KL}}^{\max}(\pi^{j_p},\hat{\pi}^{j_p})$$
$$\leq\frac{1}{\beta}\left(\mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}},\boldsymbol{a}\sim\boldsymbol{\pi}}\left[\left(\frac{\tilde{\pi}^{j_p}(a^{j_p}|s)}{\pi^{j_p}(a^{j_p}|s)}\prod_{i_k\in-j_p}\frac{\hat{\pi}^{i_k}(a^{i_k}|s)}{\pi^{i_k}(a^{i_k}|s)}-\prod_{k=1}^{n}\frac{\hat{\pi}^{i_k}(a^{i_k}|s)}{\pi^{i_k}(a^{i_k}|s)}\right)A_{\boldsymbol{\pi}}(s,\boldsymbol{a})\right]\right)$$

When jointly updating all pipelines $p=1,..,n$, we have

$$\left\{D_{\mathrm{KL}}^{\max}(\pi^{j_p},\tilde{\pi}^{j_p})-D_{\mathrm{KL}}^{\max}(\pi^{j_p},\hat{\pi}^{j_p})\right.$$
$$\left.\leq\frac{1}{\beta}\left(\mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}},\boldsymbol{a}\sim\boldsymbol{\pi}}\left[\left(\frac{\tilde{\pi}^{j_p}(a^{j_p}|s)}{\pi^{j_p}(a^{j_p}|s)}\prod_{i_k\in-j_p}\frac{\hat{\pi}^{i_k}(a^{i_k}|s)}{\pi^{i_k}(a^{i_k}|s)}-\prod_{k=1}^{n}\frac{\hat{\pi}^{i_k}(a^{i_k}|s)}{\pi^{i_k}(a^{i_k}|s)}\right)A_{\boldsymbol{\pi}}(s,\boldsymbol{a})\right]\right)\right\}_{p=1}^{n} \tag{28}$$

After executing dependence step of FP3O, we get the above inequality. Then, for any pipeline p, we compare the lower bound at the end of dependence step with the lower bound at the end of

independence step:

$$
\begin{aligned}
&\mathcal{L}_{\boldsymbol{\pi}}(\tilde{\pi}^{i_p}, \tilde{\boldsymbol{\pi}}^{-i_p}) - \mathcal{L}_{\boldsymbol{\pi}}(\hat{\pi}^{i_p}, \hat{\boldsymbol{\pi}}^{-i_p}) \\
&= \mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}}, \boldsymbol{a}\sim\boldsymbol{\pi}} \left[\left(\prod_{k=1}^{n} \frac{\tilde{\pi}^{i_k}(a^{i_k}|s)}{\pi^{i_k}(a^{i_k}|s)} - \prod_{k=1}^{n} \frac{\hat{\pi}^{i_k}(a^{i_k}|s)}{\pi^{i_k}(a^{i_k}|s)} \right) A_{\boldsymbol{\pi}}(s, \boldsymbol{a}) \right] \\
&\qquad - \sum_{k=1}^{n} \left(\mathcal{C} D_{\mathrm{KL}}^{\max}(\pi^{i_k}, \tilde{\pi}^{i_k}) - \mathcal{C} D_{\mathrm{KL}}^{\max}(\pi^{i_k}, \hat{\pi}^{i_k}) \right) \quad \text{// Equation (19)} \qquad (29) \\
&\text{which, by Equation 28, is} \\
&\geq \mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}}, \boldsymbol{a}\sim\boldsymbol{\pi}} \left[\left(\prod_{k=1}^{n} \frac{\tilde{\pi}^{i_k}(a^{i_k}|s)}{\pi^{i_k}(a^{i_k}|s)} - \prod_{k=1}^{n} \frac{\hat{\pi}^{i_k}(a^{i_k}|s)}{\pi^{i_k}(a^{i_k}|s)} \right) A_{\boldsymbol{\pi}}(s, \boldsymbol{a}) \right] \\
&- \sum_{p=1}^{n} \frac{\mathcal{C}}{\beta} \left(\mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}}, \boldsymbol{a}\sim\boldsymbol{\pi}} \left[\left(\right. \right. \right. \\
&\qquad \left. \left. \left. \frac{\tilde{\pi}^{j_p}(a^{j_p}|s)}{\pi^{j_p}(a^{j_p}|s)} \prod_{i_k \in -j_p} \frac{\tilde{\pi}^{i_k}(a^{i_k}|s)}{\pi^{i_k}(a^{i_k}|s)} - \prod_{k=1}^{n} \frac{\hat{\pi}^{i_k}(a^{i_k}|s)}{\pi^{i_k}(a^{i_k}|s)} \right) A_{\boldsymbol{\pi}}(s, \boldsymbol{a}) \right] \right) \qquad (30) \\
&\text{if we take } \frac{\mathcal{C}}{\beta} = \frac{1}{n}, \text{ this is} \\
&= \mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}}, \boldsymbol{a}\sim\boldsymbol{\pi}} \left[\left(\prod_{k=1}^{n} \frac{\tilde{\pi}^{i_k}(a^{i_k}|s)}{\pi^{i_k}(a^{i_k}|s)} - \frac{1}{n} \sum_{p=1}^{n} \left(\frac{\tilde{\pi}^{j_p}(a^{j_p}|s)}{\pi^{j_p}(a^{j_p}|s)} \prod_{i_k \in -j_p} \frac{\hat{\pi}^{i_k}(a^{i_k}|s)}{\pi^{i_k}(a^{i_k}|s)} \right) \right) A_{\boldsymbol{\pi}}(s, \boldsymbol{a}) \right] \\
&= \mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}}, \boldsymbol{a}\sim\boldsymbol{\pi}} \left[\frac{1}{n} \sum_{p=1}^{n} \left\{ \right. \right. \\
&\qquad \left. \left. \frac{\tilde{\pi}^{j_p}(a^{j_p}|s)}{\pi^{j_p}(a^{j_p}|s)} \left(\prod_{i_k \in -j_p} \frac{\tilde{\pi}^{i_k}(a^{i_k}|s)}{\pi^{i_k}(a^{i_k}|s)} - \prod_{i_k \in -j_p} \frac{\hat{\pi}^{i_k}(a^{i_k}|s)}{\pi^{i_k}(a^{i_k}|s)} \right) \right\} A_{\boldsymbol{\pi}}(s, \boldsymbol{a}) \right] \\
&\geq 0 \qquad \text{// Assumption 1}
\end{aligned}
$$

Note that we have considered that all policies are changed when jointly updating all pipelines, and also considered the non-overlapping selection and the policy approximation defined by pipeline interaction (see the superscript). Therefore, we have proved that dependence step of FP3O can further improve the theoretical lower bound: $\left\{\mathcal{L}_{\boldsymbol{\pi}}(\tilde{\pi}^{i_p}, \tilde{\boldsymbol{\pi}}^{-i_p}) \geq \mathcal{L}_{\boldsymbol{\pi}}(\hat{\pi}^{i_p}, \hat{\boldsymbol{\pi}}^{-i_p})\right\}_{p=1}^{n}$. Thus, we have proved that the lower bounds of all pipelines can be monotonically improved at both independence step and dependence step in FP3O: $\left\{\mathcal{L}_{\boldsymbol{\pi}}(\tilde{\pi}^{i_p}, \tilde{\boldsymbol{\pi}}^{-i_p}) \geq \mathcal{L}_{\boldsymbol{\pi}}(\hat{\pi}^{i_p}, \hat{\boldsymbol{\pi}}^{-i_p}) \geq \mathcal{L}_{\boldsymbol{\pi}}(\pi^{i_p}, \boldsymbol{\pi}^{-i_p})\right\}_{p=1}^{n}$. It can be derived from Equation (19) that the lower bounds of all pipelines are equivalent:

$$\mathcal{L}_{\boldsymbol{\pi}}(\tilde{\pi}^{i_1}, \tilde{\boldsymbol{\pi}}^{-i_1}) =, ..., = \mathcal{L}_{\boldsymbol{\pi}}(\tilde{\pi}^{i_n}, \tilde{\boldsymbol{\pi}}^{-i_n}). \qquad (31)$$

This means that the improved lower bounds of all pipelines are also equivalent. With non-decreasing lower bounds, the true objectives of all pipelines can be also improved.

□

We worry that $\frac{\mathcal{C}}{\beta} = \frac{1}{n}$ (i.e., penalty coefficient $\beta = n\mathcal{C}$) will impose too strict restrictions on KL divergence with the increase of the number of agents. Fortunately $\beta = n\mathcal{C}$ is just a soft condition. If a more lenient penalty coefficient is adopted (e.g., $\beta = \mathcal{C}$), the new policy will tend to be far away from the old policy when we optimize Equations (11), (12). To be specific, if $\pi(a|s)$ tends to decrease, $\frac{\tilde{\pi}(a|s)}{\pi(a|s)}$ will tend to be smaller after the update. If $\pi(a|s)$ tends to increase, $\frac{\tilde{\pi}(a|s)}{\pi(a|s)}$ will tend to be larger after the update. Therefore, we only need to take a more lenient coefficient to make Equation (26) hold with the following Lemma 4.

Lemma 4. *For $n \in \mathbb{N}$ positive numbers $\{l_k\}_{k=1}^{n} = \{l_1, l_2, ..., l_n\}$, there is a coefficient ξ that makes the equation $\prod_{k=1}^{n} l_k = \frac{1}{\xi} \sum_{k=1}^{n} l_k$ holds. If $l_k \geq 1$ and another larger n numbers $\{l'_k\}_{k=1}^{n}$ satisfies*

$l'_k > l_k$, there will be a smaller coefficient $\xi' < \xi$ that makes the equation $\prod_{k=1}^{n} l'_k = \frac{1}{\xi'} \sum_{k=1}^{n} l'_k$ holds. Similarly, if $l_k \leq 1$ and another smaller n numbers $\{l'_k\}_{k=1}^{n}$ satisfies $l'_k < l_k$, there will be a smaller coefficient $\xi' < \xi$ that makes the equation $\prod_{k=1}^{n} l'_k = \frac{1}{\xi'} \sum_{k=1}^{n} l'_k$ holds.

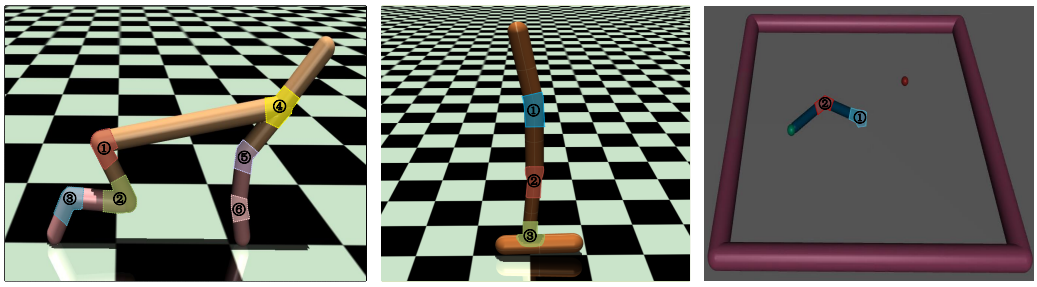
Similarly, a more lenient coefficient can be also applied to Equation (30). Specifically, we can simplify Equation (30) to $Z(\tilde{\boldsymbol{\pi}}^{-j_p}) - \sum_{p=1}^{n} \frac{C}{\beta} Z(\hat{\boldsymbol{\pi}}^{-j_p})$, where $Z(\tilde{\boldsymbol{\pi}}^{-j_p}) = \mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}},\boldsymbol{a}\sim\boldsymbol{\pi}}\left[\left(\prod_{k=1}^{n} \frac{\tilde{\pi}^{i_k}(a^{i_k}|s)}{\pi^{i_k}(a^{i_k}|s)} - \prod_{k=1}^{n} \frac{\hat{\pi}^{i_k}(a^{i_k}|s)}{\pi^{i_k}(a^{i_k}|s)}\right) A_{\boldsymbol{\pi}}(s,\boldsymbol{a})\right]$. We always have $Z(\tilde{\boldsymbol{\pi}}^{-j_p}) > Z(\hat{\boldsymbol{\pi}}^{-j_p})$, and their difference will increase if a small β is adopted. This is because, under this setting, $\tilde{\pi}$ tend to get much smaller or much larger. Thus, a more lenient coefficient penalty can guarantee $\mathcal{L}_{\boldsymbol{\pi}}(\tilde{\pi}^{i_p}, \tilde{\boldsymbol{\pi}}^{-i_p}) - \mathcal{L}_{\boldsymbol{\pi}}(\hat{\pi}^{i_p}, \hat{\boldsymbol{\pi}}^{-i_p}) \geq 0$ to hold.

In our experiments, we also demonstrate in Table 2 that FP3O can achieve 100% performances on bane vs. bane involving 24 agents without more strict coefficient on PPO-clipping ($\epsilon = 0.2$).

C Environments

C.1 MAMuJoCo

MAMuJoCo benchmark (de Witt et al., 2020b) is a continuous and partially observable task. MAMuJoCo groups different joints of a robot in the MuJoCo simulator (Todorov et al., 2012) and models them as different agents. For instance, as shown in Figure 3 (a), MAMuJoCo regards six joints of HalfCheetah as six different agents, and then lets each agent control its joints based on its local observation. Because of the diversification of body part control, the agents are regarded as heterogeneous.



(a) (b) (c)

Figure 3: Examples of MAMuJoCo. Different agents are depicted in different colors. (a) 6-Agent HalfCheetah [6×1]. (b) 3-Agent Hopper [3×1]. (c) 2-Agent Reacher [2×1]

.

C.2 SMAC

SMAC benchmark (Samvelyan et al., 2019) is a discrete and partially observable task. There are a variety of maps involving three difficulties: Easy, Hard, and Super-Hard. The goal is to train a team of ally units to defeat an opponent team of enemy units. Table 3 lists the types of units in different scenarios. According to whether the types of ally units are the same or not, the scenarios be classified into homogeneous-agent or heterogeneous-agent tasks. For example, agents are regarded to be heterogeneous in scenarios bane vs bane and 3s5z since the ally contains two different types of units. While agents are regarded to be homogeneous in the remaining scenarios (i.e., 2c vs 64zg, 5m vs 6m, corridor, 6h vs 8z) because of the single type of units.

Table 3: Unit types in SMAC benchmark.

Scenarios	Ally Units	Enemy Units	Type
bane vs bane	20 Zerglings & 4 Banelings	20 Zerglings & 4 Banelings	HE
2c vs 64zg	2 Colossi	64 Zerglings	HO
3s5z	3 Stalkers & 5 Zealots	3 Stalkers & 5 Zealots	HE
5m vs 6m	5 Marines	6 Marines	HO
corridor	6 Zealots	24 Zerglings	HO
6h vs 8z	6 Hydralisks	8 Zealots	HO

D TRAINING DETAILS

Computing infrastructure. We implement our FP3O algorithm in the PyTorch framework version 1.9.0 with GPU acceleration. All of our models are trained on TITAN X GPU, running 64-bit Linux 4.4.0.

Repetitive experiments. In our experiments on MAMuJoCo, the score is averaged over 32 test episodes after each training iteration. The final evaluation score is further averaged over 5 different seeds. In SMAC, We follow the median win rate evaluation metric adpoted in Wang et al. (2021b); Yu et al. (2021). The win rate is computed over 32 test episodes after each training iteration and the final evaluation result is averaged over 5 different seeds.

D.1 HYPERPARAMETER SETTINGS IN MAMUJOCO DOMAIN

The hyperparameter settings in MAMuJoCo are shown in Table 4 and Table 5.

Table 4: Common hyperparameters in MAMuJoCo.

hyperparameter	value	hyperparameter	value
actor lr	1e-5	gain	0.01
critic lr	3e-4	episode length	1000
gamma	0.99	rollout threads	4
gae lamda	0.95	batch size	4000
optimizer	Adam	num mini-batch	32
optimizer epsilon	1e-5	gradient clip norm	10
weight decay	0	hidden layer dim	64
activation	ReLU	num hidden layer	2
network initialization	Orthogonal	num layer after	1
training threads	8	value loss	huber loss
ppo epochs	5	huber delta	10.0
stacked frames	1	entropy coef	0.001
std x coef	1	std y coef	0.5
obsk	5		

Table 5: Adopted hyperparameters for different tasks in MAMuJoCo. FuPS, PaPS, and NoPS denote full parameter sharing, partial parameter sharing, and non-parameter sharing respectively.

task	ppo clip	actor network	steps		
			FuPS	PaPS	NoPS
2-Agent Reacher [2×1]	0.2	mlp	10e6	10e6	10e6
2-Agent Ant [2×4]	0.2	mlp	10e6	10e6	10e6
2-Agent Walker [2×3]	0.05	mlp	10e6	10e6	10e6
3-Agent Hopper [3×1]	0.05	mlp	10e6	10e6	10e6
6-Agent HalfCheetah [6×1]	0.2	mlp	10e6	10e6	10e6
Manyagent Swimmer [8×2]	0.2	mlp	10e6	10e6	10e6

D.2 Hyperparameter Settings in SMAC Domain

The hyperparameter settings in SMAC are shown in Table 6 and Table 7.

Table 6: Common hyperparameters in SMAC.

hyperparameters	value	hyperparameters	value
actor lr	5e-4	gain	0.01
critic lr	5e-4	episode length	400
gamma	0.99	rollout threads	8
gae lamda	0.95	batch size	3200
optimizer	Adam	num mini-batch	1
optimizer epsilon	1e-5	gradient clip norm	10
weight decay	0	hidden layer dim	64
activation	ReLU	num hidden layer	2
network initialization	Orthogonal	num layer after	1
training threads	32	value loss	huber loss
ppo epochs	5	huber delta	10.0
stacked frames	1	entropy coef	0.01

Table 7: Adopted hyperparameters for different maps in SMAC. FuPS, PaPS, and NoPS denote full parameter sharing, partial parameter sharing, and non-parameter sharing respectively.

map	ppo clip	actor network	steps		
			FuPS	PaPS	NoPS
bane vs. bane	0.2	rnn	2e6	2e6	2e6
2c vs. 64zg	0.2	rnn	5e6	5e6	5e6
3s5z	0.2	rnn	5e6	5e6	5e6
5m vs. 6m	0.05	rnn	8e6	10e6	10e6
corridor	0.2	mlp	8e6	10e6	10e6
6h vs. 8z	0.2	mlp	10e6	15e6	20e6
10m vs. 11m	0.2	rnn	8e6	-	-
3s5z vs. 3s6z	0.05	rnn	12e6	-	-

E ADDITIONAL EXPERIMENTAL RESULTS

E.1 KL DIVERGENCE DISTRIBUTION IN FP3O

To verify that FP3O will not suffer from the limitation in HAPPO that has been discussed in Appendix A, we execute the same experiment to train FP3O with full parameter sharing network type on 3s5z map in SMAC. we demonstrate in Figure 4 the distribution KL divergence between the updated policy and the old policy of each agent after finishing their optimization at one policy interation. Different from the KL divergence in HAPPO that gradually becomes larger during the update process, FP3O can effectively control KL divergences of all agents within the trust region, which is essential for the monotonic improvement guarantee to be valid in a variety of tasks and settings. Moreover, we want to emphasize that every agent has to start a one-separation trust region optimization pipeline in FP3O so that all agents can be updated in parallel (see Equation (4)). Thus, the update policies of all agents start from their old policies at one policy iteration (i.e, the ratio $\pi^{i_p}_{\theta^{i_p}}(a^{i_p}|s)\pi^{i_p}_{\theta^{i_p}_{\text{old}}}(a^{i_p}|s)$ of all agents is equal to 1). Compared with Figure 1, each agent in FP3O can make full use of the collected data to update in the expected optimization direction, which is a data-efficient update scheme.

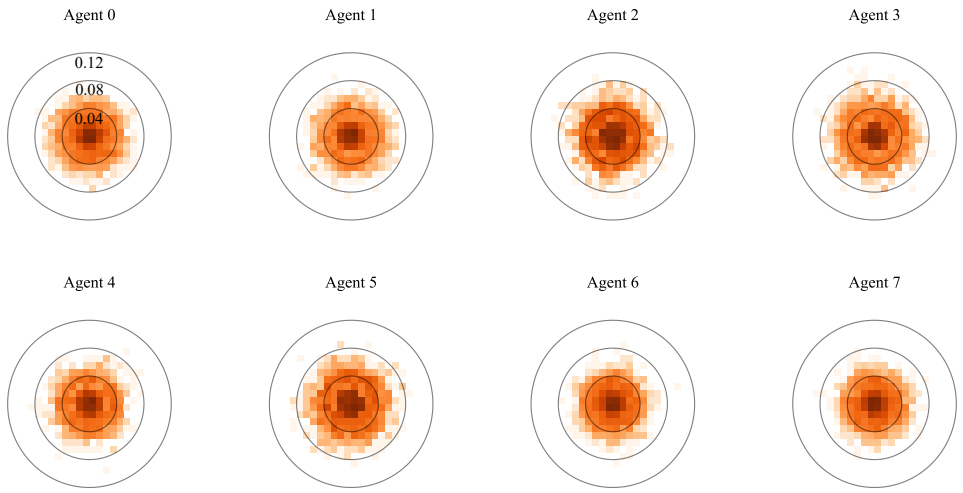


Figure 4: The distribution of the KL divergence between the updated policy and the old policy at one policy interation of FP3O. The darker the color is, the more data is distributed. The center represents that KL divergence is equal to 0, and the outward direction represents that KL divergence gets greater. The places where KL divergence equals 0.04, 0.08, and 0.12 are indicated by the black circles.

E.2 ADDITIONAL RESULTS ON SMAC

We perform additional experiments with CoPPO baseline (Wu et al., 2021) on more SMAC scenarios. As we can see in Figure 5, due to the constantly changing of each agent's optimization objective after every mini-batch update, the training process of CoPPO is non-stationary, even in the easy scenarios (i.e., bane vs. bane and 2c vs. 64zg). Our FP3O gets superior performances on all tasks.

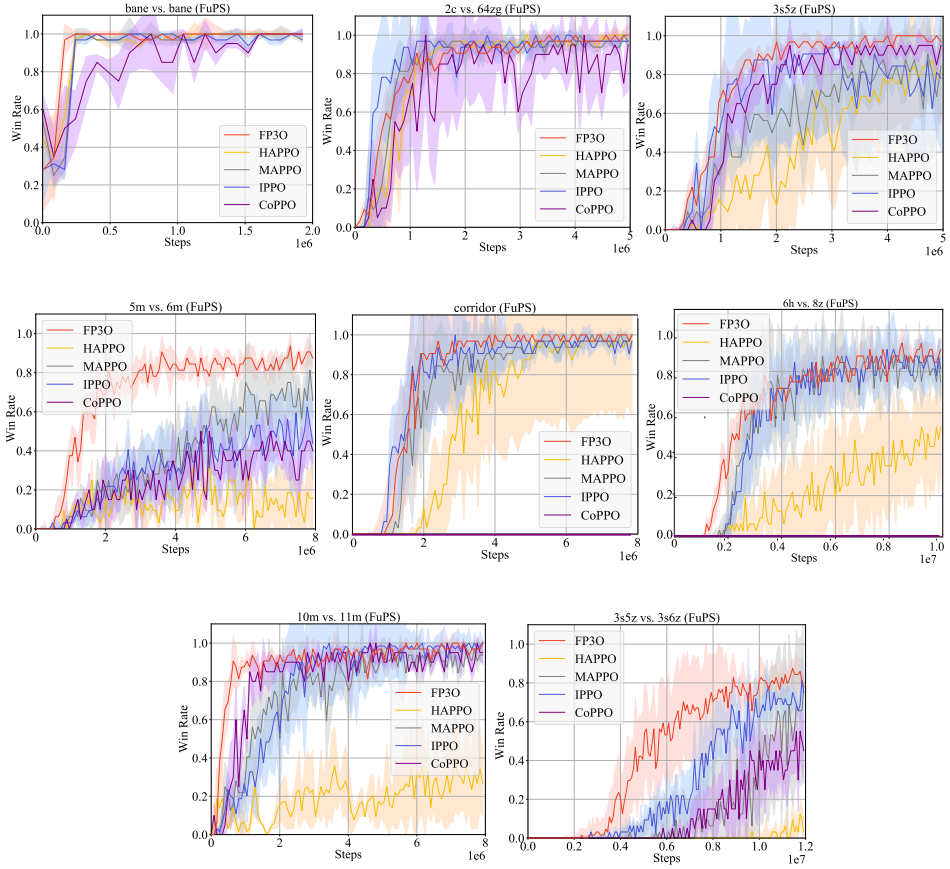


Figure 5: Additional results on SMAC.

E.3 The Training Results of Tables 1 And 2

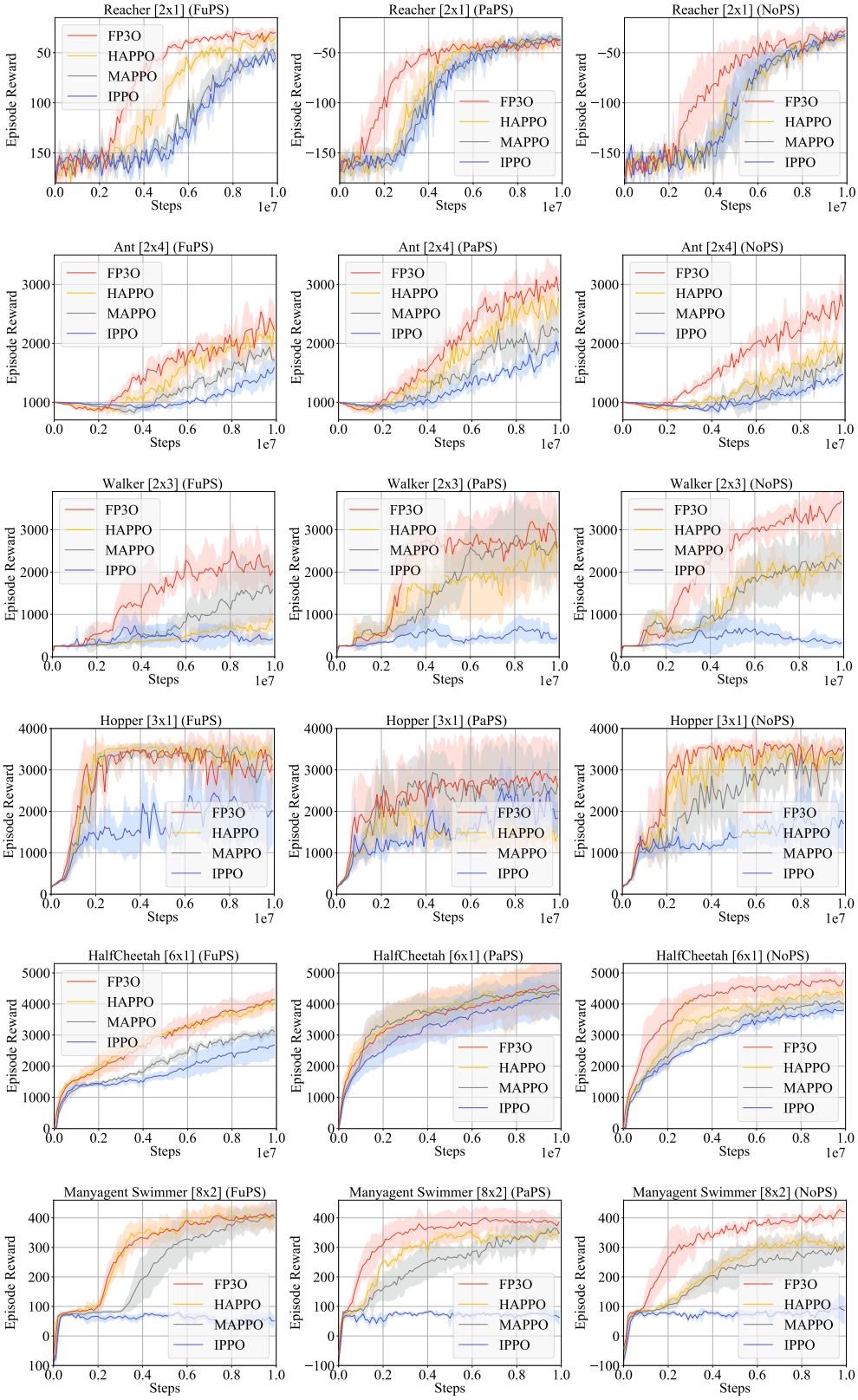


Figure 6: Average evaluation reward in the MAMuJoCo domain.

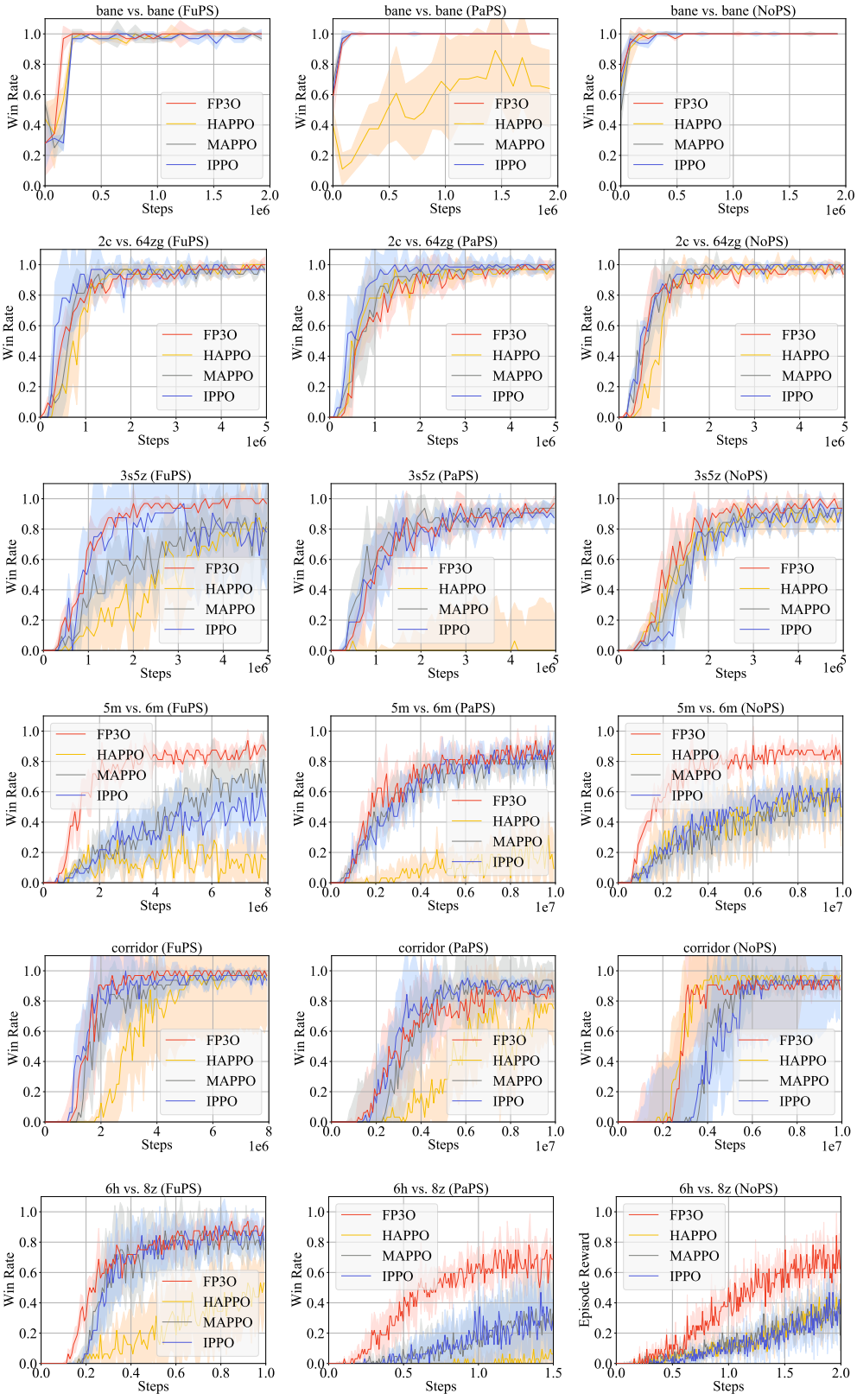


Figure 7: Median evaluation win rate in the SMAC domain.