Journal of Machine Learning Research 25 (2024) 1-67 Submitted 4/23; Revised 10/23; Published 1/24

Heterogeneous-Agent Reinforcement Learning

Yifan Zhong[1,2,*] ZHONGYIFAN@STU.PKU.EDU.CN
Jakub Grudzien Kuba[3,*] JAKUB.GRUDZIEN@NEW.OX.AC.UK
Xidong Feng[4,*] XIDONG.FENG.20@UCL.AC.UK
Siyi Hu[5] SIYI.HU@STUDENT.UTS.EDU.AU
Jiaming Ji[1] JIAMG.JI@STU.PKU.EDU.CN
Yaodong Yang[1,†] YAODONG.YANG@PKU.EDU.CN

[1] *Institute for Artificial Intelligence, Peking University*
[2] *Beijing Institute for General Artificial Intelligence*
[3] *University of Oxford*
[4] *University College London*
[5] *ReLER, AAII, University of Technology Sydney*
[*] *Equal contribution* † *Corresponding author*

Editor: George Konidaris

Abstract

The necessity for cooperation among intelligent machines has popularised cooperative *multi-agent reinforcement learning* (MARL) in AI research. However, many research endeavours heavily rely on parameter sharing among agents, which confines them to only *homogeneous*-agent setting and leads to training instability and lack of convergence guarantees. To achieve effective cooperation in the general *heterogeneous*-agent setting, we propose *Heterogeneous-Agent Reinforcement Learning* (HARL) algorithms that resolve the aforementioned issues. Central to our findings are the *multi-agent advantage decomposition lemma* and the *sequential update scheme*. Based on these, we develop the provably correct *Heterogeneous-Agent Trust Region Learning* (HATRL), and derive HATRPO and HAPPO by tractable approximations. Furthermore, we discover a novel framework named *Heterogeneous-Agent Mirror Learning* (HAML), which strengthens theoretical guarantees for HATRPO and HAPPO and provides a general template for cooperative MARL algorithmic designs. We prove that all algorithms derived from HAML inherently enjoy monotonic improvement of joint return and convergence to Nash Equilibrium. As its natural outcome, HAML validates more novel algorithms in addition to HATRPO and HAPPO, including HAA2C, HADDPG, and HATD3, which generally outperform their existing MA-counterparts. We comprehensively test HARL algorithms on six challenging benchmarks and demonstrate their superior effectiveness and stability for coordinating heterogeneous agents compared to strong baselines such as MAPPO and QMIX.[1]

Keywords: cooperative multi-agent reinforcement learning, heterogeneous-agent trust region learning, heterogeneous-agent mirror learning, heterogeneous-agent reinforcement learning algorithms, sequential update scheme

1. Our code is available at `https://github.com/PKU-MARL/HARL`.

1. Introduction

Cooperative Multi-Agent Reinforcement Learning (MARL) is a natural model of learning in multi-agent systems, such as robot swarms (Hüttenrauch et al., 2017, 2019), autonomous cars (Cao et al., 2012), and traffic signal control (Calvo and Dusparic, 2018). To solve cooperative MARL problems, one naive approach is to directly apply single-agent reinforcement learning algorithm to each agent and consider other agents as a part of the environment, a paradigm commonly referred to as *Independent Learning* (Tan, 1993; de Witt et al., 2020). Though effective in certain tasks, independent learning fails in the face of more complex scenarios (Hu et al., 2022b; Foerster et al., 2018), which is intuitively clear: once a learning agent updates its policy, so do its teammates, which causes changes in the effective environment of each agent which single-agent algorithms are not prepared for (Claus and Boutilier, 1998). To address this, a learning paradigm named *Centralised Training with Decentralised Execution* (CTDE) (Lowe et al., 2017; Foerster et al., 2018; Zhou et al., 2023) was developed. The CTDE framework learns a joint value function which, during training, has access to the global state and teammates' actions. With the help of the centralised value function that accounts for the non-stationarity caused by others, each agent adapts its policy parameters accordingly. Thus, it effectively leverages global information while still preserving decentralised agents for execution. As such, the CTDE paradigm allows a straightforward extension of single-agent policy gradient theorems (Sutton et al., 2000; Silver et al., 2014) to multi-agent scenarios (Lowe et al., 2017; Kuba et al., 2021; Mguni et al., 2021). Consequently, numerous multi-agent policy gradient algorithms have been developed (Foerster et al., 2018; Peng et al., 2017; Zhang et al., 2020; Wen et al., 2018, 2020; Yang et al., 2018; Ackermann et al., 2019).

Though existing methods have achieved reasonable performance on common benchmarks, several limitations remain. Firstly, some algorithms (Yu et al., 2022; de Witt et al., 2020) rely on parameter sharing and require agents to be *homogeneous* (*i.e.*, share the same observation space and action space, and play similar roles in a cooperation task), which largely limits their applicability to *heterogeneous*-agent settings (*i.e.*, no constraint on the observation spaces, action spaces, and the roles of agents) and potentially harms the performance (Christianos et al., 2021). While there has been work extending parameter sharing for heterogeneous agents (Terry et al., 2020), their methods rely on padding, which is neither elegant nor general. Secondly, existing algorithms update the agents simultaneously. As we show in Section 2.3.1 later, the agents are unaware of partners' update directions under this update scheme, which could lead to potentially conflicting updates, resulting in training instability and failure of convergence. Lastly, some algorithms, such as IPPO and MAPPO, are developed based on intuition and empirical results. The lack of theory compromises their trustworthiness for important usage.

To resolve these challenges, in this work we propose *Heterogeneous-Agent Reinforcement Learning* (HARL) algorithm series, that is meant for the general *heterogeneous*-agent settings, achieves effective coordination through a novel *sequential update scheme*, and is grounded theoretically.

In particular, we capitalize on the *multi-agent advantage decomposition lemma* (Kuba et al., 2021) and derive the theoretically underpinned multi-agent extension of trust region learning, which is proved to enjoy monotonic improvement property and convergence to

the Nash Equilibrium (NE) guarantee. Based on this, we propose Heterogeneous-Agent Trust Region Policy Optimisation (HATRPO) and Heterogeneous-Agent Proximal Policy Optimisation (HAPPO) as tractable approximations to theoretical procedures.

Furthermore, inspired by Mirror Learning (Kuba et al., 2022b) that provides a theoretical explanation for the effectiveness of TRPO and PPO , we discover a novel framework named *Heterogeneous-Agent Mirror Learning* (HAML), which strengthens theoretical guarantees for HATRPO and HAPPO and provides a general template for cooperative MARL algorithmic designs. We prove that all algorithms derived from HAML inherently satisfy the desired property of the monotonic improvement of joint return and the convergence to Nash equilibrium. Thus, HAML dramatically expands the theoretically sound algorithm space and, potentially, provides cooperative MARL solutions to more practical settings. We explore the HAML class and derive more theoretically underpinned and practical heterogeneous-agent algorithms, including HAA2C, HADDPG, and HATD3.

To facilitate the usage of HARL algorithms, we open-source our PyTorch-based integrated implementation. Based on this, we test HARL algorithms comprehensively on Multi-Agent Particle Environment (MPE) (Lowe et al., 2017; Mordatch and Abbeel, 2018), Multi-Agent MuJoCo (MAMuJoCo) (Peng et al., 2021), StarCraft Multi-Agent Challenge (SMAC) (Samvelyan et al., 2019), SMACv2 (Ellis et al., 2022), Google Research Football Environment (GRF) (Kurach et al., 2020), and Bi-DexterousHands (Chen et al., 2022). The empirical results confirm the algorithms' effectiveness in practice. On all benchmarks with heterogeneous agents including MPE, MAMuJoCo, GRF, and Bi-Dexteroushands, HARL algorithms generally outperform their existing MA-counterparts, and their performance gaps become larger as the heterogeneity of agents increases, showing that HARL algorithms are more robust and better suited for the general heterogeneous-agent settings. While all HARL algorithms show competitive performance, they culminate in HAPPO and HATD3 in particular, which establish the new state-of-the-art results. As an off-policy algorithm, HATD3 also improves sample efficiency, leading to more efficient learning and faster convergence. On tasks where agents are mostly homogeneous such as SMAC and SMACv2, HAPPO and HATRPO attain comparable or superior win rates at convergence while not relying on the parameter-sharing trick, demonstrating their general applicability. Through ablation analysis, we empirically show the novelties introduced by HARL theory and algorithms are crucial for learning the optimal cooperation strategy, thus signifying their importance. Finally, we systematically analyse the computational overhead of sequential update and conclude that it does not need to be a concern.

2. Preliminaries

In this section, we first introduce problem formulation and notations for cooperative MARL, and then review existing work and analyse their limitations.

2.1 Cooperative MARL Problem Formulation and Notations

We consider a fully cooperative multi-agent task that can be described as a Markov game (MG) (Littman, 1994), also known as a stochastic game (Shapley, 1953).

Definition 1 *A cooperative Markov game is defined by a tuple $\langle \mathcal{N}, \mathcal{S}, \boldsymbol{\mathcal{A}}, r, P, \gamma, d \rangle$. Here, $\mathcal{N} = \{1, \dots, n\}$ is a set of n agents, $\mathcal{S}$ is the state space, $\boldsymbol{\mathcal{A}} = \times_{i=1}^{n} \mathcal{A}^i$ is the products of all agents' action spaces, known as the joint action space. Further, $r : \mathcal{S} \times \boldsymbol{\mathcal{A}} \to \mathbb{R}$ is the joint reward function, $P : \mathcal{S} \times \boldsymbol{\mathcal{A}} \times \mathcal{S} \to [0, 1]$ is the transition probability kernel, $\gamma \in [0, 1)$ is the discount factor, and $d \in \mathcal{P}(\mathcal{S})$ (where $\mathcal{P}(X)$ denotes the set of probability distributions over a set X) is the positive initial state distribution.*

Although our results hold for general compact state and action spaces, in this paper we assume that they are finite, for simplicity. In this work, we will also use the notation $\mathbb{P}(X)$ to denote the power set of a set X. At time step $t \in \mathbb{N}$, the agents are at state s_t; they take independent actions $\mathrm{a}_t^i, \forall i \in \mathcal{N}$ drawn from their policies $\pi^i(\cdot^i|\mathrm{s}_t) \in \mathcal{P}(\mathcal{A}^i)$, and equivalently, they take a joint action $\mathbf{a}_t = (\mathrm{a}_t^1, \dots, \mathrm{a}_t^n)$ drawn from their joint policy $\boldsymbol{\pi}(\cdot|\mathrm{s}_t) = \prod_{i=1}^{n} \pi^i(\cdot^i|\mathrm{s}_t) \in \mathcal{P}(\boldsymbol{\mathcal{A}})$. We write $\Pi^i \triangleq \{\times_{s \in \mathcal{S}} \pi^i(\cdot^i|s) \ | \forall s \in \mathcal{S}, \pi^i(\cdot^i|s) \in \mathcal{P}(\mathcal{A}^i)\}$ to denote the policy space of agent i, and $\boldsymbol{\Pi} \triangleq (\Pi^1, \dots, \Pi^n)$ to denote the joint policy space. It is important to note that when $\pi^i(\cdot^i|s)$ is a Dirac delta distribution, $\forall s \in \mathcal{S}$, the policy is referred to as *deterministic* (Silver et al., 2014) and we write $\mu^i(s)$ to refer to its centre. Then, the environment emits the joint reward $\mathrm{r}_t = r(\mathrm{s}_t, \mathbf{a}_t)$ and moves to the next state $\mathrm{s}_{t+1} \sim P(\cdot|\mathrm{s}_t, \mathbf{a}_t) \in \mathcal{P}(\mathcal{S})$. The joint policy $\boldsymbol{\pi}$, the transition probabililty kernel P, and the initial state distribution d, induce a marginal state distribution at time t, denoted by $\rho_{\boldsymbol{\pi}}^t$. We define an (improper) marginal state distribution $\rho_{\boldsymbol{\pi}} \triangleq \sum_{t=0}^{\infty} \gamma^t \rho_{\boldsymbol{\pi}}^t$. The state value function and the state-action value function are defined as:

$$V_{\boldsymbol{\pi}}(s) \triangleq \mathbb{E}_{\mathbf{a}_{0:\infty} \sim \boldsymbol{\pi}, \mathrm{s}_{1:\infty} \sim P} \big[\sum_{t=0}^{\infty} \gamma^t \mathrm{r}_t \big| \ \mathrm{s}_0 = s \big]$$

and[2]

$$Q_{\boldsymbol{\pi}}(s, \boldsymbol{a}) \triangleq \mathbb{E}_{\mathrm{s}_{1:\infty} \sim P, \mathbf{a}_{1:\infty} \sim \boldsymbol{\pi}} \big[\sum_{t=0}^{\infty} \gamma^t \mathrm{r}_t \big| \ \mathrm{s}_0 = s, \ \mathbf{a}_0 = \boldsymbol{a} \big].$$

The advantage function is defined to be

$$A_{\boldsymbol{\pi}}(s, \boldsymbol{a}) \triangleq Q_{\boldsymbol{\pi}}(s, \boldsymbol{a}) - V_{\boldsymbol{\pi}}(s).$$

In this paper, we consider the fully-cooperative setting where the agents aim to maximise the expected joint return, defined as

$$J(\boldsymbol{\pi}) \triangleq \mathbb{E}_{\mathrm{s}_{0:\infty} \sim \rho_{\boldsymbol{\pi}}^{0:\infty}, \mathbf{a}_{0:\infty} \sim \boldsymbol{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t \mathrm{r}_t \right].$$

We adopt the most common solution concept for multi-agent problems which is that of Nash equilibrium (NE) (Nash, 1951; Yang and Wang, 2020; Filar and Vrieze, 2012; Başar and Olsder, 1998), defined as follows.

Definition 2 *In a fully-cooperative game, a joint policy $\boldsymbol{\pi}_* = (\pi_*^1, \dots, \pi_*^n)$ is a Nash equilibrium (NE) if for every $i \in \mathcal{N}$, $\pi^i \in \Pi^i$ implies $J(\boldsymbol{\pi}_*) \geq J\left(\pi^i, \boldsymbol{\pi}_*^{-i}\right)$.*

2. We write a^i, $\boldsymbol{a}$, and s when we refer to the action, joint action, and state as to values, and a^i, $\mathbf{a}$, and s as to random variables.

NE is a well-established game-theoretic solution concept. Definition 2 characterises the equilibrium point at convergence for cooperative MARL tasks. To study the problem of finding a NE, we pay close attention to the contribution to performance from different subsets of agents. To this end, we introduce the following novel definitions.

Definition 3 *Let $i_{1:m}$ denote an ordered subset $\{i_1, \dots, i_m\}$ of $\mathcal{N}$. We write $-i_{1:m}$ to refer to its complement, and i and $-i$, respectively, when $m = 1$. We write i_k when we refer to the k^{th} agent in the ordered subset. Correspondingly, the multi-agent state-action value function is defined as*

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

*In particular, when $m = n$ (the joint action of all agents is considered), then $i_{1:n} \in Sym(n)$, where $Sym(n)$ denotes the set of permutations of integers $1, \dots, n$, known as the **symmetric group**. In that case, $Q_{\boldsymbol{\pi}}^{i_{1:n}}(s, \boldsymbol{a}^{i_{1:n}})$ is equivalent to $Q_{\boldsymbol{\pi}}(s, \boldsymbol{a})$. On the other hand, when $m = 0$, i.e., $i_{1:m} = \emptyset$, the function takes the form of $V_{\boldsymbol{\pi}}(s)$. Moreover, consider two disjoint subsets of agents, $j_{1:k}$ and $i_{1:m}$. Then, the multi-agent advantage function of $i_{1:m}$ with respect to $j_{1:k}$ is defined as*

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

In words, $Q_{\boldsymbol{\pi}}^{i_{1:m}}\left(s, \boldsymbol{a}^{i_{1:m}}\right)$ evaluates the value of agents $i_{1:m}$ taking actions $\boldsymbol{a}^{i_{1:m}}$ in state s while marginalizing out $\mathbf{a}^{-i_{1:m}}$, and $A_{\boldsymbol{\pi}}^{i_{1:m}}\left(s, \boldsymbol{a}^{j_{1:k}}, \boldsymbol{a}^{i_{1:m}}\right)$ evaluates the advantage of agents $i_{1:m}$ taking actions $\boldsymbol{a}^{i_{1:m}}$ in state s given that the actions taken by agents $j_{1:k}$ are $\boldsymbol{a}^{j_{1:k}}$, with the rest of agents' actions marginalized out by expectation. As we show later in Section 3, these functions allow to decompose the joint advantage function, thus shedding new light on the credit assignment problem.

2.2 Dealing With Partial Observability

Notably, in some cooperative multi-agent tasks, the global state s may be only partially observable to the agents. That is, instead of the omniscient global state, each agent can only perceive a local observation of the environment, which does not satisfy the *Markov property*. The model that accounts for partial observability is Decentralized Partially Observable Markov Decision Process (Dec-POMDP) (Oliehoek and Amato, 2016). However, Dec-POMDP is proved to be NEXP-complete (Bernstein et al., 2002) and requires super-exponential time to solve in the worst case (Zhang et al., 2021). To obtain tractable results, we assume full observability in theoretical derivations and let each agent take actions conditioning on the global state, *i.e.*, $a_t^i \sim \pi^i(\cdot^i|s)$, thereby arriving at practical algorithms. In literature (Yang et al., 2018; Kuba et al., 2021; Wang et al., 2023), this is a common modeling choice for rigor, consistency, and simplicity of the proofs.

In our implementation, we either compensate for partial observability by employing RNN so that agent actions are conditioned on the action-observation history, or directly use the MLP network so that agent actions are conditioned on the partial observations. Both of them are common approaches adopted by existing work, including MAPPO (Yu et al., 2022), QMIX (Rashid et al., 2018), COMA (Foerster et al., 2018), OB (Kuba et al., 2021), MACPF (Wang et al., 2023) etc.. From our experiments (Section 5), we show that both approaches are capable of solving partially observable tasks.

2.3 The State of Affairs in Cooperative MARL

Before we review existing SOTA algorithms for cooperative MARL, we introduce two settings in which the algorithms can be implemented. Both of them can be considered appealing depending on the application, but their benefits also come with limitations which, if not taken care of, may deteriorate an algorithm's performance and applicability.

2.3.1 Homogeneity *vs.* Heterogeneity

The first setting is that of *homogeneous* policies, *i.e.*, those where all agents share one set of policy parameters: $\pi^i = \pi, \forall i \in \mathcal{N}$, so that $\boldsymbol{\pi} = (\pi, \dots, \pi)$ (de Witt et al., 2020; Yu et al., 2022), commonly referred to as *Full Parameter Sharing* (FuPS) (Christianos et al., 2021). This approach enables a straightforward adoption of an RL algorithm to MARL, and it does not introduce much computational and sample complexity burden with the increasing number of agents. As such, it has been a common practice in the MARL community to improve sample efficiency and boost algorithm performance (Sunehag et al., 2018; Foerster et al., 2018; Rashid et al., 2018). However, FuPS could lead to an exponentially-suboptimal outcome in the extreme case (see Example 2 in Appendix A). While agent identity information could be added to observation to alleviate this difficulty, FuPS+id still suffers from interference during agents' learning process in scenarios where they have different abilities and goals, resulting in poor performance, as analysed by Christianos et al. (2021) and shown by our experiments (Figure 8). One remedy is the *Selective Parameter Sharing* (SePS) (Christianos et al., 2021), which only shares parameters among similar agents. Nevertheless, this approach has been shown to be suboptimal and highly scenario-dependent, emphasizing the need for prior understanding of task and agent attributes to effectively utilize the SePS strategy (Hu et al., 2022a). More severely, both FuPS and SePS require the observation and action spaces of agents in a sharing group to be the same, restricting their applicability to the general heterogeneous-agent setting. Existing work that extends parameter sharing to heterogeneous agents relies on *padding* (Terry et al., 2020), which also cannot be generally applied. To summarize, algorithms relying on parameter sharing potentially suffer from compromised performance and applicability.

A more ambitious approach to MARL is to allow for *heterogeneity* of policies among agents, *i.e.*, to let π^i and π^j be different functions when $i \neq j \in \mathcal{N}$. This setting has greater applicability as heterogeneous agents can operate in different action spaces. Furthermore, thanks to this model's flexibility they may learn more sophisticated joint behaviors. Lastly, they can recover homogeneous policies as a result of training, if that is indeed optimal.

Nevertheless, training heterogeneous agents is highly non-trivial. Given a joint reward, an individual agent may not be able to distill its own contribution to it — a problem known as *credit assignment* (Foerster et al., 2018; Kuba et al., 2021). Furthermore, even if an agent identifies its improvement direction, it may conflict with those of other agents when not optimised properly. We provide two examples to illustrate this phenomenon.

The first one is shown in Figure 1. We design a single-state differentiable game where two agents play continuous actions $a^1, a^2 \in \mathbb{R}$ respectively, and the reward function is $r(a^1, a^2) = a^1 a^2$. When we initialise agent policies in the second or fourth quadrants and set a large learning rate, the simultaneous update approach could result in a decrease in joint

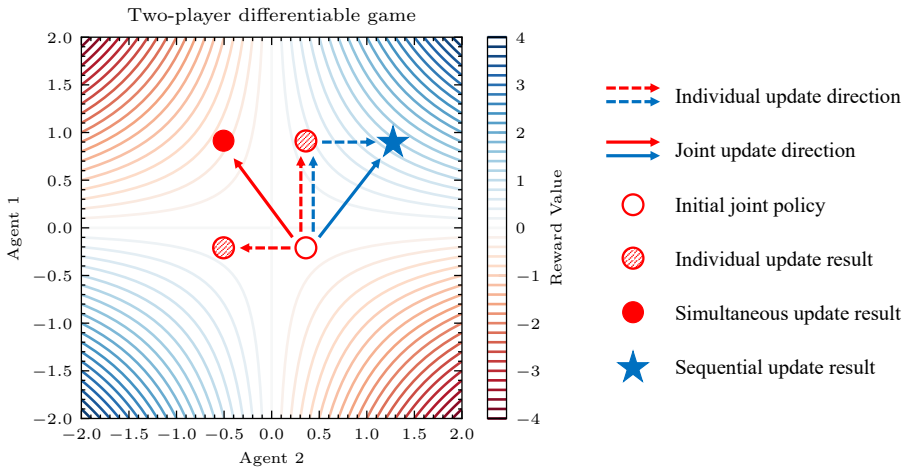


Figure 1: Example of a two-agent differentiable game with $r(a^1, a^2) = a^1 a^2$. We initialise the two policies in the fourth quadrant. Under the straightforward simultaneous update scheme (red), agent 1 takes a positive update to improve the joint reward, meanwhile agent 2 moves towards the negative axis for the same purpose. However, their update directions conflict with each other and lead to a decrease in the joint return. By contrast, under our proposed sequential update scheme (blue), agent 1 updates first, and agent 2 adapts to agent 1' updated policy, jointly leading to improvement.

reward. In contrast, the sequential update proposed in this paper enables agent 2 to fully adapt to agent 1's updated policy and improves the joint reward.

We consider a matrix game with discrete action space as the second example. Our matrix game is illustrated as follows:

Example 1 *Let's consider a fully-cooperative game with* 2 *agents, one state, and the joint action space* $\{0, 1\}^2$*, where the reward is given by* $r(0, 0) = 0, r(0, 1) = r(1, 0) = 2$*, and* $r(1, 1) = -1$*. Suppose that* $\pi^i_{old}(0) > 0.6$ *for* $i = 1, 2$*. Then, if agents* i *update their policies by*

$$\pi^i_{new} = \arg\max_{\pi^i} \mathbb{E}_{\mathrm{a}^i \sim \pi^i, \mathrm{a}^{-i} \sim \pi^{-i}_{old}} \left[A_{\boldsymbol{\pi}_{old}}(\mathrm{a}^i, \mathrm{a}^{-i}) \right], \forall i \in \mathcal{N},$$

then the resulting policy will yield a lower return,

$$J(\boldsymbol{\pi}_{old}) > J(\boldsymbol{\pi}_{new}) = \min_{\boldsymbol{\pi}} J(\boldsymbol{\pi}).$$

This example helpfully illustrates the miscoordination problem when agents conduct independent reward maximisation simultaneously. A similar miscoordination problem when heterogeneous agents update at the same time is also shown in Example 2 of Alós-Ferrer and Netzer (2010).

Therefore, our discussion in this section not only implies that homogeneous algorithms could have restricted performance and applicability, but also highlight that heterogeneous algorithms should be developed with extra care when not optimised properly (large learning

rate in Figure 1 and independent reward maximisation in Example 1), which could be common in complex high-dimensional problems. In the next subsection, we describe existing SOTA actor-critic algorithms which, while often very effective, are still not impeccable, as they suffer from one of the above two limitations.

2.3.2 Analysis of Existing Work

MAA2C (Papoudakis et al., 2021) extends the A2C (Mnih et al., 2016) to MARL by replacing the RL optimisation (single-agent policy) objective with the MARL one (joint policy),

$$\mathcal{L}^{\text{MAA2C}}(\boldsymbol{\pi}) \triangleq \mathbb{E}_{\text{s}\sim\boldsymbol{\pi},\mathbf{a}\sim\boldsymbol{\pi}}\big[A_{\boldsymbol{\pi}_{\text{old}}}(\text{s},\mathbf{a})\big], \tag{2}$$

which computes the gradient with respect to every agent i's policy parameters, and performs a gradient-ascent update for each agent. This algorithm is straightforward to implement and is capable of solving simple multi-agent problems (Papoudakis et al., 2021). We point out, however, that by simply following their own MAPG, the agents could perform uncoordinated updates, as illustrated in Figure 1. Furthermore, MAPG estimates have been proved to suffer from large variance which grows linearly with the number of agents (Kuba et al., 2021), thus making the algorithm unstable. To assure greater stability, the following MARL methods, inspired by stable RL approaches, have been developed.

MADDPG (Lowe et al., 2017) is a MARL extension of the popular DDPG algorithm (Lillicrap et al., 2016). At every iteration, every agent i updates its deterministic policy by maximising the following objective

$$\mathcal{L}_i^{\text{MADDPG}}(\mu^i) \triangleq \mathbb{E}_{\text{s}\sim\beta_{\boldsymbol{\mu}_{\text{old}}}}\Big[Q^i_{\boldsymbol{\mu}_{\text{old}}}\big(\text{s},\mu^i(\text{s})\big)\Big] = \mathbb{E}_{\text{s}\sim\beta_{\boldsymbol{\mu}_{\text{old}}}}\Big[Q_{\boldsymbol{\mu}_{\text{old}}}\big(\text{s},\mu^i(\text{s}),\boldsymbol{\mu}^{-i}_{\text{old}}(\text{s})\big)\Big], \tag{3}$$

where $\beta_{\boldsymbol{\mu}_{\text{old}}}$ is a state distribution that is not necessarily equivalent to $\rho_{\boldsymbol{\mu}_{\text{old}}}$, thus allowing for off-policy training. In practice, MADDPG maximises Equation (3) by a few steps of gradient ascent. The main advantages of MADDPG include a small variance of its MAPG estimates—a property granted by deterministic policies (Silver et al., 2014), as well as low sample complexity due to learning from off-policy data. Such a combination makes the algorithm competitive on certain continuous-action tasks (Lowe et al., 2017). However, MADDPG does not address the multi-agent credit assignment problem (Foerster et al., 2018). Plus, when training the decentralised actors, MADDPG does not take into account the updates agents have made and naively uses the off-policy data from the replay buffer which, much like in Section 2.3.1, leads to uncoordinated updates and suboptimal performance in the face of harder tasks (Peng et al., 2021; Ray-Team, accessed on 2023-03-14). MATD3 (Ackermann et al., 2019) proposes to reduce overestimation bias in MADDPG using double centralized critics, which improves its performance and stability but does not help with getting rid of the aforementioned limitations.

MAPPO (Yu et al., 2022) is a relatively straightforward extension of PPO (Schulman et al., 2017) to MARL. In its default formulation, the agents employ the trick of *parameter sharing* described in the previous subsection. As such, the policy is updated to maximise

$$\mathcal{L}^{\text{MAPPO}}(\pi) \triangleq \mathbb{E}_{\text{s}\sim\rho_{\boldsymbol{\pi}_{\text{old}}},\mathbf{a}\sim\boldsymbol{\pi}_{\text{old}}}\Bigg[\sum_{i=1}^{n}\min\Big(\frac{\pi(\text{a}^i|\text{s})}{\pi_{\text{old}}(\text{a}^i|\text{s})}A_{\boldsymbol{\pi}_{\text{old}}}(\text{s},\mathbf{a}),\,\text{clip}\big(\frac{\pi(\text{a}^i|\text{s})}{\pi_{\text{old}}(\text{a}^i|\text{s})},1\pm\epsilon\big)A_{\boldsymbol{\pi}_{\text{old}}}(\text{s},\mathbf{a})\Big)\Bigg], \tag{4}$$

where the clip$(\cdot, 1 \pm \epsilon)$ operator clips the input to $1-\epsilon/1+\epsilon$ if it is below/above this value. Such an operation removes the incentive for agents to make large policy updates, thus stabilising the training effectively. Indeed, the algorithm's performance on the StarCraftII benchmark is remarkable, and it is accomplished by using only on-policy data. Nevertheless, the parameter-sharing strategy limits the algorithm's applicability and could lead to its suboptimality when agents have different roles. In trying to avoid this issue, one can implement the algorithm without parameter sharing, thus making the agents simply take simultaneous PPO updates meanwhile employing a joint advantage estimator. In this case, the updates could be uncoordinated, as we discussed in Section 2.3.1.

In summary, all these algorithms do not possess performance guarantees. Altering their implementation settings to avoid one of the limitations from Section 2.3.1 makes them, at best, fall into another. This shows that the MARL problem introduces additional complexity into the single-agent RL setting, and needs additional care to be rigorously solved. With this motivation, in the next section, we propose novel heterogeneous-agent methods based on *sequential update* with correctness guarantees.

3. Our Methods

The purpose of this section is to introduce Heterogeneous-Agent Reinforcement Learning (HARL) algorithm series which we prove to solve cooperative problems theoretically. HARL algorithms are designed for the general and expressive setting of heterogeneous agents, and their essence is to coordinate agents' updates, thus resolving the challenges in Section 2.3.1. We start by developing a theoretically justified Heterogeneous-Agent Trust Region Learning (HATRL) procedure in Section 3.1 and deriving practical algorithms, namely HATRPO and HAPPO, as its tractable approximations in Section 3.2. We further introduce the novel Heterogeneous-Agent Mirror Learning (HAML) framework in Section 3.3, which strengthens performance guarantees of HATRPO and HAPPO (Section 3.4) and provides a general template for cooperative MARL algorithmsic design, leading to more HARL algorithms (Section 3.5).

3.1 Heterogeneous-Agent Trust Region Learning (HATRL)

Intuitively, if we parameterise all agents separately and let them learn one by one, then we will break the homogeneity constraint and allow the agents to coordinate their updates, thereby avoiding the two limitations from Section 2.3. Such coordination can be achieved, for example, by accounting for previous agents' updates in the optimization objective of the current one along the aforementioned sequence. Fortunately, this idea is embodied in the multi-agent advantage function $A^{i_m}_{\boldsymbol{\pi}}\left(s, \boldsymbol{a}^{i_{1:m-1}}, a^{i_m}\right)$ which allows agent i_m to evaluate the utility of its action a^{i_m} given actions of previous agents $\boldsymbol{a}^{i_{1:m-1}}$. Intriguingly, multi-agent advantage functions allow for rigorous decomposition of the joint advantage function, as described by the following pivotal lemma.

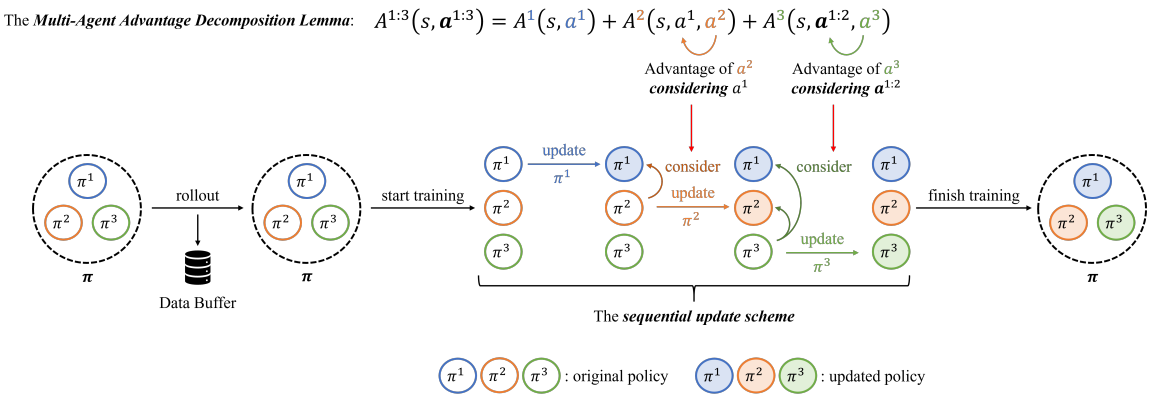


Figure 2: The *multi-agent advantage decomposition lemma* and the *sequential update scheme* are naturally consistent. The former (upper in the figure) decomposes joint advantage into sequential advantage evaluations, each of which takes into consideration previous agents' actions. Based on this, the latter (lower in the figure) allows each policy to be updated considering previous updates during the training stage. The rigor of their connection is embodied in Lemma 6 and Lemma 13, where multi-agent advantage decomposition lemma is crucial for the proofs and leads to algorithms that employ sequential update scheme.

Lemma 4 (Multi-Agent Advantage Decomposition) *In any cooperative Markov games, given a joint policy $\boldsymbol{\pi}$, for any state s, and any agent subset $i_{1:m}$, the below equation holds.*

$$A_{\boldsymbol{\pi}}^{i_{1:m}}\left(s, \boldsymbol{a}^{i_{1:m}}\right) = \sum_{j=1}^{m} A_{\boldsymbol{\pi}}^{i_j}\left(s, \boldsymbol{a}^{i_{1:j-1}}, a^{i_j}\right).$$

For proof see Appendix B. Notably, Lemma 4 holds in general for cooperative Markov games, with no need for any assumptions on the decomposability of the joint value function such as those in VDN (Sunehag et al., 2018), QMIX (Rashid et al., 2018) or Q-DPP (Yang et al., 2020).

Lemma 4 confirms that a sequential update is an effective approach to search for the direction of performance improvement (i.e., joint actions with positive advantage values) in multi-agent learning. That is, imagine that agents take actions sequentially by following an arbitrary order $i_{1:n}$. Let agent i_1 take action $\bar{a}^{i_1}$ such that $A_{\boldsymbol{\pi}}^{i_1}(s, \bar{a}^{i_1}) > 0$, and then, for the remaining $m = 2, \ldots, n$, each agent i_m takes an action $\bar{a}^{i_m}$ such that $A_{\boldsymbol{\pi}}^{i_m}(s, \bar{\boldsymbol{a}}^{i_{1:m-1}}, \bar{a}^{i_m}) > 0$. For the induced joint action $\bar{\boldsymbol{a}}$, Lemma 4 assures that $A_{\boldsymbol{\pi}}(s, \bar{\boldsymbol{a}})$ is positive, thus the performance is guaranteed to improve. To formally extend the above process into a policy iteration procedure with monotonic improvement guarantee, we begin by introducing the following definitions.

Definition 5 *Let $\boldsymbol{\pi}$ be a joint policy, $\bar{\boldsymbol{\pi}}^{i_{1:m-1}} = \prod_{j=1}^{m-1} \bar{\pi}^{i_j}$ be some **other** joint policy of agents $i_{1:m-1}$, and $\hat{\pi}^{i_m}$ be some **other** policy of agent i_m. Then*

$$L_{\boldsymbol{\pi}}^{i_{1:m}}\left(\bar{\boldsymbol{\pi}}^{i_{1:m-1}}, \hat{\pi}^{i_m}\right) \triangleq \mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}}, \mathbf{a}^{i_{1:m-1}}\sim\bar{\boldsymbol{\pi}}^{i_{1:m-1}}, \mathrm{a}^{i_m}\sim\hat{\pi}^{i_m}}\left[A_{\boldsymbol{\pi}}^{i_m}\left(\mathrm{s}, \mathbf{a}^{i_{1:m-1}}, \mathrm{a}^{i_m}\right)\right].$$

Note that, for any $\bar{\boldsymbol{\pi}}^{i_{1:m-1}}$, we have

$$L_{\boldsymbol{\pi}}^{i_{1:m}}\left(\bar{\boldsymbol{\pi}}^{i_{1:m-1}}, \pi^{i_m}\right) = \mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}}, \mathbf{a}^{i_{1:m-1}}\sim\bar{\boldsymbol{\pi}}^{i_{1:m-1}}, \mathrm{a}^{i_m}\sim\pi^{i_m}}\left[A_{\boldsymbol{\pi}}^{i_m}\left(\mathrm{s}, \mathbf{a}^{i_{1:m-1}}, \mathrm{a}^{i_m}\right)\right]$$
$$= \mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}}, \mathbf{a}^{i_{1:m-1}}\sim\bar{\boldsymbol{\pi}}^{i_{1:m-1}}}\left[\mathbb{E}_{\mathrm{a}^{i_m}\sim\pi^{i_m}}\left[A_{\boldsymbol{\pi}}^{i_m}\left(\mathrm{s}, \mathbf{a}^{i_{1:m-1}}, \mathrm{a}^{i_m}\right)\right]\right] = 0. \quad (5)$$

Building on Lemma 4 and Definition 5, we derive the bound for joint policy update.

Lemma 6 *Let $\boldsymbol{\pi}$ be a joint policy. Then, for any joint policy $\bar{\boldsymbol{\pi}}$, we have*

$$J(\bar{\boldsymbol{\pi}}) \geq J(\boldsymbol{\pi}) + \sum_{m=1}^{n}\left[L_{\boldsymbol{\pi}}^{i_{1:m}}\left(\bar{\boldsymbol{\pi}}^{i_{1:m-1}}, \bar{\pi}^{i_m}\right) - C\mathrm{D}_{KL}^{max}(\pi^{i_m}, \bar{\pi}^{i_m})\right],$$
$$\text{where } C = \frac{4\gamma \max_{s,\boldsymbol{a}}|A_{\boldsymbol{\pi}}(s,\boldsymbol{a})|}{(1-\gamma)^2}. \quad (6)$$

For proof see Appendix B.2. This lemma provides an idea about how a joint policy can be improved. Namely, by Equation (5), we know that if any agents were to set the values of the above summands $L_{\boldsymbol{\pi}}^{i_{1:m}}(\bar{\boldsymbol{\pi}}^{i_{1:m-1}}, \bar{\pi}^{i_m}) - C\mathrm{D}_{\mathrm{KL}}^{\max}(\pi^{i_m}, \bar{\pi}^{i_m})$ by sequentially updating their policies, each of them can always make its summand be zero by making no policy update (i.e., $\bar{\pi}^{i_m} = \pi^{i_m}$). This implies that any positive update will lead to an increment in summation. Moreover, as there are n agents making policy updates, the compound increment can be large, leading to a substantial improvement. Lastly, note that this property holds with no requirement on the specific order by which agents make their updates; this allows for flexible scheduling on the update order at each iteration. To summarise, we propose the following Algorithm 1.

Algorithm 1: Multi-Agent Policy Iteration with Monotonic Improvement Guarantee

Initialise the joint policy $\boldsymbol{\pi}_0 = (\pi_0^1, \ldots, \pi_0^n)$.
for $k = 0, 1, \ldots$ **do**
  Compute the advantage function $A_{\boldsymbol{\pi}_k}(s, \boldsymbol{a})$ for all state-(joint)action pairs $(s, \boldsymbol{a})$.
  Compute $\epsilon = \max_{s,\boldsymbol{a}}|A_{\boldsymbol{\pi}_k}(s,\boldsymbol{a})|$ and $C = \frac{4\gamma\epsilon}{(1-\gamma)^2}$.
  Draw a permutation $i_{1:n}$ of agents at random.
  for $m = 1 : n$ **do**
    Make an update
    $\pi_{k+1}^{i_m} = \arg\max_{\pi^{i_m}}\left[L_{\boldsymbol{\pi}_k}^{i_{1:m}}\left(\boldsymbol{\pi}_{k+1}^{i_{1:m-1}}, \pi^{i_m}\right) - C\mathrm{D}_{\mathrm{KL}}^{\max}(\pi_k^{i_m}, \pi^{i_m})\right]$.

We want to highlight that the algorithm is markedly different from naively applying the TRPO update on the joint policy of all agents. Firstly, our Algorithm 1 does not update the entire joint policy at once, but rather updates each agent's individual policy sequentially. Secondly, during the sequential update, each agent has a unique optimisation objective that takes into account all previous agents' updates, which is also the key for the monotonic improvement property to hold. We justify by the following theorem that Algorithm 1 enjoys monotonic improvement property.

Theorem 7 *A sequence $(\boldsymbol{\pi}_k)_{k=0}^{\infty}$ of joint policies updated by Algorithm 1 has the monotonic improvement property, i.e., $J(\boldsymbol{\pi}_{k+1}) \geq J(\boldsymbol{\pi}_k)$ for all $k \in \mathbb{N}$.*

For proof see Appendix B.2. With the above theorem, we claim a successful development of Heterogeneous-Agent Trust Region Learning (HATRL), as it retains the monotonic improvement property of trust region learning. Moreover, we take a step further to prove Algorithm 1's asymptotic convergence behavior towards NE.

Theorem 8 *Supposing in Algorithm 1 any permutation of agents has a fixed non-zero probability to begin the update, a sequence $(\boldsymbol{\pi}_k)_{k=0}^{\infty}$ of joint policies generated by the algorithm, in a cooperative Markov game, has a non-empty set of limit points, each of which is a Nash equilibrium.*

For proof see Appendix B.3. In deriving this result, the novel details introduced by Algorithm 1 played an important role. The monotonic improvement property (Theorem 7), achieved through the multi-agent advantage decomposition lemma and the sequential update scheme, provided us with a guarantee of the convergence of the return. Furthermore, randomisation of the update order ensured that, at convergence, none of the agents is incentified to make an update. The proof is finalised by excluding the possibility that the algorithm converges at non-equilibrium points.

3.2 Practical Algorithms

When implementing Algorithm 1 in practice, large state and action spaces could prevent agents from designating policies $\pi^i(\cdot|s)$ for each state s separately. To handle this, we parameterise each agent's policy $\pi^i_{\theta^i}$ by θ^i, which, together with other agents' policies, forms a joint policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ parametrised by $\boldsymbol{\theta} = (\theta^1, \dots, \theta^n)$. In this subsection, we develop two deep MARL algorithms to optimise the $\boldsymbol{\theta}$.

3.2.1 HATRPO

Computing $\mathrm{D}^{\max}_{\mathrm{KL}}\big(\pi^{i_m}_{\theta^{i_m}_k}, \pi^{i_m}_{\theta^{i_m}}\big)$ in Algorithm 1 is challenging; it requires evaluating the KL-divergence for all states at each iteration. Similar to TRPO, one can ease this maximal KL-divergence penalty $\mathrm{D}^{\max}_{\mathrm{KL}}\big(\pi^{i_m}_{\theta^{i_m}_k}, \pi^{i_m}_{\theta^{i_m}}\big)$ by replacing it with the expected KL-divergence constraint $\mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}_{\boldsymbol{\theta}_k}}}\Big[\mathrm{D}_{\mathrm{KL}}\big(\pi^{i_m}_{\theta^{i_m}_k}(\cdot|\mathrm{s}), \pi^{i_m}_{\theta^{i_m}}(\cdot|\mathrm{s})\big)\Big] \leq \delta$ where δ is a threshold hyperparameter and the expectation can be easily approximated by stochastic sampling. With the above amendment, we propose practical HATRPO algorithm in which, at every iteration $k+1$, given a permutation of agents $i_{1:n}$, agent $i_{m\in\{1,\dots,n\}}$ sequentially optimises its policy parameter $\theta^{i_m}_{k+1}$ by maximising a constrained objective:

$$\theta^{i_m}_{k+1} = \arg\max_{\theta^{i_m}} \mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}_{\boldsymbol{\theta}_k}}, \mathbf{a}^{i_{1:m-1}}\sim\boldsymbol{\pi}^{i_{1:m-1}}_{\boldsymbol{\theta}^{i_{1:m-1}}_{k+1}}, \mathrm{a}^{i_m}\sim\pi^{i_m}_{\theta^{i_m}}}\Big[A^{i_m}_{\boldsymbol{\pi}_{\boldsymbol{\theta}_k}}(\mathrm{s}, \mathbf{a}^{i_{1:m-1}}, \mathrm{a}^{i_m})\Big],$$
$$\text{subject to } \mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}_{\boldsymbol{\theta}_k}}}\Big[\mathrm{D}_{\mathrm{KL}}\big(\pi^{i_m}_{\theta^{i_m}_k}(\cdot|\mathrm{s}), \pi^{i_m}_{\theta^{i_m}}(\cdot|\mathrm{s})\big)\Big] \leq \delta. \tag{7}$$

To compute the above equation, similar to TRPO, one can apply a linear approximation to the objective function and a quadratic approximation to the KL constraint; the optimisation

problem in Equation (7) can be solved by a closed-form update rule as

$$\theta_{k+1}^{i_m} = \theta_k^{i_m} + \alpha^j \sqrt{\frac{2\delta}{\boldsymbol{g}_k^{i_m}(\boldsymbol{H}_k^{i_m})^{-1}\boldsymbol{g}_k^{i_m}}}(\boldsymbol{H}_k^{i_m})^{-1}\boldsymbol{g}_k^{i_m} \ , \tag{8}$$

where $\boldsymbol{H}_k^{i_m} = \nabla^2_{\theta^{i_m}} \mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}_{\boldsymbol{\theta}_k}}}\big[\mathrm{D}_{\mathrm{KL}}\big(\pi^{i_m}_{\theta_k^{i_m}}(\cdot|\mathrm{s}), \pi^{i_m}_{\theta^{i_m}}(\cdot|\mathrm{s})\big)\big]\big|_{\theta^{i_m}=\theta_k^{i_m}}$ is the Hessian of the expected KL-divergence, $\boldsymbol{g}_k^{i_m}$ is the gradient of the objective in Equation (7), $\alpha^j < 1$ is a positive coefficient that is found via backtracking line search, and the product of $(\boldsymbol{H}_k^{i_m})^{-1}\boldsymbol{g}_k^{i_m}$ can be efficiently computed with conjugate gradient algorithm.

Estimating $\mathbb{E}_{\mathbf{a}^{i_{1:m-1}}\sim\boldsymbol{\pi}^{i_{1:m-1}}_{\boldsymbol{\theta}_{k+1}},\mathrm{a}^{i_m}\sim\pi^{i_m}_{\theta^{i_m}}}\Big[A^{i_m}_{\boldsymbol{\pi}_{\boldsymbol{\theta}_k}}\big(\mathrm{s}, \mathbf{a}^{i_{1:m-1}}, \mathrm{a}^{i_m}\big)\Big]$ is the last missing piece for HATRPO, which poses new challenges because each agent's objective has to take into account all previous agents' updates, and the size of input values. Fortunately, with the following proposition, we can efficiently estimate this objective by a joint advantage estimator.

Proposition 9 *Let $\boldsymbol{\pi} = \prod_{j=1}^{n}\pi^{i_j}$ be a joint policy, and $A_{\boldsymbol{\pi}}(\mathrm{s}, \mathbf{a})$ be its joint advantage function. Let $\bar{\boldsymbol{\pi}}^{i_{1:m-1}} = \prod_{j=1}^{m-1}\bar{\pi}^{i_j}$ be some **other** joint policy of agents $i_{1:m-1}$, and $\hat{\pi}^{i_m}$ be some **other** policy of agent i_m. Then, for every state s,*

$$\begin{aligned}&\mathbb{E}_{\mathbf{a}^{i_{1:m-1}}\sim\bar{\boldsymbol{\pi}}^{i_{1:m-1}},\mathrm{a}^{i_m}\sim\hat{\pi}^{i_m}}\big[A^{i_m}_{\boldsymbol{\pi}}\big(s, \mathbf{a}^{i_{1:m-1}}, \mathrm{a}^{i_m}\big)\big] \\ &\qquad = \mathbb{E}_{\mathbf{a}\sim\boldsymbol{\pi}}\Big[\Big(\frac{\hat{\pi}^{i_m}(\mathrm{a}^{i_m}|s)}{\pi^{i_m}(\mathrm{a}^{i_m}|s)} - 1\Big)\frac{\bar{\boldsymbol{\pi}}^{i_{1:m-1}}(\mathbf{a}^{i_{1:m-1}}|s)}{\boldsymbol{\pi}^{i_{1:m-1}}(\mathbf{a}^{i_{1:m-1}}|s)}A_{\boldsymbol{\pi}}(s, \mathbf{a})\Big].\end{aligned} \tag{9}$$

For proof see Appendix C.1. One benefit of applying Equation (9) is that agents only need to maintain a joint advantage estimator $A_{\boldsymbol{\pi}}(\mathrm{s}, \mathbf{a})$ rather than one centralised critic for each individual agent (e.g., unlike CTDE methods such as MADDPG). Another practical benefit one can draw is that, given an estimator $\hat{A}(\mathrm{s}, \mathbf{a})$ of the advantage function $A_{\boldsymbol{\pi}_{\boldsymbol{\theta}_k}}(\mathrm{s}, \mathbf{a})$, for example, GAE (Schulman et al., 2016), $\mathbb{E}_{\mathbf{a}^{i_{1:m-1}}\sim\boldsymbol{\pi}^{i_{1:m-1}}_{\boldsymbol{\theta}^{i_{1:m-1}}_{k+1}},\mathrm{a}^{i_m}\sim\pi^{i_m}_{\theta^{i_m}}}\Big[A^{i_m}_{\boldsymbol{\pi}_{\boldsymbol{\theta}_k}}\big(s, \mathbf{a}^{i_{1:m-1}}, \mathrm{a}^{i_m}\big)\Big]$ can be estimated with an estimator of

$$\Big(\frac{\pi^{i_m}_{\theta^{i_m}}(\mathrm{a}^{i_m}|s)}{\pi^{i_m}_{\theta^{i_m}_k}(\mathrm{a}^{i_m}|s)} - 1\Big)M^{i_{1:m}}\big(s, \mathbf{a}\big), \qquad \text{where} \quad M^{i_{1:m}} = \frac{\boldsymbol{\pi}^{i_{1:m-1}}_{\boldsymbol{\theta}^{i_{1:m-1}}_{k+1}}(\mathbf{a}^{i_{1:m-1}}|s)}{\boldsymbol{\pi}^{i_{1:m-1}}_{\boldsymbol{\theta}^{i_{1:m-1}}_{k}}(\mathbf{a}^{i_{1:m-1}}|s)}\hat{A}\big(s, \mathbf{a}\big). \tag{10}$$

Notably, Equation (10) aligns nicely with the sequential update scheme in HATRPO. For agent i_m, since previous agents $i_{1:m-1}$ have already made their updates, the compound policy ratio for $M^{i_{1:m}}$ in Equation (10) is easy to compute. Given a batch $\mathcal{B}$ of trajectories with length T, we can estimate the gradient with respect to policy parameters (derived in Appendix C.2) as follows,

$$\hat{\boldsymbol{g}}_k^{i_m} = \frac{1}{|\mathcal{B}|}\sum_{\tau\in\mathcal{B}}\sum_{t=0}^{T} M^{i_{1:m}}(\mathrm{s}_t, \mathbf{a}_t)\nabla_{\theta^{i_m}}\log\pi^{i_m}_{\theta^{i_m}}(\mathrm{a}^{i_m}_t|\mathrm{s}_t)\big|_{\theta^{i_m}=\theta^{i_m}_k}.$$

The term $-1 \cdot M^{i_{1:m}}(\mathrm{s}, \mathbf{a})$ of Equation (10) is not reflected in $\hat{\boldsymbol{g}}_k^{i_m}$, as it only introduces a constant with zero gradient. Along with the Hessian of the expected KL-divergence, *i.e.*, $\boldsymbol{H}_k^{i_m}$, we can update $\theta_{k+1}^{i_m}$ by following Equation (8). The detailed pseudocode of HATRPO is listed in Appendix C.3.

3.2.2 HAPPO

To further alleviate the computation burden from $\boldsymbol{H}_k^{i_m}$ in HATRPO, one can follow the idea of PPO by considering only using first-order derivatives. This is achieved by making agent i_m choose a policy parameter $\theta_{k+1}^{i_m}$ which maximises the clipping objective of

$$\mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}_{\boldsymbol{\theta}_k}},\mathbf{a}\sim\boldsymbol{\pi}_{\boldsymbol{\theta}_k}}\left[\min\left(\frac{\pi_{\theta^{i_m}}^{i_m}(\mathrm{a}^{i_m}|\mathrm{s})}{\pi_{\theta_k^{i_m}}^{i_m}(\mathrm{a}^{i_m}|\mathrm{s})}M^{i_{1:m}}(\mathrm{s},\mathbf{a}), \mathrm{clip}\left(\frac{\pi_{\theta^{i_m}}^{i_m}(\mathrm{a}^{i_m}|\mathrm{s})}{\pi_{\theta_k^{i_m}}^{i_m}(\mathrm{a}^{i_m}|\mathrm{s})}, 1\pm\epsilon\right)M^{i_{1:m}}(\mathrm{s},\mathbf{a})\right)\right]. \tag{11}$$

The optimisation process can be performed by stochastic gradient methods such as Adam (Kingma and Ba, 2015). We refer to the above procedure as HAPPO and Appendix C.4 for its full pseudocode.

3.3 Heterogeneous-Agent Mirror Learning: A Continuum of Solutions to Cooperative MARL

Recently, Mirror Learning (Kuba et al., 2022b) provided a theoretical explanation of the effectiveness of TRPO and PPO in addition to the original trust region interpretation, and unifies a class of policy optimisation algorithms. Inspired by their work, we further discover a novel theoretical framework for cooperative MARL, named *Heterogeneous-Agent Mirror Learning* (HAML), which enhances theoretical guarantees of HATRPO and HAPPO. As a proven template for algorithmic designs, HAML substantially generalises the desired guarantees of monotonic improvement and NE convergence to a continuum of algorithms and naturally hosts HATRPO and HAPPO as its instances, further explaining their robust performance. We begin by introducing the necessary definitions of HAML attributes: the drift functional.

Definition 10 *Let $i \in \mathcal{N}$, a **heterogeneous-agent drift functional** (HADF) $\mathfrak{D}^i$ of i consists of a map, which is defined as*

$$\mathfrak{D}^i : \boldsymbol{\Pi} \times \boldsymbol{\Pi} \times \mathbb{P}(-i) \times \mathcal{S} \to \{\mathfrak{D}^i_{\boldsymbol{\pi}}(\cdot|s, \bar{\boldsymbol{\pi}}^{j_{1:m}}) : \mathcal{P}(\mathcal{A}^i) \to \mathbb{R}\},$$

such that for all arguments, under notation $\mathfrak{D}^i_{\boldsymbol{\pi}}\big(\hat{\pi}^i|s, \bar{\boldsymbol{\pi}}^{j_{1:m}}\big) \triangleq \mathfrak{D}^i_{\boldsymbol{\pi}}\big(\hat{\pi}^i(\cdot^i|s)|s, \bar{\boldsymbol{\pi}}^{j_{1:m}}(\cdot|s)\big)$,

1. *$\mathfrak{D}^i_{\boldsymbol{\pi}}\big(\hat{\pi}^i|s, \bar{\boldsymbol{\pi}}^{j_{1:m}}\big) \geq \mathfrak{D}^i_{\boldsymbol{\pi}}\big(\pi^i|s, \bar{\boldsymbol{\pi}}^{j_{1:m}}\big) = 0$ (non-negativity),*
2. *$\mathfrak{D}^i_{\boldsymbol{\pi}}\big(\hat{\pi}^i|s, \bar{\boldsymbol{\pi}}^{j_{1:m}}\big)$ has all Gâteaux derivatives zero at $\hat{\pi}^i = \pi^i$ (zero gradient).*

We say that the HADF is positive if $\mathfrak{D}^i_{\boldsymbol{\pi}}(\hat{\pi}^i|s, \bar{\boldsymbol{\pi}}^{j_{1:m}}) = 0, \forall s \in \mathcal{S}$ implies $\hat{\pi}^i = \pi^i$, and trivial if $\mathfrak{D}^i_{\boldsymbol{\pi}}(\hat{\pi}^i|s, \bar{\boldsymbol{\pi}}^{j_{1:m}}) = 0, \forall s \in \mathcal{S}$ for all $\boldsymbol{\pi}, \bar{\boldsymbol{\pi}}^{j_{1:m}}$, and $\hat{\pi}^i$.

Intuitively, the drift $\mathfrak{D}^i_{\boldsymbol{\pi}}(\hat{\pi}^i|s, \bar{\boldsymbol{\pi}}^{j_{1:m}})$ is a notion of distance between π^i and $\hat{\pi}^i$, given that agents $j_{1:m}$ just updated to $\bar{\boldsymbol{\pi}}^{j_{1:m}}$. We highlight that, under this conditionality, the same update (from π^i to $\hat{\pi}^i$) can have different sizes—this will later enable HAML agents to *softly* constraint their learning steps in a coordinated way. Before that, we introduce a notion that renders *hard* constraints, which may be a part of an algorithm design, or an inherent limitation.

Definition 11 *Let $i \in \mathcal{N}$. We say that, $\mathcal{U}^i : \boldsymbol{\Pi} \times \Pi^i \to \mathbb{P}(\Pi^i)$ is a neighbourhood operator if $\forall \pi^i \in \Pi^i$, $\mathcal{U}^i_{\boldsymbol{\pi}}(\pi^i)$ contains a closed ball,* i.e.*, there exists a state-wise monotonically non-decreasing metric $\chi : \Pi^i \times \Pi^i \to \mathbb{R}$ such that $\forall \pi^i \in \Pi^i$ there exists $\delta^i > 0$ such that $\chi(\pi^i, \bar{\pi}^i) \leq \delta^i \implies \bar{\pi}^i \in \mathcal{U}^i_{\boldsymbol{\pi}}(\pi^i)$.*

For every joint policy $\boldsymbol{\pi}$, we will associate it with its *sampling distribution*—a positive state distribution $\beta_{\boldsymbol{\pi}} \in \mathcal{P}(\mathcal{S})$ that is continuous in $\boldsymbol{\pi}$ (Kuba et al., 2022b). With these notions defined, we introduce the main definition for HAML framework.

Definition 12 *Let $i \in \mathcal{N}$, $j^{1:m} \in \mathbb{P}(-i)$, and $\mathfrak{D}^i$ be a HADF of agent i. The **heterogeneous-agent mirror operator** (HAMO) integrates the advantage function as*

$$\big[\mathcal{M}^{(\hat{\pi}^i)}_{\mathfrak{D}^i, \bar{\boldsymbol{\pi}}^{j_{1:m}}} A_{\boldsymbol{\pi}}\big](s) \triangleq \mathbb{E}_{\mathbf{a}^{j_{1:m}} \sim \bar{\boldsymbol{\pi}}^{j_{1:m}}, \mathrm{a}^i \sim \hat{\pi}^i}\Big[A^i_{\boldsymbol{\pi}}(s, \mathbf{a}^{j_{1:m}}, \mathrm{a}^i)\Big] - \mathfrak{D}^i_{\boldsymbol{\pi}}\Big(\hat{\pi}^i\Big|s, \bar{\boldsymbol{\pi}}^{j_{1:m}}\Big).$$

Note that when $\hat{\pi}^i = \pi^i$, HAMO evaluates to zero. Therefore, as the HADF is non-negative, a policy $\hat{\pi}^i$ that improves HAMO must make it positive and thus leads to the improvement of the multi-agent advantage of agent i. It turns out that, under certain configurations, agents' local improvements result in the joint improvement of all agents, as described by the lemma below, proved in Appendix D.

Lemma 13 (HAMO Is All You Need) *Let $\boldsymbol{\pi}_{old}$ and $\boldsymbol{\pi}_{new}$ be joint policies and let $i_{1:n} \in \mathrm{Sym}(n)$ be an agent permutation. Suppose that, for every state $s \in \mathcal{S}$ and every $m = 1, \ldots, n$,*

$$\big[\mathcal{M}^{(\pi^{i_m}_{new})}_{\mathfrak{D}^{i_m}, \boldsymbol{\pi}_{new}^{i_{1:m-1}}} A_{\boldsymbol{\pi}_{old}}\big](s) \geq \big[\mathcal{M}^{(\pi^{i_m}_{old})}_{\mathfrak{D}^{i_m}, \boldsymbol{\pi}_{new}^{i_{1:m-1}}} A_{\boldsymbol{\pi}_{old}}\big](s). \tag{12}$$

Then, $\boldsymbol{\pi}_{new}$ is jointly better than $\boldsymbol{\pi}_{old}$, so that for every state s,

$$V_{\boldsymbol{\pi}_{new}}(s) \geq V_{\boldsymbol{\pi}_{old}}(s).$$

Subsequently, the *monotonic improvement property* of the joint return follows naturally, as

$$J(\boldsymbol{\pi}_{\mathrm{new}}) = \mathbb{E}_{\mathrm{s}\sim d}\big[V_{\boldsymbol{\pi}_{\mathrm{new}}}(s)\big] \geq \mathbb{E}_{\mathrm{s}\sim d}\big[V_{\boldsymbol{\pi}_{\mathrm{old}}}(s)\big] = J(\boldsymbol{\pi}_{\mathrm{old}}).$$

However, the conditions of the lemma require every agent to solve $|\mathcal{S}|$ instances of Inequality (12), which may be an intractable problem. We shall design a single optimisation objective whose solution satisfies those inequalities instead. Furthermore, to have a practical application to large-scale problems, such an objective should be estimatable via sampling. To handle these challenges, we introduce the following Algorithm Template 2 which generates a continuum of HAML algorithms.

Algorithm Template 2: Heterogeneous-Agent Mirror Learning

Initialise a joint policy $\boldsymbol{\pi}_0 = (\pi_0^1, \dots, \pi_0^n)$;
for $k = 0, 1, \dots$ **do**
  Compute the advantage function $A_{\boldsymbol{\pi}_k}(s, \boldsymbol{a})$ for all state-(joint)action pairs $(s, \boldsymbol{a})$;
  Draw a permutaion $i_{1:n}$ of agents at random //from a positive distribution $p \in \mathcal{P}(\mathrm{Sym}(n))$;
  for $m = 1 : n$ **do**
    Make an update $\pi_{k+1}^{i_m} = \underset{\pi^{i_m} \in \mathcal{U}_{\boldsymbol{\pi}_k}^{i_m}(\pi_k^{i_m})}{\arg\max} \mathbb{E}_{s \sim \beta_{\boldsymbol{\pi}_k}} \Big[\big[\mathcal{M}^{(\pi^{i_m})}_{\mathfrak{D}^{i_m}, \boldsymbol{\pi}_{k+1}^{i_{1:m-1}}} A_{\boldsymbol{\pi}_k} \big](s) \Big]$;

Output: A limit-point joint policy $\boldsymbol{\pi}_\infty$

Based on Lemma 13 and the fact that $\pi^i \in \mathcal{U}^i_{\boldsymbol{\pi}}(\pi^i), \forall i \in \mathcal{N}, \pi^i \in \Pi^i$, we can know any HAML algorithm (weakly) improves the joint return at every iteration. In practical settings, such as deep MARL, the maximisation step of a HAML method can be performed by a few steps of gradient ascent on a sample average of HAMO (see Definition 10). We also highlight that if the neighbourhood operators $\mathcal{U}^i$ can be chosen so that they produce small policy-space subsets, then the resulting updates will be not only improving but also small. This, again, is a desirable property while optimising neural-network policies, as it helps stabilise the algorithm. Similar to HATRL, the order of agents in HAML updates is randomised at every iteration; this condition has been necessary to establish convergence to NE, which is intuitively comprehensible: fixed-point joint policies of this randomised procedure assure that none of the agents is incentivised to make an update, namely reaching a NE. We provide the full list of the most fundamental HAML properties in Theorem 14 which shows that any method derived from Algorithm Template 2 solves the cooperative MARL problem.

Theorem 14 (The Fundamental Theorem of Heterogeneous-Agent Mirror Learning) *Let, for every agent $i \in \mathcal{N}$, $\mathfrak{D}^i$ be a HADF, $\mathcal{U}^i$ be a neighbourhood operator, and let the sampling distributions $\beta_{\boldsymbol{\pi}}$ depend continuously on $\boldsymbol{\pi}$. Let $\boldsymbol{\pi}_0 \in \boldsymbol{\Pi}$, and the sequence of joint policies $(\boldsymbol{\pi}_k)_{k=0}^\infty$ be obtained by a HAML algorithm induced by $\mathfrak{D}^i, \mathcal{U}^i, \forall i \in \mathcal{N}$, and $\beta_{\boldsymbol{\pi}}$. Then, the joint policies induced by the algorithm enjoy the following list of properties*

1. *Attain the monotonic improvement property,*
$$J(\boldsymbol{\pi}_{k+1}) \;\geq\; J(\boldsymbol{\pi}_k),$$
2. *Their value functions converge to a Nash value function V^{NE}*
$$\lim_{k \to \infty} V_{\boldsymbol{\pi}_k} = V^{NE},$$
3. *Their expected returns converge to a Nash return,*
$$\lim_{k \to \infty} J(\boldsymbol{\pi}_k) = J^{NE},$$
4. *Their ω-limit set consists of Nash equilibria.*

See the proof in Appendix E. With the above theorem, we can conclude that HAML provides a template for generating theoretically sound, stable, monotonically improving algorithms that enable agents to learn solving multi-agent cooperation tasks.

3.4 Casting HATRPO and HAPPO as HAML Instances

In this section, we show that HATRPO and HAPPO are in fact valid instances of HAML, which provides a more direct theoretical explanation for their excellent empirical performance.

We begin with the example of HATRPO, where agent i_m (the permutation $i_{1:n}$ is drawn from the uniform distribution) updates its policy so as to maximise (in $\bar{\pi}^{i_m}$)

$$\mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}_{\text{old}}},\mathbf{a}^{i_{1:m-1}}\sim\boldsymbol{\pi}_{\text{new}}^{i_{1:m-1}},\mathrm{a}^{i_m}\sim\bar{\pi}^{i_m}}\Big[A^{i_m}_{\boldsymbol{\pi}_{\text{old}}}(\mathrm{s},\mathbf{a}^{i_{1:m-1}},\mathrm{a}^{i_m})\Big],\quad \text{subject to } \overline{D}_{\mathrm{KL}}(\pi^{i_m}_{\text{old}},\bar{\pi}^{i_m})\leq\delta.$$

This optimisation objective can be casted as a HAMO with the HADF $\mathfrak{D}^{i_m}\equiv 0$, and the KL-divergence neighbourhood operator

$$\mathcal{U}^{i_m}_{\boldsymbol{\pi}}(\pi^{i_m})=\Big\{\bar{\pi}^{i_m}\;\Big|\;\mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}}}\Big[\mathrm{KL}\big(\pi^{i_m}(\cdot^{i_m}|\mathrm{s}),\bar{\pi}^{i_m}(\cdot^{i_m}|\mathrm{s})\big)\Big]\leq\delta\Big\}.$$

The sampling distribution used in HATRPO is $\beta_{\boldsymbol{\pi}}=\rho_{\boldsymbol{\pi}}$. Lastly, as the agents update their policies in a random loop, the algorithm is an instance of HAML. Hence, it is monotonically improving and converges to a Nash equilibrium set.

In HAPPO, the update rule of agent i_m is changed with respect to HATRPO as

$$\mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}_{\text{old}}},\mathbf{a}^{i_{1:m-1}}\sim\boldsymbol{\pi}_{\text{new}}^{i_{1:m-1}},\mathrm{a}^{i_m}\sim\pi^{i_m}_{\text{old}}}\Big[\min\big(\mathrm{r}(\bar{\pi}^{i_m})A^{i_{1:m}}_{\boldsymbol{\pi}_{\text{old}}}(\mathrm{s},\mathbf{a}^{i_{1:m}}),\mathrm{clip}\big(\mathrm{r}(\bar{\pi}^{i_m}),1\pm\epsilon\big)A^{i_{1:m}}_{\boldsymbol{\pi}_{\text{old}}}(\mathrm{s},\mathbf{a}^{i_{1:m}})\big)\Big],$$

where $\mathrm{r}(\bar{\pi}^i)=\frac{\bar{\pi}^i(\mathrm{a}^i|\mathrm{s})}{\pi^i_{\text{old}}(\mathrm{a}^i|\mathrm{s})}$. We show in Appendix F that this optimisation objective is equivalent to

$$\mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}_{\text{old}}}}\Big[\mathbb{E}_{\mathbf{a}^{i_{1:m-1}}\sim\boldsymbol{\pi}_{\text{new}}^{i_{1:m-1}},\mathrm{a}^{i_m}\sim\bar{\pi}^{i_m}}\big[A^{i_m}_{\boldsymbol{\pi}_{\text{old}}}(\mathrm{s},\mathbf{a}^{i_{1:m-1}},\mathrm{a}^{i_m})\big]$$
$$-\mathbb{E}_{\mathbf{a}^{i_{1:m-1}}\sim\boldsymbol{\pi}_{\text{new}}^{i_{1:m-1}},\mathrm{a}^{i_m}\sim\pi^{i_m}_{\text{old}}}\big[\mathrm{ReLU}\big(\big[\mathrm{r}(\bar{\pi}^{i_m})-\mathrm{clip}\big(\mathrm{r}(\bar{\pi}^{i_m}),1\pm\epsilon\big)\big]A^{i_{1:m}}_{\boldsymbol{\pi}_{\text{old}}}(s,\mathbf{a}^{i_{1:m}})\big)\big]\Big].$$

The purple term is clearly non-negative due to the presence of the ReLU function. Furthermore, for policies $\bar{\pi}^{i_m}$ sufficiently close to $\pi^{i_m}_{\text{old}}$, the clip operator does not activate, thus rendering $\mathrm{r}(\bar{\pi}^{i_m})$ unchanged. Therefore, the purple term is zero at and in a region around $\bar{\pi}^{i_m}=\pi^{i_m}_{\text{old}}$, which also implies that its Gâteaux derivatives are zero. Hence, it evaluates a HADF for agent i_m, thus making HAPPO a valid HAML instance.

Finally, we would like to highlight that these conclusions about HATRPO and HAPPO strengthen the results in Section 3.1 and 3.2. In addition to their origin in HATRL, we now show that their optimisation objectives directly enjoy favorable theoretical properties endowed by HAML framework. Both interpretations underpin their empirical performance.

3.5 More HAML Instances

In this subsection, we exemplify how HAML can be used for derivation of principled MARL algorithms, solely by constructing valid drift functional, neighborhood operator, and sampling distribution. Our goal is to verify the correctness of HAML theory and enrich the cooperative MARL with more theoretically guaranteed and practical algorithms. The results are more robust heterogeneous-agent versions of popular RL algorithms including A2C, DDPG, and TD3, different from those in Section 2.3.2.

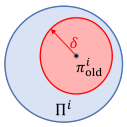
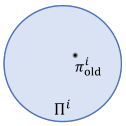
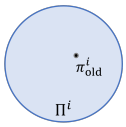
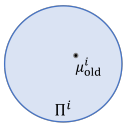
	Heterogeneous-Agent Mirror Learning			
	HATRPO	**HAPPO**	**HAA2C**	**HADDPG & HATD3**
Drift functional	$\mathfrak{D}^i \equiv 0$	$\mathfrak{D}^i = \mathbb{E}\left[\mathrm{ReLU}\left(\left[\mathrm{r}(\pi^i) - \mathrm{clip}\left(\mathrm{r}(\pi^i)\right)\right] A^{1:i}_{\boldsymbol{\pi}_{\text{old}}}(\mathrm{s}, \mathbf{a}^{1:i})\right)\right]$	$\mathfrak{D}^i \equiv 0$	$\mathfrak{D}^i \equiv 0$
Neighborhood operator	 Within a KL-ball $\mathfrak{B}_\delta$	 Entire policy space	 Entire policy space	 Entire policy space
Sampling distribution	Sample from *on-policy trajectory data*			Sample from *off-policy data buffer*

Figure 3: This figure presents a simplified schematic overview of HARL algorithms represented as valid instances of HAML. The complete details are available in Appendix H. By recasting HATRPO and HAPPO as HAML formulations, we demonstrate that their guarantees pertaining to monotonic improvement and NE convergence are enhanced by leveraging the HAML framework. Moreover, HAA2C, HADDPG, and HATD3 are obtained by designing HAML components, thereby securing those same performance guarantees. The variety of drift functionals, neighborhood operators, and sampling distributions utilised by these approaches further attests to the versatility and richness of the HAML framework.

3.5.1 HAA2C

HAA2C intends to optimise the policy for the joint advantage function at every iteration, and similar to A2C, does not impose any penalties or constraints on that procedure. This learning procedure is accomplished by, first, drawing a random permutation of agents $i_{1:n}$, and then performing a few steps of gradient ascent on the objective of

$$\mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}_{\text{old}}}, \mathbf{a}^{i_{1:m}}\sim\boldsymbol{\pi}^{i_{1:m}}_{\text{old}}}\left[\frac{\boldsymbol{\pi}^{i_{1:m-1}}_{\text{new}}(\mathbf{a}^{i_{1:m-1}}|\mathrm{s})\pi^{i_m}(\mathrm{a}^{i_m}|\mathrm{s})}{\boldsymbol{\pi}^{i_{1:m-1}}_{\text{old}}(\mathbf{a}^{i_{1:m-1}}|\mathrm{s})\pi^{i_m}_{\text{old}}(\mathrm{a}^{i_m}|\mathrm{s})} A^{i_m}_{\boldsymbol{\pi}_{\text{old}}}(\mathrm{s}, \mathbf{a}^{i_{1:m-1}}, \mathrm{a}^{i_m})\right], \tag{13}$$

with respect to π^{i_m} parameters, for each agent i_m in the permutation, sequentially. In practice, we replace the multi-agent advantage $A^{i_m}_{\boldsymbol{\pi}_{\text{old}}}(\mathrm{s}, \mathbf{a}^{i_{1:m-1}}, \mathrm{a}^{i_m})$ with the joint advantage estimate which, thanks to the joint importance sampling in Equation (13), poses the same objective on the agent (see Appendix G for full pseudocode).

3.5.2 HADDPG

HADDPG exploits the fact that $\beta_{\boldsymbol{\pi}}$ can be independent of $\boldsymbol{\pi}$ and aims to maximise the state-action value function *off-policy*. As it is a deterministic-action method, importance sampling in its case translates to replacement of the old action inputs to the critic with the new ones. Namely, agent i_m in a random permutation $i_{1:n}$ maximises

$$\mathbb{E}_{\mathrm{s}\sim\beta_{\boldsymbol{\mu}_{\text{old}}}}\left[Q^{i_{1:m}}_{\boldsymbol{\mu}_{\text{old}}}(\mathrm{s}, \boldsymbol{\mu}^{i_{1:m-1}}_{\text{new}}(\mathrm{s}), \mu^{i_m}(\mathrm{s}))\right], \tag{14}$$

with respect to μ^{i_m}, also with a few steps of gradient ascent. Similar to HAA2C, optimising the state-action value function (with the old action replacement) is equivalent to the original multi-agent value (see Appendix G for full pseudocode).

3.5.3 HATD3

HATD3 improves HADDPG with tricks proposed by Fujimoto et al. (2018). Similar to HADDPG, HATD3 is also an off-policy algorithm and optimises the same target, but it employs target policy smoothing, clipped double Q-learning, and delayed policy updates techniques (see Appendix G for full pseudocode). We observe that HATD3 consistently outperforms HADDPG on all tasks, showing that relevant reinforcement learning can be directly applied to MARL without the need for rediscovery, another benefit of the HAML.

As the HADDPG and HATD3 algorithms have been derived, it is logical to consider the possibility of HADQN, given that DQN can be viewed as a pure value-based version of DDPG for discrete action problems. In light of this, we introduce HAD3QN, a value-based approximation of HADDPG that incorporates techniques proposed by Van Hasselt et al. (2016) and Wang et al. (2016). The details of HAD3QN are presented in Appendix I, which includes the pseudocode, performance analysis, and an ablation study demonstrating the importance of the dueling double Q-network architecture for achieving stable and efficient multi-agent learning.

To elucidate the formulations and differences of HARL approaches in their HAML representation, we provide a simplified summary in Figure 3 and list the full details in Appendix H. While these approaches have already tailored HADFs, neighbourhood operators, and sampling distributions, we speculate that the entire abundance of the HAML framework can still be explored with more future work. Nevertheless, we commence addressing the heterogeneous-agent cooperation problem with these five methods, and analyse their performance in Section 5.

4. Related Work

There have been previous attempts that tried to solve the cooperative MARL problem by developing multi-agent trust region learning theories. Despite empirical successes, most of them did not manage to propose a theoretically-justified trust region protocol in multi-agent learning, or maintain the monotonic improvement property. Instead, they tend to impose certain assumptions to enable direct implementations of TRPO/PPO in MARL problems. For example, IPPO (de Witt et al., 2020) assumes homogeneity of action spaces for all agents and enforces parameter sharing. Yu et al. (2022) proposed MAPPO which enhances IPPO by considering a joint critic function and finer implementation techniques for on-policy methods. Yet, it still suffers similar drawbacks of IPPO due to the lack of monotonic improvement guarantee especially when the parameter-sharing condition is switched off. Wen et al. (2022) adjusted PPO for MARL by considering a game-theoretical approach at the meta-game level among agents. Unfortunately, it can only deal with two-agent cases due to the intractability of Nash equilibrium. Recently, Li and He (2023) tried to implement TRPO for MARL through distributed consensus optimisation; however, they enforced the same ratio $\bar{\pi}^i(a^i|s)/\pi^i(a^i|s)$ for all agents (see their Equation (7)), which, similar to parameter sharing, largely limits the policy space for optimisation. Moreover, their method comes with a δ/n KL-constraint threshold that fails to consider scenarios with large agent number. While Coordinated PPO (CoPPO) (Wu et al., 2021) derived a theoretically-grounded joint objective and obtained practical algorithms through a set of approximations, it still suffers from the non-stationarity problem as it updates agents simultaneously.

One of the key ideas behind our Heterogeneous-Agent algorithm series is the sequential update scheme. A similar idea of multi-agent sequential update was also discussed in the context of dynamic programming (Bertsekas, 2019) where artificial "in-between" states have to be considered. On the contrary, our sequential update scheme is developed based on Lemma 4, which does not require any artificial assumptions and holds for any cooperative games. The idea of sequential update also appeared in principal component analysis; in EigenGame (Gemp et al., 2021) eigenvectors, represented as players, maximise their own utility functions one-by-one. Although EigenGame provably solves the PCA problem, it is of little use in MARL, where a single iteration of sequential updates is insufficient to learn complex policies. Furthermore, its design and analysis rely on closed-form matrix calculus, which has no extension to MARL.

Lastly, we would like to highlight the importance of the decomposition result in Lemma 4. This result could serve as an effective solution to value-based methods in MARL where tremendous efforts have been made to decompose the joint Q-function into individual Q-functions when the joint Q-function is decomposable (Rashid et al., 2018). Lemma 4, in contrast, is a general result that holds for any cooperative MARL problems regardless of decomposability. As such, we think of it as an appealing contribution to future developments on value-based MARL methods.

Our work is an extension of previous work HATRPO / HAPPO, which was originally proposed in a conference paper (Kuba et al., 2022a). The main additions in our work are:

- Introducing Heterogeneous-Agent Mirror Learning (HAML), a more general theoretical framework that strengthens theoretical guarantees for HATRPO and HAPPO and can induce a continuum of sound algorithms with guarantees of monotonic improvement and convergence to Nash Equilibrium;

- Designing novel algorithm instances of HAML including HAA2C, HADDPG, and HATD3, which attain better performance than their existing MA-counterparts, with HATD3 establishing the new SOTA results for off-policy algorithms;

- Releasing PyTorch-based implementation of HARL algorithms, which is more unified, modularised, user-friendly, extensible, and effective than the previous one;

- Conducting comprehensive experiments evaluating HARL algorithms on six challenging benchmarks Multi-Agent Particle Environment (MPE), Multi-Agent MuJoCo (MA-MuJoCo), StarCraft Multi-Agent Challenge (SMAC), SMACv2, Google Research Football Environment (GRF), and Bi-DexterousHands.

5. Experiments and Analysis

In this section, we evaluate and analyse HARL algorithms on six cooperative multi-agent benchmarks — Multi-Agent Particle Environment (MPE) (Lowe et al., 2017; Mordatch and Abbeel, 2018), Multi-Agent MuJoCo (MAMuJoCo) (Peng et al., 2021), StarCraft Multi-Agent Challenge (SMAC) (Samvelyan et al., 2019), SMACv2 (Ellis et al., 2022), Google Research Football Environment (GRF) (Kurach et al., 2020), and Bi-DexterousHands (Chen et al., 2022), as shown in Figure 4 — and compare their performance to existing SOTA

(a) MPE (b) MAMuJoCo (c) SMAC (d) SMACv2 (e) GRF (f) Bi-DexterousHands

Figure 4: The six environments used for testing HARL algorithms.

methods. These benchmarks are diverse in task difficulty, agent number, action type, dimensionality of observation space and action space, and cooperation strategy required, and hence provide a comprehensive assessment of the effectiveness, stability, robustness, and generality of our methods. The experimental results demonstrate that HAPPO, HADDPG, and HATD3 generally outperform their MA-counterparts on heterogeneous-agent cooperation tasks. Moreover, HARL algorithms culminate in HAPPO and HATD3, which exhibit superior effectiveness and stability for heterogeneous-agent cooperation tasks over existing strong baselines such as MAPPO, QMIX, MADDPG, and MATD3, refreshing the state-of-the-art results. Our ablation study also reveals that the novel details introduced by HATRL and HAML theories, namely non-sharing of parameters and randomised order in sequential update, are crucial for obtaining the strong performance. Finally, we empirically show that the computational overhead introduced by sequential update does not need to be a concern.

Our implementation of HARL algorithms takes advantage of the sequential update scheme and the CTDE framework that HARL algorithms share in common, and unifies them into either the on-policy or the off-policy training pipeline, resulting in modularisation and extensibility. It also naturally hosts MAPPO, MADDPG, and MATD3 as special cases and provides the (re)implementation of these three algorithms along with HARL algorithms. For fair comparisons, we use our (re)implementation of MAPPO, MADDPG, and MATD3 as baselines on MPE and MAMuJoCo, where their publicly acknowledged performance report under exactly the same settings is lacking, and we ensure that their performance matches or exceeds the results reported by their original paper and subsequent papers; on the other benchmarks, the original implementations of baselines are used. To be consistent with the officially reported results of MAPPO, we let it utilize parameter sharing on all but Bi-DexterousHands and the Speaker Listener task in MPE. Details of hyper-parameters and experiment setups can be found in Appendix K.

5.1 MPE Testbed

We consider the three fully cooperative tasks in MPE (Lowe et al., 2017): Spread, Reference, and Speaker Listener. These tasks require agents to explore and then learn the optimal cooperation strategies, such as spreading to targets as quickly as possible without collision, instructing companions, and so on. The Speaker Listener scenario, in particular, explicitly designs different roles and fails the homogeneous agent approach. As the original codebase of MPE is no longer maintained, we choose to use its PettingZoo version (Terry et al., 2021). To make it compatible with the cooperative MARL problem formulation in Section 2, we implement the interface of MPE so that agents do not have access to their individual rewards, as opposed to the setting used by MADDPG. Instead, individual rewards of agents are summed up to form the joint reward, which is available during centralised training. We

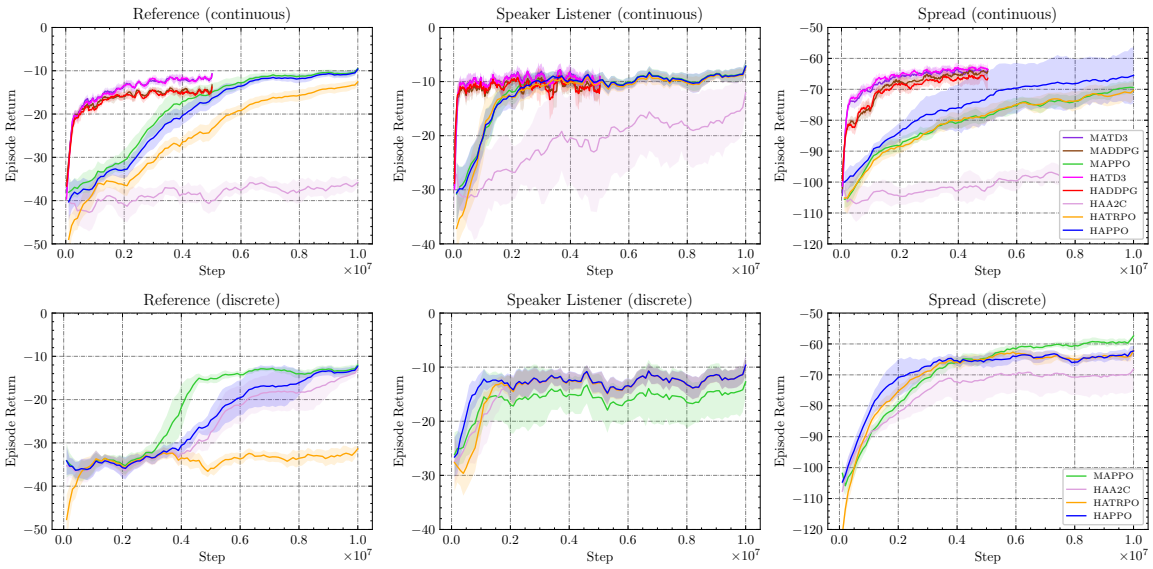


Figure 5: Comparisons of average episode return on Multi-Agent Particle Environments. The "continuous" and "discrete" in parenthesis refer to the type of action space in each task.

evaluate HAPPO, HATRPO, HAA2C, HADDPG, and HATD3 on the continuous action-space version of these three tasks against MAPPO, MADDPG, and MATD3, with on-policy algorithms running for 10 million steps and off-policy ones for 5 million steps. Since the stochastic policy algorithms, namely HAPPO, HATRPO, and HAA2C, can also be applied to discrete action-space scenarios, we additionally compare them with MAPPO on the discrete version of these three tasks, using the same number of timesteps. The learning curves plotted from training data across three random seeds are shown in Figure 5.

While MPE tasks are relatively simple, it is sufficient for identifying several patterns. HAPPO consistently solves all six combinations of tasks, with its performance comparable to or better than MAPPO. With a single set of hyper-parameters, HATRPO also solves five combinations easily and achieves steady learning curves due to the explicitly specified distance constraint and reward improvement between policy updates. It should be noted that the oscillations observed after convergence are due to the randomness of test environments which affects the maximum reward an algorithm can attain. HAA2C, on the other hand, is equally competitive on the discrete version of tasks, but shows higher variance and is empirically harder to achieve the same level of episode return on the continuous versions, which is a limitation of this method since its update rule can not be precisely realised in practice and meanwhile it imposes no constraint. Nevertheless, it still constitutes a potentially competitive solution.

Furthermore, two off-policy HARL methods, HADDPG and HATD3, exhibit extremely fast mastery of the three tasks with small variance, demonstrating their advantage in high sample efficiency. Their performances are similar to MA-counterparts on these simple tasks, with TD3-based methods achieving faster convergence rate and higher total rewards, establishing new SOTA off-policy results. Off-policy HARL methods consistently converge with much fewer samples than on-policy methods across all tasks, holding the potential to

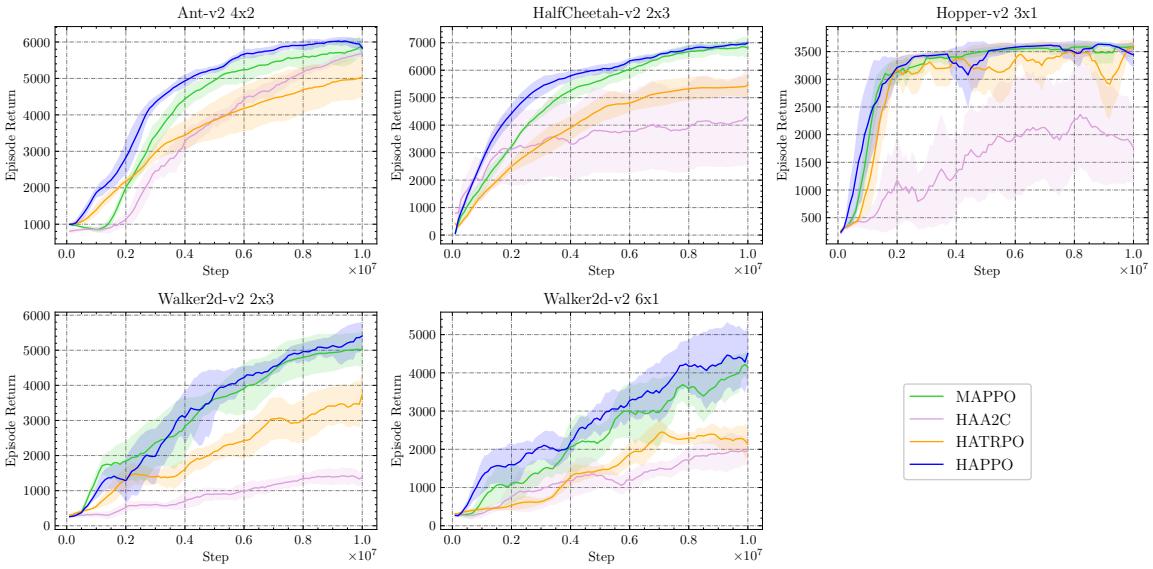


Figure 6: Comparisons of average episode return of on-policy algorithms on Multi-Agent MuJoCo. HAPPO generally outperforms MAPPO, refreshing the state-of-the-art (SOTA) results for on-policy algorithms.

alleviate the high sample complexity and slow training speed problems, which are commonly observed in MARL experiments.

These observations show that while HARL algorithms have the same improvement and convergence guarantees in theory, they differ in learning behaviours due to diverse algorithmic designs. In general, they complement each other and collectively solve all tasks.

5.2 MAMuJoCo Testbed

The Multi-Agent MuJoCo (MAMuJoCo) environment is a multi-agent extension of MuJoCo. While the MuJoCo tasks challenge a robot to learn an optimal way of motion, MAMuJoCo models each part of a robot as an independent agent — for example, a leg for a spider or an arm for a swimmer — and requires the agents to collectively perform efficient motion. With the increasing variety of the body parts, modeling heterogeneous policies becomes **necessary**. Thus, we believe that MAMuJoCo is a suitable task suite for evaluating the effectiveness of our heterogeneous-agent methods. We evaluate HAPPO, HATRPO, HAA2C, HADDPG, and HATD3 on the five most representative tasks against MAPPO, MADDPG, and MATD3 and plot the learning curves across at least three seeds in Figure 6 and 7.

We observe that on all five tasks, HAPPO, HADDPG, and HATD3 achieves generally better average episode return than their MA-counterparts. HATRPO and HAA2C also achieve strong and steady learning behaviours on most tasks. Since the running motion are hard to be realised by any subset of all agents, the episode return metric measures the quality of agents' cooperation. Rendered videos from the trained models of HARL algorithms confirm that agents develop effective cooperation strategies for controlling their corresponding body parts. For example, on the 2-agent HalfCheetah task, agents trained by

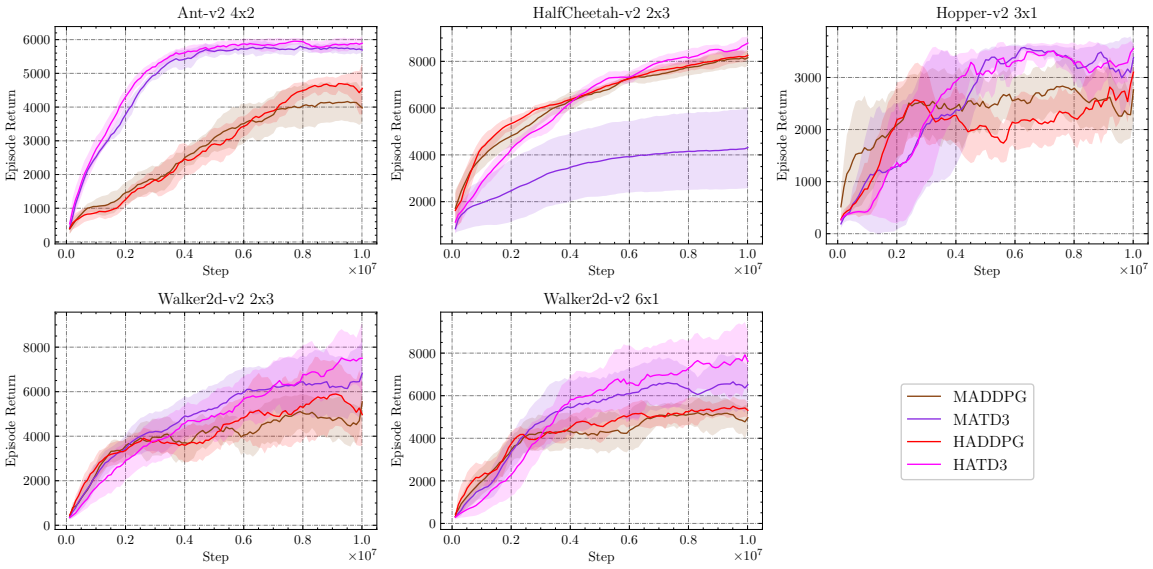


Figure 7: Comparisons of average episode return of off-policy algorithms on Multi-Agent MuJoCo. HADDPG and HATD3 generally outperform MADDPG and MATD3, while HATD3 achieves the highest average return across all tasks, thereby refreshing the state-of-the-art (SOTA) results for off-policy algorithms.

HAPPO learn to alternately hit the ground, forming a swift kinematic gait that resembles a real cheetah. The motion performed by each agent is meaningless alone and only takes effect when combined with the other agent's actions. In other words, all agents play indispensable roles and have unique contributions in completing the task, which is the most desirable form of cooperation. Empirically, HARL algorithms prove their capability to generate this level of cooperation from random initialisation.

As for the off-policy HARL algorithms, HATD3 outperforms both MATD3 and HADDPG on all tasks, due to the beneficial combination of sequential update and the stabilising effects brought by twin critics, delayed actor update, and target action smoothing tricks. This also admits the feasibility of introducing RL tricks to MARL. Its performance is even generally better than HAPPO, showing the competence to handle continuous tasks. Experimental results on MAMuJoCo not only prove the superiority of HARL algorithms over existing strong baselines, but also reveal that HARL renders multiple effective solutions to multi-agent cooperation tasks.

Though MAMuJoCo tasks are heterogeneous in nature, parameter sharing is still effective in scenarios where learning a "versatile" policy to control all body parts by relying on the expressiveness of neural network is enough. As a result, on these five tasks, MAPPO underperforms HAPPO by not very large margins. To fully distinguish HAPPO from MAPPO, we additionally compare them on the 17-agent Humanoid task and report the learning curves averaged across three seeds in Figure 8. In this scenario, the 17 agents control dissimilar body parts and it is harder for a single policy to select the right action for each part. Indeed, MAPPO completely fails to learn. In contrast, HAPPO still manages to coordinate the agents' updates with its sequential update scheme which leads to a walking humanoid with

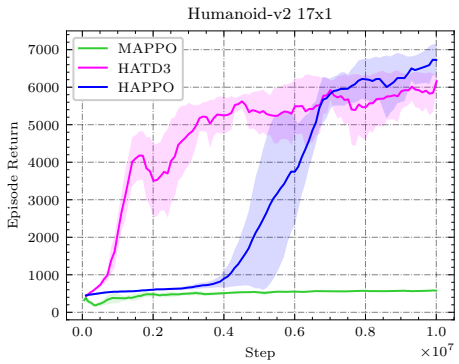


Figure 8: Comparisons of average episode return on the 17-agent Humanoid control task in Multi-Agent MuJoCo. In the face of this many-heterogeneous-agent task, HAPPO and HATD3 achieve state-of-the-art (SOTA) performance, while MAPPO fails completely. This highlights the superior effectiveness of HARL algorithms for promoting cooperation among heterogeneous agents.

the joint effort from all agents. With the same theoretical properties granted by HAML, HATD3 also successfully learns to control the 17-agent humanoid. Therefore, HARL algorithms are more applicable and effective for the general many-heterogeneous-agent cases. Their advantage becomes increasingly significant with the increasing heterogeneity of agents.

5.3 SMAC & SMACv2 Testbed

The StarCraft Multi-Agent Challenge (SMAC) contains a set of StarCraft maps in which a team of mostly homogeneous ally units aims to defeat the opponent team. It challenges an algorithm to develop effective teamwork and decentralised unit micromanagement, and serves as a common arena for algorithm comparison. We benchmark HAPPO and HATRPO on five hard maps and five super hard maps in SMAC against QMIX (Rashid et al., 2018) and MAPPO (Yu et al., 2022), which are known to achieve supreme results. Furthermore, as Ellis et al. (2022) proposes SMACv2 to increase randomness of tasks and diversity among unit types in SMAC, we additionally test HAPPO and HATRPO on five maps in SMACv2 against QMIX and MAPPO. On these two sets of tasks, we adopt the implementations of QMIX and MAPPO that have achieved the best-reported results, *i.e.* in SMAC we use the implementation by Yu et al. (2022) and in SMACv2 we use the implementation by Ellis et al. (2022). Following the evaluation metric proposed by Wang et al. (2021), we report the win rates computed across at least three seeds in Table 1 and provide the learning curves in Appendix J.

We observe that HAPPO and HATRPO are able to achieve comparable or superior performance to QMIX and MAPPO across five hard maps and five super hard maps in SMAC, while not relying on the restrictive parameter-sharing trick, as opposed to MAPPO. From the learning curves, it shows that HAPPO and HATRPO exhibit steadily improving learning behaviours, while baselines experience large oscillations on `25m` and `27m_vs_30m`, again demonstrating the monotonic improvement property of our methods. On SMACv2, though randomness and heterogeneity increase, HAPPO and HATRPO robustly achieve competi-

Map	Difficulty	HAPPO	HATRPO	MAPPO	QMIX	Steps
8m_vs_9m	Hard	83.8(4.1)	**92.5**(3.7)	87.5(4.0)	**92.2**(1.0)	1e7
25m	Hard	95.0(2.0)	**100.0**(0.0)	**100.0**(0.0)	89.1(3.8)	1e7
5m_vs_6m	Hard	**77.5**(7.2)	**75.0**(6.5)	**75.0**(18.2)	**77.3**(3.3)	1e7
3s5z	Hard	**97.5**(1.2)	93.8(1.2)	**96.9**(0.7)	89.8(2.5)	1e7
10m_vs_11m	Hard	87.5(6.7)	**98.8**(0.6)	96.9(4.8)	95.3(2.2)	1e7
MMM2	Super Hard	88.8(2.0)	**97.5**(6.4)	**93.8**(4.7)	87.5(2.5)	2e7
3s5z_vs_3s6z	Super Hard	66.2(3.1)	72.5(14.7)	70.0(10.7)	**87.5**(12.6)	2e7
27m_vs_30m	Super Hard	76.6(1.3)	**93.8**(2.1)	80.0(6.2)	45.3(14.0)	2e7
corridor	Super Hard	92.5(13.9)	88.8(2.7)	**97.5**(1.2)	82.8(4.4)	2e7
6h_vs_8z	Super Hard	**76.2**(3.1)	**78.8**(0.6)	**85.0**(2.0)	**92.2**(26.2)	4e7
protoss_5_vs_5	-	57.5(1.2)	50.0(2.4)	56.2(3.2)	**65.6**(3.9)	1e7
terran_5_vs_5	-	57.5(1.3)	56.8(2.9)	53.1(2.7)	**62.5**(3.8)	1e7
zerg_5_vs_5	-	42.5(2.5)	**43.8**(1.2)	40.6(7.0)	34.4(2.2)	1e7
zerg_10_vs_10	-	28.4(2.2)	34.6(0.2)	37.5(3.2)	**40.6**(3.4)	1e7
zerg_10_vs_11	-	16.2(0.6)	19.3(2.1)	**29.7**(3.8)	25.0(3.9)	1e7

Table 1: Median evaluation win rate and standard deviation on ten SMAC maps (upper in the table) and five SMACv2 maps (lower in the table) for different methods. All values within 1 standard deviation of the maximum win rate are marked in bold. The column labeled "Steps" specifies the number of steps used for training. Our results suggest that HAPPO and HATRPO perform comparably or better than MAPPO and QMIX on these tasks, which mainly involve homogeneous agents. Moreover, HAPPO and HATRPO do not rely on the restrictive parameter-sharing technique, demonstrating their versatility in various scenarios.

tive win rates and are comparable to QMIX and MAPPO. Another important observation is that HATRPO is more effective than HAPPO in SMAC and SMACv2, outperforming HAPPO on 10 out of 15 tasks. This implies that HATRPO could enhance learning stability by imposing explicit constraints on update distance and reward improvement, making it a promising approach to tackling novel and challenging tasks. Overall, the performance of HAPPO and HATRPO in SMAC and SMACv2 confirms their capability to coordinate agents' training in largely homogeneous settings.

5.4 Google Research Football Testbed

Google Research Football Environment (GRF) composes a series of tasks where agents are trained to play football in an advanced, physics-based 3D simulator. From literature (Yu et al., 2022), it is shown that GRF is still challenging to existing methods. We apply HAPPO to the five academy tasks of GRF, namely 3 vs 1 with keeper (3v.1), counterattack (CA) easy and hard, pass and shoot with keeper (PS), and run pass and shoot with keeper (RPS), with MAPPO and QMIX as baselines. As GRF does not provide a global state interface, our solution is to implement a global state based on agents' observations following the `Simple115StateWrapper` of GRF. Concretely, the global state consists of common components in agents' observations and the concatenation of agent-specific parts, and is taken as

scenarios	HAPPO	MAPPO	QMIX
PS	**96.93**(1.11)	94.92(0.85)	8.05(5.58)
RPS	**77.30**(7.40)	**76.83**(3.57)	8.08(3.29)
3v.1	**94.74**(3.05)	88.03(4.15)	8.12(4.46)
CA(easy)	**92.00**(1.62)	87.76(6.40)	15.98(11.77)
CA(hard)	**88.14**(5.77)	77.38(10.95)	3.22(4.39)

Table 2: Average evaluation score rate and standard deviation (over six seeds) on GRF scenarios for different methods. All values within 1 standard deviation of the maximum score rate are marked in bold. Our results reveal that HAPPO generally outperforms MAPPO and QMIX on all tasks, setting a new state-of-the-art performance benchmark.

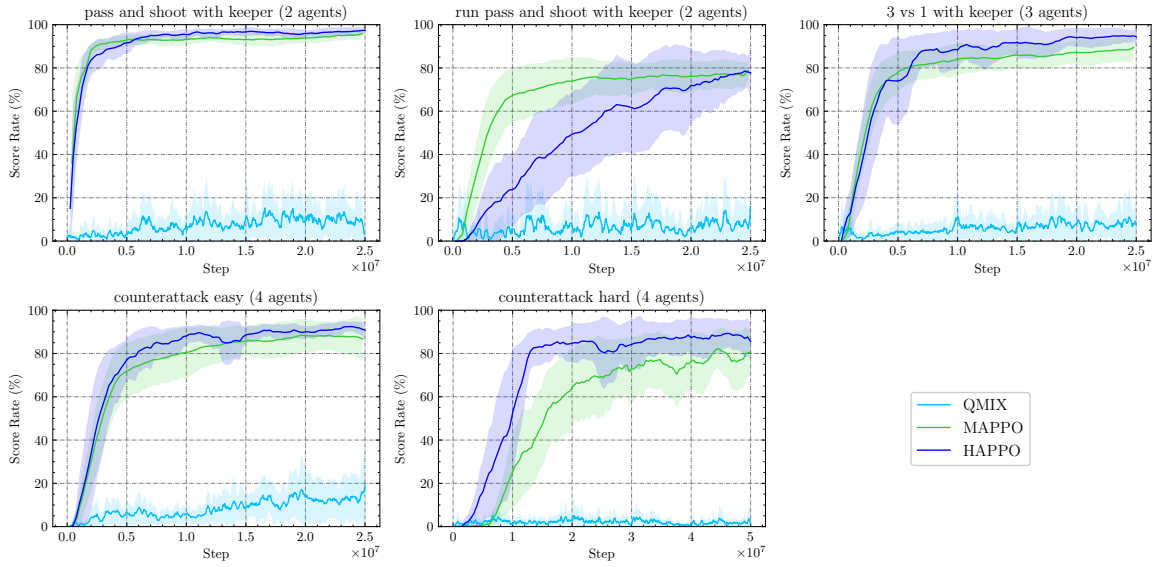


Figure 9: The figure displays the average score rate comparisons for different methods on GRF, and also illustrates how the performance gaps between HAPPO and MAPPO widen as the roles and difficulty levels of tasks increase. Overall, our results demonstrate that HAPPO outperforms MAPPO in tackling complex multi-agent scenarios.

input by the centralised critic for value prediction. We also utilize the dense-reward setting in GRF. All methods are trained for 25 million environment steps in all scenarios with the exception of CA (hard), in which methods are trained for 50 million environment steps. We compute the success rate over 100 rollouts of the game and report the average success rate over the last 10 evaluations across 6 seeds in Table 2. We also report the learning curves of the algorithms in Figure 9.

We observe that HAPPO is generally better than MAPPO, establishing new state-of-the-art results, and they both significantly outperform QMIX. In particular, as the number of agents increases and the roles they play become more diverse, the performance gap between HAPPO and MAPPO becomes larger, again showing the effectiveness and advantage of HARL algorithms for the many-heterogeneous-agent settings. From the rendered videos, it

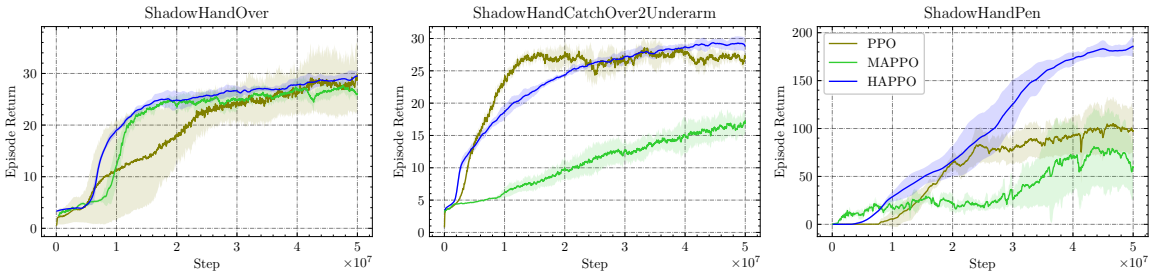


Figure 10: Comparisons of average episode return on Bi-DexterousHands. The learning curves demonstrate that HAPPO consistently achieves the highest return, outperforming both MAPPO and PPO.

is shown that agents trained by HAPPO develop clever teamwork strategies for ensuring a high score rate, such as cooperative breakthroughs to form one-on-one chances, etc. This result further supports the effectiveness of applying HAPPO to cooperative MARL problems.

5.5 Bi-DexterousHands Testbed

Based on IsaacGym, Bi-DexterousHands provides a suite of tasks for learning human-level bimanual dexterous manipulation. It leverages GPU parallelisation and enables simultaneous instantiation of thousands of environments. Compared with other CPU-based environments, Bi-DexterousHands significantly increases the number of samples generated in the same time interval, thus alleviating the sample efficiency problem of on-policy algorithms. We choose three representative tasks and compare HAPPO with MAPPO as well as PPO. As the existing reported results of MAPPO on these tasks do not utilize parameter sharing, we follow them in order to be consistent. The learning curves plotted from training data across three random seeds are shown in Figure 10. On all three tasks, HAPPO consistently outperforms MAPPO, and is at least comparable to or better than the single-agent baseline PPO, while also showing less variance. The comparison between HAPPO and MAPPO demonstrates the superior competence of the sequential update scheme adopted by HARL algorithms over simultaneous updates for coordinating multiple heterogeneous agents.

5.6 Ablation Experiments

In this subsection, we conduct ablation study to investigate the importance of two key novelties that our HARL algorithms introduced; they are heterogeneity of agents' parameters and the randomisation of order of agents in the sequential update scheme. We compare the performance of original HAPPO with a version that shares parameters, and with a version where the order in sequential update scheme is fixed throughout training. We run the experiments on two MAMuJoCo tasks, namely 2-agent Walker and 6-agent Walker.

The experiments reveal that the deviation from the theory harms performance. In particular, parameter sharing introduces unreasonable policy constraints to training, harms the monotonic improvement property (Theorem 7 assumes heterogeneity), and causes HAPPO to converge to suboptimal policies. The suboptimality is more severe in the task with more diverse agents, as discussed in Section 2.3.1. Similarly, fixed order in the sequential up-

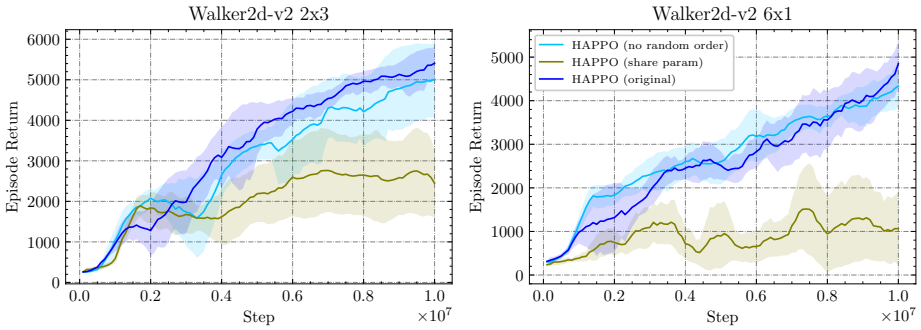


Figure 11: Performance comparison between original HAPPO, and its modified versions: HAPPO with parameter sharing, and HAPPO without randomisation of the sequential update scheme.

date scheme negatively affects the performance at convergence, as suggested by Theorem 8. In the 2-agent task, fixing update order leads to inferior performance throughout the training process; in the 6-agent task, while the fixed order version initially learns faster, it is gradually overtaken by the randomised order version and achieves worse convergence results. We conclude that the fine performance of HARL algorithms relies strongly on the close connection between theory and implementation.

5.7 Analysis of Computational Overhead

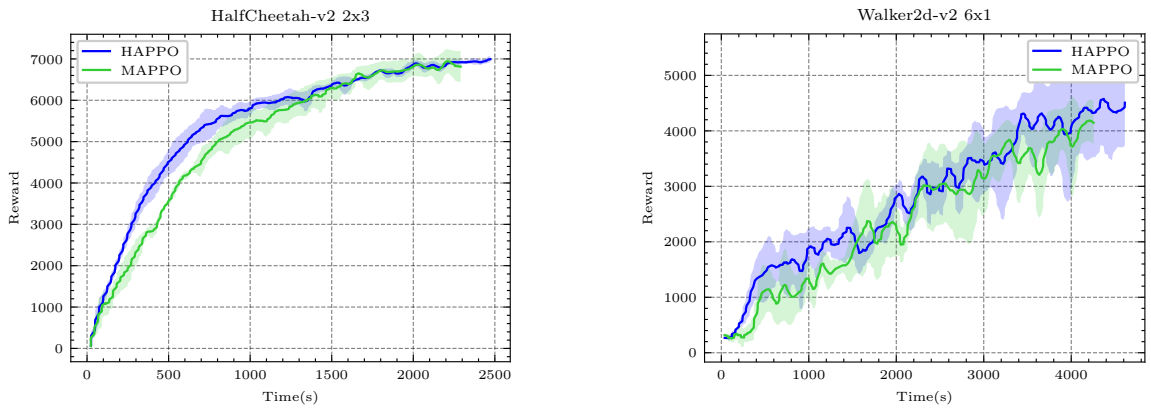
We then analyse the computational overhead introduced by the sequential update scheme. We mainly compare HAPPO with MAPPO in parameter-sharing setting, where our implementation conducts the single vectorized update [3]. We conduct experiments on seven MAMuJoCo tasks with all hyperparameters fixed. Both methods are trained for 1 million steps and we record the computational performance in Table 3. The machine for experiments in this subsection is equipped with an AMD Ryzen 9 5950X 16-Core Processor and an NVIDIA RTX 3090 Ti GPU, and we ensure that no other experiments are running.

We generally observe a linear relationship between update times for both HAPPO and MAPPO and the number of agents. For HAPPO, each agent is trained on a constant-sized batch input, denoted as $|B|$, across tasks. Thus the total time consumed to update all agents correlates directly with the agent count. For MAPPO, on the other hand, the shared parameter is trained on a batch input of size $n \times |B|$ when the number of agents is n. However, as the batch size $|B|$ used in MAMuJoCo is typically large, in this case 4000, vectorizing agents data does not significantly enhance GPU parallelization. This is evidenced by the relatively consistent FLOPS recorded across tasks. As a result, the MAPPO update timeframe also exhibits linear growth with increasing agents. The ratio of HAPPO and MAPPO update time is almost constant on the first six tasks and it nearly degenerates to 1 when both of them sufficiently utilize the computational resources, as shown in the case of 17-agent Humanoid where the significantly higher-dimensional observation space

3. Corresponding to the original implementation at `https://github.com/marlbenchmark/on-policy/blob/0affe7f4b812ed25e280af8115f279fbffe45bbe/onpolicy/algorithms/r_mappo/r_mappo.py#L205`.

scenarios	HAPPO			MAPPO		
	experiment time(s)	agents update time(s)	FLOPS	experiment time(s)	share param update time(s)	FLOPS
HalfCheetah 2x3	$203.4_{0.4}$	$8.6_{0.0}$	368	$197.6_{1.0}$	$4.9_{0.1}$	588
HalfCheetah 3x2	$264.7_{1.0}$	$12.9_{0.0}$	368	$256.0_{1.1}$	$6.2_{0.0}$	644
HalfCheetah 6x1	$451.0_{1.4}$	$25.3_{0.0}$	366	$441.4_{0.9}$	$12.3_{0.0}$	717
Walker 2x3	$193.9_{2.8}$	$8.6_{0.0}$	368	$187.4_{4.8}$	$4.9_{0.0}$	588
Walker 3x2	$245.0_{6.2}$	$12.9_{0.2}$	368	$232.6_{1.5}$	$6.3_{0.0}$	649
Walker 6x1	$408.6_{9.9}$	$25.4_{0.2}$	370	$383.2_{5.6}$	$12.2_{0.0}$	711
Humanoid 17x1	$912.0_{11.8}$	$76.7_{0.9}$	568	$988.0_{2.1}$	$71.3_{0.1}$	738

Table 3: Computational performance comparisons between HAPPO and MAPPO on seven MAMuJoCo tasks across three seeds. As for the comparison items, "experiment time" denotes the overall running time of a single experiment; "agents update time" of HAPPO denotes the total time of all agent updates; "share param update time" of MAPPO denotes the total time consumed in updating the shared parameters; "FLOPS" (floating-point operations per second) during the update is calculated as the total floating-point operations in a network forward pass divided by data transfer time plus computation time (unit: GFLOPS). The main figure represents the mean and the subscript represents the standard deviation. These figures suggest that the sequential update scheme does not introduce much computational burden compared to a single vectorized update.



(a) Return vs. time on 2-agent HalfCheetah.

(b) Return vs. time on 6-agent Walker.

Figure 12: Performance comparison between HAPPO and MAPPO with the x-axis being the wall-time. At the same time, HAPPO generally outperforms parameter-sharing MAPPO.

leads to increased GPU utilization, *i.e.* FLOPS, for HAPPO. These facts suggest that the sequential update scheme does not introduce much computational burden compared to the single vectorized update. As the update only constitutes a small portion of the whole experiment, such an additional computational overhead is almost negligible.

In Figure 12, we further provide the learning curves of HAPPO and MAPPO on two MAMuJoCo tasks corresponding to Figure 6, with the x-axis being wall-time. The oscillation observed in Figure 12(b) is due to a slight difference in training time across the seeds rather than the instability of algorithms. These figures demonstrate that HAPPO generally outperforms MAPPO at the same wall-time. To run 10 million steps, HAPPO needs 8.12% and 8.64% more time than MAPPO respectively, an acceptable tradeoff to enjoy the benefits of the sequential update scheme in terms of improved performance and rigorous theoretical guarantees. Thus, we justify that computational overhead does not need to be a concern.

6. Conclusion

In this paper, we present Heterogeneous-Agent Reinforcement Learning (HARL) algorithm series, a set of powerful solutions to cooperative multi-agent problems with theoretical guarantees of monotonic improvement and convergence to Nash Equilibrium. Based on the multi-agent advantage decomposition lemma and the sequential update scheme, we successfully develop Heterogeneous-Agent Trust Region Learning (HATRL) and introduce two practical algorithms — HATRPO and HAPPO — by tractable approximations. We further discover the Heterogeneous-Agent Mirror Learning (HAML) framework, which strengthens validations for HATRPO and HAPPO and is a general template for designing provably correct MARL algorithms whose properties are rigorously profiled. Its consequences are the derivation of more HARL algorithms, HAA2C, HADDPG, and HATD3, which significantly enrich the tools for solving cooperative MARL problems. Experimental analysis on MPE, MAMuJoCo, SMAC, SMACv2, GRF, and Bi-DexterousHands confirms that HARL algorithms generally outperform existing MA-counterparts and refresh SOTA results on heterogeneous-agent benchmarks, showing their superior effectiveness for heterogeneous-agent cooperation over strong baselines such as MAPPO and QMIX. Ablation studies further substantiate the key novelties required in theoretical reasoning and enhance the connection between HARL theory and implementation. For future work, we plan to consider more possibilities of the HAML framework and validate the effectiveness of HARL algorithms on real-world multi-robot cooperation tasks.

Acknowledgments

We would like to thank Chengdong Ma for insightful discussions; the authors of MAPPO (Yu et al., 2022) for providing original training data of MAPPO and QMIX on SMAC and GRF; the authors of SMACv2 (Ellis et al., 2022) for providing original training data of MAPPO and QMIX on SMACv2; and the authors of Bi-DexterousHands (Chen et al., 2022) for providing original training data of MAPPO and PPO on Bi-DexterousHands.

This project is funded by National Key R&D Program of China (2022ZD0114900) , Collective Intelligence & Collaboration Laboratory (QXZ23014101) , CCF-Tencent Open

Research Fund (RAGR20220109) , Young Elite Scientists Sponsorship Program by CAST (2022QNRC002), Beijing Municipal Science & Technology Commission (Z221100003422004).

Appendix A. Proofs of Example 2 and 1

Example 2 *Consider a fully-cooperative game with an even number of agents n, one state, and the joint action space $\{0,1\}^n$, where the reward is given by $r(\mathbf{0}^{n/2},\mathbf{1}^{n/2}) = r(\mathbf{1}^{n/2},\mathbf{0}^{n/2}) = 1$, and $r(\boldsymbol{a}^{1:n}) = 0$ for all other joint actions. Let J^* be the optimal joint reward, and J^*_{share} be the optimal joint reward under the shared policy constraint. Then*

$$\frac{J^*_{share}}{J^*} = \frac{2}{2^n}.$$

Proof Clearly $J^* = 1$. An optimal joint policy in this case is, for example, the deterministic policy with joint action $(\mathbf{0}^{n/2},\mathbf{1}^{n/2})$.

Now, let the shared policy be $(\theta, 1-\theta)$, where θ determines the probability that an agent takes action 0. Then, the expected reward is

$$J(\theta) = \Pr\left(\boldsymbol{a}^{1:n} = (\mathbf{0}^{n/2},\mathbf{1}^{n/2})\right)\cdot 1 + \Pr\left(\boldsymbol{a}^{1:n} = (\mathbf{1}^{n/2},\mathbf{0}^{n/2})\right)\cdot 1 = 2\cdot\theta^{n/2}(1-\theta)^{n/2}.$$

In order to maximise $J(\theta)$, we must maximise $\theta(1-\theta)$, or equivalently, $\sqrt{\theta(1-\theta)}$. By the arithmetic-geometric means inequality, we have

$$\sqrt{\theta(1-\theta)} \leq \frac{\theta + (1-\theta)}{2} = \frac{1}{2},$$

where the equality holds if and only if $\theta = 1-\theta$, that is $\theta = \frac{1}{2}$. In such case we have

$$J^*_{\text{share}} = J\left(\frac{1}{2}\right) = 2\cdot 2^{-n/2}\cdot 2^{-n/2} = \frac{2}{2^n},$$

which finishes the proof. ■

Example 1 *Let's consider a fully-cooperative game with 2 agents, one state, and the joint action space $\{0,1\}^2$, where the reward is given by $r(0,0) = 0, r(0,1) = r(1,0) = 2$, and $r(1,1) = -1$. Suppose that $\pi^i_{old}(0) > 0.6$ for $i = 1,2$. Then, if agents i update their policies by*

$$\pi^i_{new} = \arg\max_{\pi^i} \mathbb{E}_{\mathrm{a}^i\sim\pi^i,\mathrm{a}^{-i}\sim\pi^{-i}_{old}}\left[A_{\boldsymbol{\pi}_{old}}(\mathrm{a}^i,\mathrm{a}^{-i})\right], \forall i \in \mathcal{N},$$

then the resulting policy will yield a lower return,

$$J(\boldsymbol{\pi}_{old}) > J(\boldsymbol{\pi}_{new}) = \min_{\boldsymbol{\pi}} J(\boldsymbol{\pi}).$$

Proof As there is only one state, we can ignore the infinite horizon and the discount factor γ, thus making the state-action value and the reward functions equivalent, $Q \equiv r$.

Let us, for brevity, define $\pi^i = \pi^i_{\text{old}}(0) > 0.6$, for $i = 1, 2$. We have

$$J(\boldsymbol{\pi}_{\text{old}}) = \Pr(\mathrm{a}^1 = \mathrm{a}^2 = 0) r(0,0) + \big(1 - \Pr(\mathrm{a}^1 = \mathrm{a}^2 = 0)\big)\mathbb{E}[r(\mathrm{a}^1, \mathrm{a}^2) | (\mathrm{a}^1, \mathrm{a}^2) \neq (0,0)]$$
$$> 0.6^2 \times 0 - (1 - 0.6^2) = -0.64.$$

The update rule stated in the proposition can be equivalently written as

$$\pi^i_{\text{new}} = \arg\max_{\pi^i} \mathbb{E}_{\mathrm{a}^i \sim \pi^i, \mathrm{a}^{-i} \sim \pi^{-i}_{\text{old}}} \big[Q_{\boldsymbol{\pi}_{\text{old}}}(\mathrm{a}^i, \mathrm{a}^{-i}) \big]. \tag{15}$$

We have

$$\mathbb{E}_{\mathrm{a}^{-i} \sim \pi^{-i}_{\text{old}}} \big[Q_{\boldsymbol{\pi}_{\text{old}}}(0, \mathrm{a}^{-i}) \big] = \pi^{-i} Q(0,0) + (1 - \pi^{-i}) Q(0,1) = \pi^{-i} r(0,0) + (1 - \pi^{-i}) r(0,1) = 2(1 - \pi^{-i}),$$

and similarly

$$\mathbb{E}_{\mathrm{a}^{-i} \sim \pi^{-i}_{\text{old}}} \big[Q_{\boldsymbol{\pi}_{\text{old}}}(1, \mathrm{a}^{-i}) \big] = \pi^{-i} r(1,0) + (1 - \pi^{-i}) r(1,1) = 2\pi^{-i} - (1 - \pi^{-i}) = 3\pi^{-i} - 1.$$

Hence, if $\pi^{-i} > 0.6$, then

$$\mathbb{E}_{\mathrm{a}^{-i} \sim \pi^{-i}_{\text{old}}} \big[Q_{\boldsymbol{\pi}_{\text{old}}}(1, \mathrm{a}^{-i}) \big] = 3\pi^{-i} - 1 > 3 \times 0.6 - 1 = 0.8 > 2 - 2\pi^{-i} = \mathbb{E}_{\mathrm{a}^{-i} \sim \pi^{-i}_{\text{old}}} \big[Q_{\boldsymbol{\pi}_{\text{old}}}(0, \mathrm{a}^{-i}) \big].$$

Therefore, for every i, the solution to Equation (15) is the greedy policy $\pi^i_{\text{new}}(1) = 1$. Therefore,

$$J(\boldsymbol{\pi}_{\text{new}}) = Q(1,1) = r(1,1) = -1,$$

which finishes the proof. ■

Appendix B. Derivation and Analysis of Algorithm 1

B.1 Recap of Existing Results

Lemma 15 (Performance Difference) *Let $\bar{\pi}$ and π be two policies. Then, the following identity holds,*

$$J(\bar{\pi}) - J(\pi) = \mathbb{E}_{\mathrm{s} \sim \rho_{\bar{\pi}}, \mathrm{a} \sim \bar{\pi}} \left[A_\pi(\mathrm{s}, \mathrm{a}) \right].$$

Proof See Kakade and Langford (2002) (Lemma 6.1) or Schulman et al. (2015) (Appendix A). ■

Theorem 16 *(Schulman et al., 2015, Theorem 1) Let π be the current policy and $\bar{\pi}$ be the next candidate policy. We define $L_\pi(\bar{\pi}) = J(\pi) + \mathbb{E}_{\mathrm{s} \sim \rho_\pi, \mathrm{a} \sim \bar{\pi}} \left[A_\pi(s, a) \right], \mathrm{D}^{max}_{KL}(\pi, \bar{\pi}) = \max_s \mathrm{D}_{KL}\left(\pi(\cdot|s), \bar{\pi}(\cdot|s)\right)$. Then the inequality of*

$$J(\bar{\pi}) \geq L_\pi(\bar{\pi}) - C \mathrm{D}^{max}_{KL}\big(\pi, \bar{\pi}\big) \tag{16}$$

holds, where $C = \frac{4\gamma \max_{s,a} |A_\pi(s,a)|}{(1-\gamma)^2}$.

Proof See Schulman et al. (2015) (Appendix A and Equation (9) of the paper). ■

B.2 Analysis of Training of Algorithm 1

Lemma 17 (Multi-Agent Advantage Decomposition) *In any cooperative Markov games, given a joint policy $\boldsymbol{\pi}$, for any state s, and any agent subset $i_{1:m}$, the below equation holds.*

$$A_{\boldsymbol{\pi}}^{i_{1:m}}\left(s, \boldsymbol{a}^{i_{1:m}}\right)=\sum_{j=1}^{m} A_{\boldsymbol{\pi}}^{i_j}\left(s, \boldsymbol{a}^{i_{1:j-1}}, a^{i_j}\right).$$

Proof By the definition of multi-agent advantage function,

$$\begin{aligned}
&A_{\boldsymbol{\pi}}^{i_{1:m}}(s, \boldsymbol{a}^{i_{1:m}}) = Q_{\boldsymbol{\pi}}^{i_{1:m}}(s, \boldsymbol{a}^{i_{1:m}}) - V_{\boldsymbol{\pi}}(s) \\
&= \sum_{k=1}^{m}\left[Q_{\boldsymbol{\pi}}^{i_{1:k}}(s, \boldsymbol{a}^{i_{1:k}}) - Q_{\boldsymbol{\pi}}^{i_{1:k-1}}(s, \boldsymbol{a}^{i_{1:k-1}})\right] \\
&= \sum_{k=1}^{m} A_{\boldsymbol{\pi}}^{i_k}(s, \boldsymbol{a}^{i_{1:k-1}}, a^{i_k}),
\end{aligned}$$

which finishes the proof.
Note that a similar finding has been shown in Kuba et al. (2021). ■

Lemma 18 *Let $\boldsymbol{\pi}=\prod_{i=1}^{n} \pi^i$ and $\bar{\boldsymbol{\pi}}=\prod_{i=1}^{n} \bar{\pi}^i$ be joint policies. Then*

$$\mathrm{D}_{KL}^{max}(\boldsymbol{\pi}, \bar{\boldsymbol{\pi}}) \leq \sum_{i=1}^{n} \mathrm{D}_{KL}^{max}\left(\pi^i, \bar{\pi}^i\right)$$

Proof For any state s, we have

$$\begin{aligned}
&\mathrm{D}_{\mathrm{KL}}\left(\boldsymbol{\pi}(\cdot|s), \bar{\boldsymbol{\pi}}(\cdot|s)\right) = \mathbb{E}_{\mathbf{a}\sim\boldsymbol{\pi}}\left[\log \boldsymbol{\pi}(\mathbf{a}|s) - \log \bar{\boldsymbol{\pi}}(\mathbf{a}|s)\right] \\
&= \mathbb{E}_{\mathbf{a}\sim\boldsymbol{\pi}}\left[\log\left(\prod_{i=1}^{n} \pi^i(\mathrm{a}^i|s)\right) - \log\left(\prod_{i=1}^{n} \bar{\pi}^i(\mathrm{a}^i|s)\right)\right] \\
&= \mathbb{E}_{\mathbf{a}\sim\boldsymbol{\pi}}\left[\sum_{i=1}^{n} \log \pi^i(\mathrm{a}^i|s) - \sum_{i=1}^{n} \log \bar{\pi}^i(\mathrm{a}^i|s)\right] \\
&= \sum_{i=1}^{n} \mathbb{E}_{\mathrm{a}^i\sim\pi^i, \mathbf{a}^{-i}\sim\boldsymbol{\pi}^{-i}}\left[\log \pi^i(\mathrm{a}^i|s) - \log \bar{\pi}^i(\mathrm{a}^i|s)\right] = \sum_{i=1}^{n} \mathrm{D}_{\mathrm{KL}}\left(\pi^i(\cdot|s), \bar{\pi}^i(\cdot|s)\right). \qquad (17)
\end{aligned}$$

Now, taking maximum over s on both sides yields

$$\mathrm{D}_{\mathrm{KL}}^{\max}(\boldsymbol{\pi}, \bar{\boldsymbol{\pi}}) \leq \sum_{i=1}^{n} \mathrm{D}_{\mathrm{KL}}^{\max}\left(\pi^i, \bar{\pi}^i\right),$$

as required. ■

Lemma 19 *Let $\boldsymbol{\pi}$ be a joint policy. Then, for any joint policy $\bar{\boldsymbol{\pi}}$, we have*

$$J(\bar{\boldsymbol{\pi}}) \geq J(\boldsymbol{\pi}) + \sum_{m=1}^{n} \left[L_{\boldsymbol{\pi}}^{i_{1:m}} \left(\bar{\boldsymbol{\pi}}^{i_{1:m-1}}, \bar{\pi}^{i_m} \right) - C \mathrm{D}_{KL}^{max}(\pi^{i_m}, \bar{\pi}^{i_m}) \right], \\ \text{where } C = \frac{4\gamma \max_{s,\boldsymbol{a}} |A_{\boldsymbol{\pi}}(s, \boldsymbol{a})|}{(1-\gamma)^2}. \tag{6}$$

Proof By Theorem 16

$$J(\bar{\boldsymbol{\pi}}) \geq L_{\boldsymbol{\pi}}(\bar{\boldsymbol{\pi}}) - C \mathrm{D}_{\mathrm{KL}}^{\max}(\boldsymbol{\pi}, \bar{\boldsymbol{\pi}}) \\ = J(\boldsymbol{\pi}) + \mathbb{E}_{\mathrm{s} \sim \rho_{\boldsymbol{\pi}}, \mathbf{a} \sim \bar{\boldsymbol{\pi}}} \left[A_{\boldsymbol{\pi}}(\mathrm{s}, \mathbf{a}) \right] - C \mathrm{D}_{\mathrm{KL}}^{\max}(\boldsymbol{\pi}, \bar{\boldsymbol{\pi}}) \\ \text{which by Lemma 4 equals} \\ = J(\boldsymbol{\pi}) + \mathbb{E}_{\mathrm{s} \sim \rho_{\boldsymbol{\pi}}, \mathbf{a} \sim \bar{\boldsymbol{\pi}}} \left[\sum_{m=1}^{n} A_{\boldsymbol{\pi}}^{i_m} \left(\mathrm{s}, \mathbf{a}^{i_{1:m-1}}, \mathrm{a}^{i_m} \right) \right] - C \mathrm{D}_{\mathrm{KL}}^{\max}(\boldsymbol{\pi}, \bar{\boldsymbol{\pi}})$$

and by Lemma 18 this is at least

$$\geq J(\boldsymbol{\pi}) + \mathbb{E}_{\mathrm{s} \sim \rho_{\boldsymbol{\pi}}, \mathbf{a} \sim \bar{\boldsymbol{\pi}}} \left[\sum_{m=1}^{n} A_{\boldsymbol{\pi}}^{i_m} \left(\mathrm{s}, \mathbf{a}^{i_{1:m-1}}, \mathrm{a}^{i_m} \right) \right] - \sum_{m=1}^{n} C \mathrm{D}_{\mathrm{KL}}^{\max}(\pi^{i_m}, \bar{\pi}^{i_m}) \\ = J(\boldsymbol{\pi}) + \sum_{m=1}^{n} \mathbb{E}_{\mathrm{s} \sim \rho_{\boldsymbol{\pi}}, \mathbf{a}^{i_{1:m-1}} \sim \bar{\boldsymbol{\pi}}^{i_{1:m-1}}, \mathrm{a}^{i_m} \sim \bar{\pi}^{i_m}} \left[A_{\boldsymbol{\pi}}^{i_m} \left(\mathrm{s}, \mathbf{a}^{i_{1:m-1}}, \mathrm{a}^{i_m} \right) \right] - \sum_{m=1}^{n} C \mathrm{D}_{\mathrm{KL}}^{\max}(\pi^{i_m}, \bar{\pi}^{i_m}) \\ = J(\boldsymbol{\pi}) + \sum_{m=1}^{n} \left(L_{\boldsymbol{\pi}}^{i_{1:m}} \left(\bar{\boldsymbol{\pi}}^{i_{1:m-1}}, \bar{\pi}^{i_m} \right) - C \mathrm{D}_{\mathrm{KL}}^{\max}(\pi^{i_m}, \bar{\pi}^{i_m}) \right).$$

■

Theorem 7 *A sequence $(\boldsymbol{\pi}_k)_{k=0}^{\infty}$ of joint policies updated by Algorithm 1 has the monotonic improvement property, i.e., $J(\boldsymbol{\pi}_{k+1}) \geq J(\boldsymbol{\pi}_k)$ for all $k \in \mathbb{N}$.*

Proof Let $\boldsymbol{\pi}_0$ be any joint policy. For every $k \geq 0$, the joint policy $\boldsymbol{\pi}_{k+1}$ is obtained from $\boldsymbol{\pi}_k$ by Algorithm 1 update; for $m = 1, \ldots, n$,

$$\pi_{k+1}^{i_m} = \arg\max_{\pi^{i_m}} \left[L_{\boldsymbol{\pi}_k}^{i_{1:m}} \left(\boldsymbol{\pi}_{k+1}^{i_{1:m-1}}, \pi^{i_m} \right) - C \mathrm{D}_{\mathrm{KL}}^{\max} \left(\pi_k^{i_m}, \pi^{i_m} \right) \right].$$

By Theorem 16, we have

$$J(\boldsymbol{\pi}_{k+1}) \geq L_{\boldsymbol{\pi}_k}(\boldsymbol{\pi}_{k+1}) - C\mathrm{D}_{\mathrm{KL}}^{\max}(\boldsymbol{\pi}_k, \boldsymbol{\pi}_{k+1}),$$

which by Lemma 18 is lower-bounded by

$$\geq L_{\boldsymbol{\pi}_k}(\boldsymbol{\pi}_{k+1}) - \sum_{m=1}^{n} C\mathrm{D}_{\mathrm{KL}}^{\max}(\pi_k^{i_m}, \pi_{k+1}^{i_m})$$

$$= J(\boldsymbol{\pi}_k) + \sum_{m=1}^{n} \left(L_{\boldsymbol{\pi}_k}^{i_{1:m}}(\boldsymbol{\pi}_{k+1}^{i_{1:m-1}}, \pi_{k+1}^{i_m}) - C\mathrm{D}_{\mathrm{KL}}^{\max}(\pi_k^{i_m}, \pi_{k+1}^{i_m}) \right), \tag{18}$$

and as for every m, $\pi_{k+1}^{i_m}$ is the argmax, this is lower-bounded by

$$\geq J(\boldsymbol{\pi}_k) + \sum_{m=1}^{n} \left(L_{\boldsymbol{\pi}_k}^{i_{1:m}}(\boldsymbol{\pi}_{k+1}^{i_{1:m-1}}, \pi_k^{i_m}) - C\mathrm{D}_{\mathrm{KL}}^{\max}(\pi_k^{i_m}, \pi_k^{i_m}) \right),$$

which, as mentioned in Definition 5, equals

$$= J(\boldsymbol{\pi}_{\boldsymbol{k}}) + \sum_{m=1}^{n} 0 = J(\boldsymbol{\pi}_k),$$

where the last inequality follows from Equation (5). This proves that Algorithm 1 achieves monotonic improvement. ■

B.3 Analysis of Convergence of Algorithm 1

Theorem 8 *Supposing in Algorithm 1 any permutation of agents has a fixed non-zero probability to begin the update, a sequence $(\boldsymbol{\pi}_k)_{k=0}^{\infty}$ of joint policies generated by the algorithm, in a cooperative Markov game, has a non-empty set of limit points, each of which is a Nash equilibrium.*

Proof

Step 1 (convergence). Firstly, it is clear that the sequence $(J(\boldsymbol{\pi}_k))_{k=0}^{\infty}$ converges as, by Theorem 7, it is non-decreasing and bounded above by $\frac{R_{\max}}{1-\gamma}$. Let us denote the limit by $\bar{J}$. For every k, we denote the tuple of agents, according to whose order the agents perform the sequential updates, by $i_{1:n}^k$, and we note that $\left(i_{1:n}^k\right)_{k\in\mathbb{N}}$ is a random process. Furthermore, we know that the sequence of policies $(\boldsymbol{\pi}_k)$ is bounded, so by Bolzano-Weierstrass Theorem, it has at least one convergent subsequence. Let $\bar{\boldsymbol{\pi}}$ be any limit point of the sequence (note that the set of limit points is a random set), and $\left(\boldsymbol{\pi}_{k_j}\right)_{j=0}^{\infty}$ be a subsequence converging to $\bar{\boldsymbol{\pi}}$ (which is a random subsequence as well). By continuity of J in $\boldsymbol{\pi}$, we have

$$J(\bar{\boldsymbol{\pi}}) = J\left(\lim_{j\to\infty} \boldsymbol{\pi}_{k_j}\right) = \lim_{j\to\infty} J\left(\boldsymbol{\pi}_{k_j}\right) = \bar{J}. \tag{19}$$

For now, we introduce an auxiliary definition.

Definition 20 (TR-Stationarity) *A joint policy $\bar{\boldsymbol{\pi}}$ is trust-region-stationary (TR-stationary) if, for every agent i,*

$$\bar{\pi}^i = \arg\max_{\pi^i} \left[\mathbb{E}_{\mathrm{s}\sim\rho_{\bar{\boldsymbol{\pi}}},\mathrm{a}^i\sim\pi^i} \left[A^i_{\bar{\boldsymbol{\pi}}}(\mathrm{s},\mathrm{a}^i) \right] - C_{\bar{\boldsymbol{\pi}}} \mathrm{D}^{max}_{KL}\left(\bar{\pi}^i, \pi^i\right) \right],$$

where $C_{\bar{\boldsymbol{\pi}}} = \frac{4\gamma\epsilon}{(1-\gamma)^2}$, *and* $\epsilon = \max_{s,\boldsymbol{a}} |A_{\bar{\boldsymbol{\pi}}}(s,\boldsymbol{a})|$.

We will now establish the TR-stationarity of any limit point joint policy $\bar{\boldsymbol{\pi}}$ (which, as stated above, is a random variable). Let $\mathbb{E}_{i^{0:\infty}_{1:n}}[\cdot]$ denote the expected value operator under the random process $(i^{0:\infty}_{1:n})$. Let also $\epsilon_k = \max_{s,\boldsymbol{a}} |A_{\boldsymbol{\pi}_k}(s,\boldsymbol{a})|$, and $C_k = \frac{4\gamma\epsilon_k}{(1-\gamma)^2}$. We have

$$\begin{aligned}
0 &= \lim_{k\to\infty} \mathbb{E}_{i^{0:\infty}_{1:n}} \left[J(\boldsymbol{\pi}_{k+1}) - J(\boldsymbol{\pi}_k) \right] \\
&\geq \lim_{k\to\infty} \mathbb{E}_{i^{0:\infty}_{1:n}} \left[L_{\boldsymbol{\pi}_k}(\boldsymbol{\pi}_{k+1}) - C_k \mathrm{D}^{\max}_{\mathrm{KL}}(\boldsymbol{\pi}_k, \boldsymbol{\pi}_{k+1}) \right] \quad \text{by Theorem 16} \\
&\geq \lim_{k\to\infty} \mathbb{E}_{i^{0:\infty}_{1:n}} \left[L^{i^k_1}_{\boldsymbol{\pi}_k}\left(\pi^{i^k_1}_{k+1}\right) - C_k \mathrm{D}^{\max}_{\mathrm{KL}}\left(\pi^{i^k_1}_k, \pi^{i^k_1}_{k+1}\right) \right] \\
&\text{by Equation (18) and the fact that each of its summands is non-negative.}
\end{aligned}$$

Now, we consider an arbitrary limit point $\bar{\boldsymbol{\pi}}$ from the (random) limit set, and a (random) subsequence $\left(\boldsymbol{\pi}_{k_j}\right)_{j=0}^{\infty}$ that converges to $\bar{\boldsymbol{\pi}}$. We get

$$0 \geq \lim_{j\to\infty} \mathbb{E}_{i^{0:\infty}_{1:n}} \left[L^{i^{k_j}_1}_{\boldsymbol{\pi}_{k_j}}\left(\pi^{i^{k_j}_1}_{k_j+1}\right) - C_{k_j} \mathrm{D}^{\max}_{\mathrm{KL}}\left(\pi^{i^{k_j}_1}_{k_j}, \pi^{i^{k_j}_1}_{k_j+1}\right) \right].$$

As the expectation is taken of non-negative random variables, and for every $i \in \mathcal{N}$ and $k \in \mathbb{N}$, with some positive probability p_i, we have $i^{k_j}_1 = i$ (because every permutation has non-zero probability), the above is bounded from below by

$$\begin{aligned}
&p_i \lim_{j\to\infty} \max_{\pi^i} \left[L^i_{\boldsymbol{\pi}_{k_j}}(\pi^i) - C_{k_j} \mathrm{D}^{\max}_{\mathrm{KL}}\left(\pi^i_{k_j}, \pi^i\right) \right], \\
&\text{which, as } \boldsymbol{\pi}_{k_j} \text{ converges to } \bar{\boldsymbol{\pi}}, \text{ equals to} \\
&p_i \max_{\pi^i} \left[L^i_{\bar{\boldsymbol{\pi}}}(\pi^i) - C_{\bar{\boldsymbol{\pi}}} \mathrm{D}^{\max}_{\mathrm{KL}}\left(\bar{\pi}^i, \pi^i\right) \right] \geq 0, \quad \text{by Equation (5).}
\end{aligned}$$

This proves that, for any limit point $\bar{\boldsymbol{\pi}}$ of the random process $(\boldsymbol{\pi}_k)$ induced by Algorithm 1, $\max_{\pi^i} \left[L^i_{\bar{\boldsymbol{\pi}}}(\pi^i) - C_{\bar{\boldsymbol{\pi}}} \mathrm{D}^{\max}_{\mathrm{KL}}\left(\bar{\pi}^i, \pi^i\right) \right] = 0$, which is equivalent with Definition 20.

Step 2 (dropping the penalty term). Now, we have to prove that TR-stationary points are NEs of cooperative Markov games. The main step is to prove the following statement: *a TR-stationary joint policy* $\bar{\boldsymbol{\pi}}$, *for every state* $s \in \mathcal{S}$, *satisfies*

$$\bar{\pi}^i = \arg\max_{\pi^i} \mathbb{E}_{\mathrm{a}^i\sim\pi^i} \left[A^i_{\bar{\boldsymbol{\pi}}}(s,\mathrm{a}^i) \right]. \tag{20}$$

We will use the technique of the proof by contradiction. Suppose that there is a state s_0 such that there exists a policy $\hat{\pi}^i$ with

$$\mathbb{E}_{\mathrm{a}^i\sim\hat{\pi}^i} \left[A^i_{\bar{\boldsymbol{\pi}}}(s_0,\mathrm{a}^i) \right] > \mathbb{E}_{\mathrm{a}^i\sim\bar{\pi}^i} \left[A^i_{\bar{\boldsymbol{\pi}}}(s_0,\mathrm{a}^i) \right]. \tag{21}$$

Let us parametrise the policies π^i according to the template

$$\pi^i(\cdot|s_0) = \big(x^i_1, \dots, x^i_{d^i-1}, 1 - \sum_{j=1}^{d^i-1} x^i_j\big)$$

where the values of x^i_j $(j = 1, \dots, d^i - 1)$ are such that $\pi^i(\cdot|s_0)$ is a valid probability distribution. Then we can rewrite our quantity of interest (the objective of Equation (20) as

$$\begin{aligned}\mathbb{E}_{\mathrm{a}^i\sim\pi^i}\big[A^i_{\overline{\boldsymbol{\pi}}}(s_0, \mathrm{a}^i)\big] &= \sum_{j=1}^{d^i-1} x^i_j \cdot A^i_{\overline{\boldsymbol{\pi}}}\big(s_0, a^i_j\big) + (1 - \sum_{h=1}^{d^i-1} x^i_h) A^i_{\overline{\boldsymbol{\pi}}}\big(s_0, a^i_{d^i}\big) \\ &= \sum_{j=1}^{d^i-1} x^i_j \big[A^i_{\overline{\boldsymbol{\pi}}}\big(s_0, a^i_j\big) - A^i_{\overline{\boldsymbol{\pi}}}\big(s_0, a^i_{d^i}\big)\big] + A^i_{\overline{\boldsymbol{\pi}}}\big(s_0, a^i_{d^i}\big),\end{aligned}$$

which is an affine function of the policy parameterisation. It follows that its gradient (with respect to x^i) and directional derivatives are constant in the space of policies at state s_0. The existance of policy $\hat{\pi}^i(\cdot|s_0)$, for which Inequality (21) holds, implies that the directional derivative in the direction from $\bar{\pi}^i(\cdot|s_0)$ to $\hat{\pi}^i(\cdot|s_0)$ is strictly positive. We also have

$$\begin{aligned}\frac{\partial \mathrm{D}_{\mathrm{KL}}(\bar{\pi}^i(\cdot|s_0), \pi^i(\cdot|s_0))}{\partial x^i_j} &= \frac{\partial}{\partial x^i_j}\left[(\bar{\pi}^i(\cdot|s_0))^T(\log \bar{\pi}^i(\cdot|s_0) - \log \pi^i(\cdot|s_0))\right] \\ &= \frac{\partial}{\partial x^i_j}\left[-(\bar{\pi}^i)^T \log \pi^i\right] \text{ (omitting state } s_0 \text{ for brevity)} \\ &= -\frac{\partial}{\partial x^i_j}\sum_{k=1}^{d_i-1} \bar{\pi}^i_k \log x^i_k - \frac{\partial}{\partial x^i_j}\bar{\pi}^i_{d_i} \log\left(1 - \sum_{k=1}^{d_i-1} x^i_k\right) \\ &= -\frac{\bar{\pi}^i_j}{x^i_j} + \frac{\bar{\pi}^i_{d_i}}{1 - \sum_{k=1}^{d_i-1} x^i_k} \\ &= -\frac{\bar{\pi}^i_j}{\pi^i_j} + \frac{\bar{\pi}^i_{d_i}}{\pi^i_{d_i}} = 0, \quad \text{when evaluated at } \pi^i = \bar{\pi}^i, \end{aligned} \tag{22}$$

which means that the KL-penalty has zero gradient at $\bar{\pi}^i(\cdot|s_0)$. Hence, when evaluated at $\pi^i(\cdot|s_0) = \bar{\pi}^i(\cdot|s_0)$, the objective

$$\rho_{\overline{\boldsymbol{\pi}}}(s_0)\mathbb{E}_{\mathrm{a}^i\sim\pi^i}\big[A^i_{\overline{\boldsymbol{\pi}}}(s_0, \mathrm{a}^i)\big] - C_{\overline{\boldsymbol{\pi}}}\mathrm{D}_{\mathrm{KL}}\big(\bar{\pi}^i(\cdot|s_0), \pi^i(\cdot|s_0)\big)$$

has a strictly positive directional derivative in the direction of $\hat{\pi}^i(\cdot|s_0)$. Thus, there exists a policy $\widetilde{\pi}^i(\cdot|s_0)$, sufficiently close to $\bar{\pi}^i(\cdot|s_0)$ on the path joining it with $\hat{\pi}^i(\cdot|s_0)$, for which

$$\rho_{\overline{\boldsymbol{\pi}}}(s_0)\mathbb{E}_{\mathrm{a}^i\sim\widetilde{\pi}^i}\big[A^i_{\overline{\boldsymbol{\pi}}}(s_0, \mathrm{a}^i)\big] - C_{\overline{\boldsymbol{\pi}}}\mathrm{D}_{\mathrm{KL}}\big(\bar{\pi}^i(\cdot|s_0), \widetilde{\pi}^i(\cdot|s_0)\big) > 0.$$

Let π^i_* be a policy such that $\pi^i_*(\cdot|s_0) = \widetilde{\pi}^i(\cdot|s_0)$, and $\pi^i_*(\cdot|s) = \bar{\pi}^i(\cdot|s)$ for states $s \neq s_0$. As for these states we have

$$\rho_{\overline{\boldsymbol{\pi}}}(s)\mathbb{E}_{\mathrm{a}^i\sim\pi^i_*}\big[A^i_{\overline{\boldsymbol{\pi}}}(s, \mathrm{a}^i)\big] = \rho_{\overline{\boldsymbol{\pi}}}(s)\mathbb{E}_{\mathrm{a}^i\sim\bar{\pi}^i}\big[A^i_{\overline{\boldsymbol{\pi}}}(s, \mathrm{a}^i)\big] = 0, \quad \text{and} \quad \mathrm{D}_{\mathrm{KL}}(\bar{\pi}^i(\cdot|s), \pi^i_*(\cdot|s)) = 0,$$

it follows that

$$L^i_{\bar{\boldsymbol{\pi}}}(\pi^i_*) - C_{\bar{\boldsymbol{\pi}}}\mathrm{D}^{\max}_{\mathrm{KL}}(\bar{\pi}^i, \pi^i_*) = \rho_{\bar{\boldsymbol{\pi}}}(s_0)\mathbb{E}_{\mathrm{a}^i\sim\widetilde{\pi}^i}\big[A^i_{\bar{\boldsymbol{\pi}}}(s_0, \mathrm{a}^i)\big] - C_{\bar{\boldsymbol{\pi}}}\mathrm{D}_{\mathrm{KL}}\big(\bar{\pi}^i(\cdot|s_0), \widetilde{\pi}^i(\cdot|s_0)\big)$$
$$> 0 = L^i_{\bar{\boldsymbol{\pi}}}(\bar{\pi}^i) - C_{\bar{\boldsymbol{\pi}}}\mathrm{D}^{\max}_{\mathrm{KL}}(\bar{\pi}^i, \bar{\pi}^i),$$

which is a contradiction with TR-stationarity of $\bar{\boldsymbol{\pi}}$. Hence, the claim of Equation (20) is proved.

Step 3 (optimality). Now, for a fixed joint policy $\bar{\boldsymbol{\pi}}^{-i}$ of other agents, $\bar{\pi}^i$ satisfies

$$\bar{\pi}^i = \arg\max_{\pi^i} \mathbb{E}_{\mathrm{a}^i\sim\pi^i}\big[A^i_{\bar{\boldsymbol{\pi}}}(s, \mathrm{a}^i)\big] = \arg\max_{\pi^i} \mathbb{E}_{\mathrm{a}^i\sim\pi^i}\big[Q^i_{\bar{\boldsymbol{\pi}}}(s, \mathrm{a}^i)\big],\ \forall s \in \mathcal{S},$$

which is the Bellman optimality equation (Sutton and Barto, 2018). Hence, for a fixed joint policy $\bar{\boldsymbol{\pi}}^{-i}$, the policy $\bar{\pi}^i$ is optimal:

$$\bar{\pi}^i = \arg\max_{\pi^i} J(\pi^i, \bar{\boldsymbol{\pi}}^{-i}).$$

As agent i was chosen arbitrarily, $\bar{\boldsymbol{\pi}}$ is a Nash equilibrium. ■

Appendix C. HATRPO and HAPPO

C.1 Proof of Proposition 9

Proposition 21 *Let $\boldsymbol{\pi} = \prod_{j=1}^n \pi^{i_j}$ be a joint policy, and $A_{\boldsymbol{\pi}}(\mathrm{s}, \mathbf{a})$ be its joint advantage function. Let $\bar{\boldsymbol{\pi}}^{i_{1:m-1}} = \prod_{j=1}^{m-1} \bar{\pi}^{i_j}$ be some **other** joint policy of agents $i_{1:m-1}$, and $\hat{\pi}^{i_m}$ be some **other** policy of agent i_m. Then, for every state s,*

$$\mathbb{E}_{\mathbf{a}^{i_{1:m-1}}\sim\bar{\boldsymbol{\pi}}^{i_{1:m-1}}, \mathrm{a}^{i_m}\sim\hat{\pi}^{i_m}}\big[A^{i_m}_{\boldsymbol{\pi}}\big(s, \mathbf{a}^{i_{1:m-1}}, \mathrm{a}^{i_m}\big)\big]$$
$$= \mathbb{E}_{\mathbf{a}\sim\boldsymbol{\pi}}\Big[\Big(\frac{\hat{\pi}^{i_m}(\mathrm{a}^{i_m}|s)}{\pi^{i_m}(\mathrm{a}^{i_m}|s)} - 1\Big)\frac{\bar{\boldsymbol{\pi}}^{i_{1:m-1}}(\mathbf{a}^{i_{1:m-1}}|s)}{\boldsymbol{\pi}^{i_{1:m-1}}(\mathbf{a}^{i_{1:m-1}}|s)}A_{\boldsymbol{\pi}}(s, \mathbf{a})\Big]. \tag{9}$$

Proof

$$
\begin{aligned}
&\mathbb{E}_{\mathbf{a}\sim\boldsymbol{\pi}}\Big[\Big(\frac{\hat{\pi}^{i_m}(\mathrm{a}^{i_m}|s)}{\pi^{i_m}(\mathrm{a}^{i_m}|s)}-1\Big)\frac{\overline{\boldsymbol{\pi}}^{i_{1:m-1}}(\mathbf{a}^{i_{1:m-1}}|s)}{\boldsymbol{\pi}^{i_{1:m-1}}(\mathbf{a}^{i_{1:m-1}}|s)}A_{\boldsymbol{\pi}}(s,\mathbf{a})\Big]\\
&=\mathbb{E}_{\mathbf{a}\sim\boldsymbol{\pi}}\left[\frac{\hat{\pi}^{i_m}(\mathrm{a}^{i_m}|s)\overline{\boldsymbol{\pi}}^{i_{1:m-1}}(\mathbf{a}^{i_{1:m-1}}|s)}{\boldsymbol{\pi}^{i_{1:m}}(\mathbf{a}^{i_{1:m}}|s)}A_{\boldsymbol{\pi}}(s,\mathbf{a})-\frac{\overline{\boldsymbol{\pi}}^{i_{1:m-1}}(\mathbf{a}^{i_{1:m-1}}|s)}{\boldsymbol{\pi}^{i_{1:m-1}}(\mathbf{a}^{i_{1:m-1}}|s)}A_{\boldsymbol{\pi}}(s,\mathbf{a})\right]\\
&=\mathbb{E}_{\mathbf{a}^{i_{1:m}}\sim\boldsymbol{\pi}^{i_{1:m}},\mathbf{a}^{-i_{1:m}}\sim\boldsymbol{\pi}^{-i_{1:m}}}\left[\frac{\hat{\pi}^{i_m}(\mathrm{a}^{i_m}|s)\overline{\boldsymbol{\pi}}^{i_{1:m-1}}(\mathbf{a}^{i_{1:m-1}}|s)}{\boldsymbol{\pi}^{i_{1:m}}(\mathbf{a}^{i_{1:m}}|s)}A_{\boldsymbol{\pi}}(s,\mathbf{a}^{i_{1:m}},\mathbf{a}^{-i_{1:m}})\right]\\
&\quad-\mathbb{E}_{\mathbf{a}^{i_{1:m-1}}\sim\boldsymbol{\pi}^{i_{1:m-1}},\mathbf{a}^{-i_{1:m-1}}\sim\boldsymbol{\pi}^{-i_{1:m-1}}}\left[\frac{\overline{\boldsymbol{\pi}}^{i_{1:m-1}}(\mathbf{a}^{i_{1:m-1}}|s)}{\boldsymbol{\pi}^{i_{1:m-1}}(\mathbf{a}^{i_{1:m-1}}|s)}A_{\boldsymbol{\pi}}(s,\mathbf{a}^{i_{1:m-1}},\mathbf{a}^{-i_{1:m-1}})\right]\\
&=\mathbb{E}_{\mathbf{a}^{i_{1:m-1}}\sim\overline{\boldsymbol{\pi}}^{i_{1:m-1}},\mathrm{a}^{i_m}\sim\hat{\pi}^{i_m},\mathbf{a}^{-i_{1:m}}\sim\boldsymbol{\pi}^{-i_{1:m}}}\left[A_{\boldsymbol{\pi}}(s,\mathbf{a}^{i_{1:m}},\mathbf{a}^{-i_{1:m}})\right]\\
&\quad-\mathbb{E}_{\mathbf{a}^{i_{1:m-1}}\sim\overline{\boldsymbol{\pi}}^{i_{1:m-1}},\mathbf{a}^{-i_{1:m-1}}\sim\boldsymbol{\pi}^{-i_{1:m-1}}}\left[A_{\boldsymbol{\pi}}(s,\mathbf{a}^{i_{1:m-1}},\mathbf{a}^{-i_{1:m-1}})\right]\\
&=\mathbb{E}_{\mathbf{a}^{i_{1:m-1}}\sim\overline{\boldsymbol{\pi}}^{i_{1:m-1}},\mathrm{a}^{i_m}\sim\hat{\pi}^{i_m}}\left[\mathbb{E}_{\mathbf{a}^{-i_{1:m}}\sim\boldsymbol{\pi}^{-i_{1:m}}}\left[A_{\boldsymbol{\pi}}(s,\mathbf{a}^{i_{1:m}},\mathbf{a}^{-i_{1:m}})\right]\right]\\
&\quad-\mathbb{E}_{\mathbf{a}^{i_{1:m-1}}\sim\overline{\boldsymbol{\pi}}^{i_{1:m-1}}}\left[\mathbb{E}_{\mathbf{a}^{-i_{1:m-1}}\sim\boldsymbol{\pi}^{-i_{1:m-1}}}\left[A_{\boldsymbol{\pi}}(s,\mathbf{a}^{i_{1:m-1}},\mathbf{a}^{-i_{1:m-1}})\right]\right]\\
&=\mathbb{E}_{\mathbf{a}^{i_{1:m-1}}\sim\overline{\boldsymbol{\pi}}^{i_{1:m-1}},\mathrm{a}^{i_m}\sim\hat{\pi}^{i_m}}\left[A_{\boldsymbol{\pi}}^{i_{1:m}}(s,\mathbf{a}^{i_{1:m}})\right]\\
&\quad-\mathbb{E}_{\mathbf{a}^{i_{1:m-1}}\sim\overline{\boldsymbol{\pi}}^{i_{1:m-1}}}\left[A_{\boldsymbol{\pi}}^{i_{1:m-1}}(s,\mathbf{a}^{i_{1:m-1}})\right],\\
&=\mathbb{E}_{\mathbf{a}^{i_{1:m-1}}\sim\overline{\boldsymbol{\pi}}^{i_{1:m-1}},\mathrm{a}^{i_m}\sim\hat{\pi}^{i_m}}\left[A_{\boldsymbol{\pi}}^{i_{1:m}}(s,\mathbf{a}^{i_{1:m}})-A_{\boldsymbol{\pi}}^{i_{1:m-1}}(s,\mathbf{a}^{i_{1:m-1}})\right]
\end{aligned}
$$

which, by Lemma 4, equals

$$
=\mathbb{E}_{\mathbf{a}^{i_{1:m-1}}\sim\overline{\boldsymbol{\pi}}^{i_{1:m-1}},\mathrm{a}^{i_m}\sim\hat{\pi}^{i_m}}\left[A_{\boldsymbol{\pi}}^{i_m}(s,\mathbf{a}^{i_{1:m-1}},\mathrm{a}^{i_m})\right].
$$

■

C.2 Derivation of the gradient estimator for HATRPO

$$
\begin{aligned}
&\nabla_{\theta^{i_m}}\mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}_{\boldsymbol{\theta}_k}},\mathbf{a}\sim\boldsymbol{\pi}_{\boldsymbol{\theta}_k}}\left[\left(\frac{\pi_{\theta^{i_m}}^{i_m}(\mathrm{a}^{i_m}|\mathrm{s})}{\pi_{\theta_k^{i_m}}^{i_m}(\mathrm{a}^{i_m}|\mathrm{s})}-1\right)M^{i_{1:m}}(\mathrm{s},\mathbf{a})\right]\\
&=\nabla_{\theta^{i_m}}\mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}_{\boldsymbol{\theta}_k}},\mathbf{a}\sim\boldsymbol{\pi}_{\boldsymbol{\theta}_k}}\left[\frac{\pi_{\theta^{i_m}}^{i_m}(\mathrm{a}^{i_m}|\mathrm{s})}{\pi_{\theta_k^{i_m}}^{i_m}(\mathrm{a}^{i_m}|\mathrm{s})}M^{i_{1:m}}(\mathrm{s},\mathbf{a})\right]-\nabla_{\theta^{i_m}}\mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}_{\boldsymbol{\theta}_k}},\mathbf{a}\sim\boldsymbol{\pi}_{\boldsymbol{\theta}_k}}\left[M^{i_{1:m}}(\mathrm{s},\mathbf{a})\right]\\
&=\mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}_{\boldsymbol{\theta}_k}},\mathbf{a}\sim\boldsymbol{\pi}_{\boldsymbol{\theta}_k}}\left[\frac{\nabla_{\theta^{i_m}}\pi_{\theta^{i_m}}^{i_m}(\mathrm{a}^{i_m}|\mathrm{s})}{\pi_{\theta_k^{i_m}}^{i_m}(\mathrm{a}^{i_m}|\mathrm{s})}M^{i_{1:m}}(\mathrm{s},\mathbf{a})\right]\\
&=\mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}_{\boldsymbol{\theta}_k}},\mathbf{a}\sim\boldsymbol{\pi}_{\boldsymbol{\theta}_k}}\left[\frac{\pi_{\theta^{i_m}}^{i_m}(\mathrm{a}^{i_m}|\mathrm{s})}{\pi_{\theta_k^{i_m}}^{i_m}(\mathrm{a}^{i_m}|\mathrm{s})}\nabla_{\theta^{i_m}}\log\pi_{\theta^{i_m}}^{i_m}(\mathrm{a}^{i_m}|\mathrm{s})M^{i_{1:m}}(\mathrm{s},\mathbf{a})\right].
\end{aligned}
$$

Evaluated at $\theta^{i_m}=\theta_k^{i_m}$, the above expression equals

$$
\mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}_{\boldsymbol{\theta}_k}},\mathbf{a}\sim\boldsymbol{\pi}_{\boldsymbol{\theta}_k}}\left[M^{i_{1:m}}(\mathrm{s},\mathbf{a})\nabla_{\theta^{i_m}}\log\pi_{\theta^{i_m}}^{i_m}(\mathrm{a}^{i_m}|\mathrm{s})\big|_{\theta^{i_m}=\theta_k^{i_m}}\right],
$$

which finishes the derivation.

C.3 Pseudocode of HATRPO

Algorithm 3: HATRPO

Input: Stepsize α, batch size B, number of: agents n, episodes K, steps per episode T, possible steps in line search L, line search acceptance threshold κ.

Initialize: Actor networks $\{\theta_0^i,\ \forall i \in \mathcal{N}\}$, Global V-value network $\{\phi_0\}$, Replay buffer $\mathcal{B}$

for $k = 0, 1, \ldots, K-1$ **do**

Collect a set of trajectories by running the joint policy $\boldsymbol{\pi}_{\boldsymbol{\theta}_k} = (\pi^1_{\theta^1_k}, \ldots, \pi^n_{\theta^n_k})$.

Push transitions $\{(s_t, o^i_t, a^i_t, r_t, s_{t+1}, o^i_{t+1}), \forall i \in \mathcal{N}, t \in T\}$ into $\mathcal{B}$.

Sample a random minibatch of B transitions from $\mathcal{B}$.

Compute advantage function $\hat{A}(\mathrm{s}, \mathbf{a})$ based on global V-value network with GAE.

Draw a random permutation of agents $i_{1:n}$.

Set $M^{i_1}(\mathrm{s}, \mathbf{a}) = \hat{A}(\mathrm{s}, \mathbf{a})$.

for *agent* $i_m = i_1, \ldots, i_n$ **do**

Estimate the gradient of the agent's maximisation objective

$$\hat{\boldsymbol{g}}^{i_m}_k = \frac{1}{B} \sum_{b=1}^{B} \sum_{t=1}^{T} \nabla_{\theta^{i_m}_k} \log \pi^{i_m}_{\theta^{i_m}_k} \left(a^{i_m}_t \mid o^{i_m}_t \right) M^{i_{1:m}}(s_t, \boldsymbol{a}_t).$$

Use the conjugate gradient algorithm to compute the update direction

$$\hat{\boldsymbol{x}}^{i_m}_k \approx (\hat{\boldsymbol{H}}^{i_m}_k)^{-1} \hat{\boldsymbol{g}}^{i_m}_k,$$

where $\hat{\boldsymbol{H}}^{i_m}_k$ is the Hessian of the average KL-divergence

$$\frac{1}{BT} \sum_{b=1}^{B} \sum_{t=1}^{T} \mathrm{D}_{\mathrm{KL}} \left(\pi^{i_m}_{\theta^{i_m}_k}(\cdot | o^{i_m}_t), \pi^{i_m}_{\theta^{i_m}}(\cdot | o^{i_m}_t) \right).$$

Estimate the maximal step size allowing for meeting the KL-constraint

$$\hat{\beta}^{i_m}_k \approx \sqrt{\frac{2\delta}{(\hat{\boldsymbol{x}}^{i_m}_k)^T \hat{\boldsymbol{H}}^{i_m}_k \hat{\boldsymbol{x}}^{i_m}_k}}.$$

Update agent i_m's policy by

$$\theta^{i_m}_{k+1} = \theta^{i_m}_k + \alpha^j \hat{\beta}^{i_m}_k \hat{\boldsymbol{x}}^{i_m}_k,$$

where $j \in \{0, 1, \ldots, L\}$ is the smallest such j which improves the sample loss by at least $\kappa \alpha^j \hat{\beta}^{i_m}_k \hat{\boldsymbol{x}}^{i_m}_k \cdot \hat{\boldsymbol{g}}^{i_m}_k$, found by the backtracking line search.

Compute $M^{i_{1:m+1}}(\mathrm{s}, \mathbf{a}) = \frac{\pi^{i_m}_{\theta^{i_m}_{k+1}}(\mathrm{a}^{i_m} | \mathrm{o}^{i_m})}{\pi^{i_m}_{\theta^{i_m}_k}(\mathrm{a}^{i_m} | \mathrm{o}^{i_m})} M^{i_{1:m}}(\mathrm{s}_t, \mathbf{a}_t)$. //Unless $m = n$.

Update V-value network by following formula:

$$\phi_{k+1} = \arg\min_\phi \frac{1}{BT} \sum_{b=1}^{B} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2$$

C.4 Pseudocode of HAPPO

Algorithm 4: HAPPO

Input: Stepsize α, batch size B, number of: agents n, episodes K, steps per episode T.

Initialize: Actor networks $\{\theta_0^i,\ \forall i \in \mathcal{N}\}$, Global V-value network $\{\phi_0\}$, Replay buffer $\mathcal{B}$

for $k = 0, 1, \ldots, K-1$ **do**

Collect a set of trajectories by running the joint policy $\boldsymbol{\pi}_{\boldsymbol{\theta}_k} = (\pi^1_{\theta^1_k}, \ldots, \pi^n_{\theta^n_k})$.

Push transitions $\{(s_t, o_t^i, a_t^i, r_t, s_{t+1}, o_{t+1}^i), \forall i \in \mathcal{N}, t \in T\}$ into $\mathcal{B}$.

Sample a random minibatch of B transitions from $\mathcal{B}$.

Compute advantage function $\hat{A}(\mathrm{s}, \mathbf{a})$ based on global V-value network with GAE.

Draw a random permutation of agents $i_{1:n}$.

Set $M^{i_1}(\mathrm{s}, \mathbf{a}) = \hat{A}(\mathrm{s}, \mathbf{a})$.

for *agent* $i_m = i_1, \ldots, i_n$ **do**

Update actor i_m with $\theta^{i_m}_{k+1}$, the argmax of the PPO-Clip objective

$$\frac{1}{BT}\sum_{b=1}^{B}\sum_{t=0}^{T}\min\left(\frac{\pi^{i_m}_{\theta^{i_m}}(a_t^{i_m}|o_t^{i_m})}{\pi^{i_m}_{\theta^{i_m}_k}(a_t^{i_m}|o_t^{i_m})}M^{i_{1:m}}(s_t, \boldsymbol{a}_t),\ \mathrm{clip}\left(\frac{\pi^{i_m}_{\theta^{i_m}}(a_t^{i_m}|o_t^{i_m})}{\pi^{i_m}_{\theta^{i_m}_k}(a_t^{i_m}|o_t^{i_m})}, 1\pm\epsilon\right)M^{i_{1:m}}(s_t, \boldsymbol{a}_t)\right).$$

Compute $M^{i_{1:m+1}}(\mathrm{s}, \mathbf{a}) = \frac{\pi^{i_m}_{\theta^{i_m}_{k+1}}(\mathrm{a}^{i_m}|\mathrm{o}^{i_m})}{\pi^{i_m}_{\theta^{i_m}_k}(\mathrm{a}^{i_m}|\mathrm{o}^{i_m})}M^{i_{1:m}}(\mathrm{s}, \mathbf{a})$. //Unless $m = n$.

Update V-value network by the following formula:

$$\phi_{k+1} = \arg\min_\phi \frac{1}{BT}\sum_{b=1}^{B}\sum_{t=0}^{T}\left(V_\phi(s_t) - \hat{R}_t\right)^2$$

Appendix D. Proof of *HAMO Is All You Need* Lemma

Lemma 22 (HAMO Is All You Need) *Let $\boldsymbol{\pi}_{old}$ and $\boldsymbol{\pi}_{new}$ be joint policies and let $i_{1:n} \in \mathrm{Sym}(n)$ be an agent permutation. Suppose that, for every state $s \in \mathcal{S}$ and every $m = 1, \ldots, n$,*

$$\left[\mathcal{M}^{(\pi^{i_m}_{new})}_{\mathfrak{D}^{i_m}, \boldsymbol{\pi}^{i_{1:m-1}}_{new}} A_{\boldsymbol{\pi}_{old}}\right](s) \geq \left[\mathcal{M}^{(\pi^{i_m}_{old})}_{\mathfrak{D}^{i_m}, \boldsymbol{\pi}^{i_{1:m-1}}_{new}} A_{\boldsymbol{\pi}_{old}}\right](s). \tag{12}$$

Then, $\boldsymbol{\pi}_{new}$ is jointly better than $\boldsymbol{\pi}_{old}$, so that for every state s,

$$V_{\boldsymbol{\pi}_{new}}(s) \geq V_{\boldsymbol{\pi}_{old}}(s).$$

Proof Let $\widetilde{\mathfrak{D}}_{\boldsymbol{\pi}_{\text{old}}}(\boldsymbol{\pi}_{\text{new}}|s) \triangleq \sum_{m=1}^{n} \mathfrak{D}^{i_m}_{\boldsymbol{\pi}_{\text{old}}}(\pi^{i_m}_{\text{new}}|s, \boldsymbol{\pi}^{i_{1:m-1}}_{\text{new}})$. Combining this with Lemma 4 gives

$$
\begin{aligned}
&\mathbb{E}_{\mathbf{a}\sim\boldsymbol{\pi}_{\text{new}}}\big[A_{\boldsymbol{\pi}_{\text{old}}}(s,\mathbf{a})\big] - \widetilde{\mathfrak{D}}_{\boldsymbol{\pi}_{\text{old}}}(\boldsymbol{\pi}_{\text{new}}|s) \\
&= \sum_{m=1}^{n} \big[\mathbb{E}_{\mathbf{a}^{i_{1:m-1}}\sim\boldsymbol{\pi}^{i_{1:m-1}}_{\text{new}}, \mathrm{a}^{i_m}\sim\pi^{i_m}_{\text{new}}}\big[A^{i_m}_{\boldsymbol{\pi}_{\text{old}}}(s,\mathbf{a}^{i_{1:m-1}},\mathrm{a}^{i_m})\big] - \mathfrak{D}^{i_m}_{\boldsymbol{\pi}_{\text{old}}}(\pi^{i_m}_{\text{new}}|s,\boldsymbol{\pi}^{i_{1:m-1}}_{\text{new}})\big] \\
&\qquad \text{by Inequality (12)} \\
&\geq \sum_{m=1}^{n} \big[\mathbb{E}_{\mathbf{a}^{i_{1:m-1}}\sim\boldsymbol{\pi}^{i_{1:m-1}}_{\text{new}}, \mathrm{a}^{i_m}\sim\pi^{i_m}_{\text{old}}}\big[A^{i_m}_{\boldsymbol{\pi}_{\text{old}}}(s,\mathbf{a}^{i_{1:m-1}},\mathrm{a}^{i_m})\big] - \mathfrak{D}^{i_m}_{\boldsymbol{\pi}_{\text{old}}}(\pi^{i_m}_{\text{old}}|s,\boldsymbol{\pi}^{i_{1:m-1}}_{\text{new}})\big] \\
&= \mathbb{E}_{\mathbf{a}\sim\boldsymbol{\pi}_{\text{old}}}\big[A_{\boldsymbol{\pi}_{\text{old}}}(s,\mathbf{a})\big] - \widetilde{\mathfrak{D}}_{\boldsymbol{\pi}_{\text{old}}}(\boldsymbol{\pi}_{\text{old}}|s).
\end{aligned}
$$

The resulting inequality can be equivalently rewritten as

$$
\mathbb{E}_{\mathbf{a}\sim\boldsymbol{\pi}_{\text{new}}}\big[Q_{\boldsymbol{\pi}_{\text{old}}}(s,\mathbf{a})\big] - \widetilde{\mathfrak{D}}_{\boldsymbol{\pi}_{\text{old}}}(\boldsymbol{\pi}_{\text{new}}|s) \geq \mathbb{E}_{\mathbf{a}\sim\boldsymbol{\pi}_{\text{old}}}\big[Q_{\boldsymbol{\pi}_{\text{old}}}(s,\mathbf{a})\big] - \widetilde{\mathfrak{D}}_{\boldsymbol{\pi}_{\text{old}}}(\boldsymbol{\pi}_{\text{old}}|s), \forall s \in \mathcal{S}. \quad (23)
$$

We use it to prove the claim as follows,

$$
\begin{aligned}
V_{\boldsymbol{\pi}_{\text{new}}}(s) &= \mathbb{E}_{\mathbf{a}\sim\boldsymbol{\pi}_{\text{new}}}\big[Q_{\boldsymbol{\pi}_{\text{new}}}(s,\mathbf{a})\big] \\
&= \mathbb{E}_{\mathbf{a}\sim\boldsymbol{\pi}_{\text{new}}}\big[Q_{\boldsymbol{\pi}_{\text{old}}}(s,\mathbf{a})\big] - \widetilde{\mathfrak{D}}_{\boldsymbol{\pi}_{\text{old}}}(\boldsymbol{\pi}_{\text{new}}|s) \\
&\quad + \widetilde{\mathfrak{D}}_{\boldsymbol{\pi}_{\text{old}}}(\boldsymbol{\pi}_{\text{new}}|s) + \mathbb{E}_{\mathbf{a}\sim\boldsymbol{\pi}_{\text{new}}}\big[Q_{\boldsymbol{\pi}_{\text{new}}}(s,\mathbf{a}) - Q_{\boldsymbol{\pi}_{\text{old}}}(s,\mathbf{a})\big], \\
&\quad \text{by Inequality (23)} \\
&\geq \mathbb{E}_{\mathbf{a}\sim\boldsymbol{\pi}_{\text{old}}}\big[Q_{\boldsymbol{\pi}_{\text{old}}}(s,\mathbf{a})\big] - \widetilde{\mathfrak{D}}_{\boldsymbol{\pi}_{\text{old}}}(\boldsymbol{\pi}_{\text{old}}|s) \\
&\quad + \widetilde{\mathfrak{D}}_{\boldsymbol{\pi}_{\text{old}}}(\boldsymbol{\pi}_{\text{new}}|s) + \mathbb{E}_{\mathbf{a}\sim\boldsymbol{\pi}_{\text{new}}}\big[Q_{\boldsymbol{\pi}_{\text{new}}}(s,\mathbf{a}) - Q_{\boldsymbol{\pi}_{\text{old}}}(s,\mathbf{a})\big], \\
&= V_{\pi_{\text{old}}}(s) + \widetilde{\mathfrak{D}}_{\boldsymbol{\pi}_{\text{old}}}(\boldsymbol{\pi}_{\text{new}}|s) + \mathbb{E}_{\mathbf{a}\sim\boldsymbol{\pi}_{\text{new}}}\big[Q_{\boldsymbol{\pi}_{\text{new}}}(s,\mathbf{a}) - Q_{\boldsymbol{\pi}_{\text{old}}}(s,\mathbf{a})\big] \\
&= V_{\pi_{\text{old}}}(s) + \widetilde{\mathfrak{D}}_{\boldsymbol{\pi}_{\text{old}}}(\boldsymbol{\pi}_{\text{new}}|s) + \mathbb{E}_{\mathbf{a}\sim\boldsymbol{\pi}_{\text{new}},\mathrm{s}'\sim P}\big[r(s,\mathbf{a}) + \gamma V_{\boldsymbol{\pi}_{\text{new}}}(\mathrm{s}') - r(s,\mathbf{a}) - \gamma V_{\boldsymbol{\pi}_{\text{old}}}(\mathrm{s}')\big] \\
&= V_{\pi_{\text{old}}}(s) + \widetilde{\mathfrak{D}}_{\boldsymbol{\pi}_{\text{old}}}(\boldsymbol{\pi}_{\text{new}}|s) + \gamma\mathbb{E}_{\mathbf{a}\sim\boldsymbol{\pi}_{\text{new}},\mathrm{s}'\sim P}\big[V_{\boldsymbol{\pi}_{\text{new}}}(\mathrm{s}') - V_{\boldsymbol{\pi}_{\text{old}}}(\mathrm{s}')\big] \\
&\geq V_{\pi_{\text{old}}}(s) + \gamma \inf_{s'}\big[V_{\boldsymbol{\pi}_{\text{new}}}(s') - V_{\boldsymbol{\pi}_{\text{old}}}(s')\big].
\end{aligned}
$$

Hence $V_{\boldsymbol{\pi}_{\text{new}}}(s) - V_{\pi_{\text{old}}}(s) \geq \gamma \inf_{s'}\big[V_{\boldsymbol{\pi}_{\text{new}}}(s') - V_{\boldsymbol{\pi}_{\text{old}}}(s')\big]$.

Taking infimum over s and simplifying

$(1-\gamma)\inf_{s}\big[V_{\boldsymbol{\pi}_{\text{new}}}(s) - V_{\boldsymbol{\pi}_{\text{old}}}(s)\big] \geq 0.$

Therefore, $\inf_s\big[V_{\boldsymbol{\pi}_{\text{new}}}(s) - V_{\boldsymbol{\pi}_{\text{old}}}(s)\big] \geq 0$, which proves the lemma. ■

Appendix E. Proof of Theorem 14

Lemma 23 *Suppose an agent i_m maximises the expected HAMO*

$$\pi^{i_m}_{new} = \underset{\pi^{i_m} \in \mathcal{U}^{i_m}_{\boldsymbol{\pi}_{old}}(\pi^{i_m}_{old})}{\arg\max} \mathbb{E}_{\mathrm{s}\sim\beta_{\boldsymbol{\pi}_{old}}}\Big[\big[\mathcal{M}^{(\pi^{i_m})}_{\mathfrak{D}^{i_m},\boldsymbol{\pi}^{i_{1:m-1}}_{new}} A_{\boldsymbol{\pi}_{old}}\big](\mathrm{s})\Big]. \tag{24}$$

Then, for every state $s \in \mathcal{S}$

$$\big[\mathcal{M}^{(\pi^{i_m}_{new})}_{\mathfrak{D}^{i_m},\boldsymbol{\pi}^{i_{1:m-1}}_{new}} A_{\boldsymbol{\pi}_{old}}\big](s) \geq \big[\mathcal{M}^{(\pi^{i_m}_{old})}_{\mathfrak{D}^{i_m},\boldsymbol{\pi}^{i_{1:m-1}}_{new}} A_{\boldsymbol{\pi}_{old}}\big](s).$$

Proof We will prove this statement by contradiction. Suppose that there exists $s_0 \in \mathcal{S}$ such that

$$\big[\mathcal{M}^{(\pi^{i_m}_{\mathrm{new}})}_{\mathfrak{D}^{i_m},\boldsymbol{\pi}^{i_{1:m-1}}_{\mathrm{new}}} A_{\boldsymbol{\pi}_{\mathrm{old}}}\big](s_0) < \big[\mathcal{M}^{(\pi^{i_m}_{\mathrm{old}})}_{\mathfrak{D}^{i_m},\boldsymbol{\pi}^{i_{1:m-1}}_{\mathrm{new}}} A_{\boldsymbol{\pi}_{\mathrm{old}}}\big](s_0). \tag{25}$$

Let us define the following policy $\hat{\pi}^{i_m}$.

$$\hat{\pi}^{i_m}(\cdot^{i_m}|s) = \begin{cases} \pi^{i_m}_{\mathrm{old}}(\cdot^{i_m}|s), \text{ at } s = s_0 \\ \pi^{i_m}_{\mathrm{new}}(\cdot^{i_m}|s), \text{ at } s \neq s_0 \end{cases}$$

Note that $\hat{\pi}^{i_m}$ is (weakly) closer to $\pi^{i_m}_{\mathrm{old}}$ than $\pi^{i_m}_{\mathrm{new}}$ at s_0, and at the same distance at other states. Together with $\pi^{i_m}_{\mathrm{new}} \in \mathcal{U}^{i_m}_{\boldsymbol{\pi}_{\mathrm{old}}}(\pi^{i_m}_{\mathrm{old}})$, this implies that $\hat{\pi}^{i_m} \in \mathcal{U}^{i_m}_{\boldsymbol{\pi}_{\mathrm{old}}}(\pi^{i_m}_{\mathrm{old}})$. Further,

$$\begin{aligned} &\mathbb{E}_{\mathrm{s}\sim\beta_{\boldsymbol{\pi}_{\mathrm{old}}}}\Big[\big[\mathcal{M}^{(\hat{\pi}^{i_m})}_{\mathfrak{D}^{i_m},\boldsymbol{\pi}^{i_{1:m-1}}_{\mathrm{new}}} A_{\boldsymbol{\pi}_{\mathrm{old}}}\big](\mathrm{s})\Big] - \mathbb{E}_{\mathrm{s}\sim\beta_{\boldsymbol{\pi}_{\mathrm{old}}}}\Big[\big[\mathcal{M}^{(\pi^{i_m}_{\mathrm{new}})}_{\mathfrak{D}^{i_m},\boldsymbol{\pi}^{i_{1:m-1}}_{\mathrm{new}}} A_{\boldsymbol{\pi}_{\mathrm{old}}}\big](\mathrm{s})\Big] \\ &= \beta_{\boldsymbol{\pi}_{\mathrm{old}}}(s_0)\big(\big[\mathcal{M}^{(\pi^{i_m}_{\mathrm{old}})}_{\mathfrak{D}^{i_m},\boldsymbol{\pi}^{i_{1:m-1}}_{\mathrm{new}}} A_{\boldsymbol{\pi}_{\mathrm{old}}}\big](s_0) - \big[\mathcal{M}^{(\pi^{i_m}_{\mathrm{new}})}_{\mathfrak{D}^{i_m},\boldsymbol{\pi}^{i_{1:m-1}}_{\mathrm{new}}} A_{\boldsymbol{\pi}_{\mathrm{old}}}\big](s_0)\big) > 0. \end{aligned}$$

The above contradicts $\pi^{i_m}_{\mathrm{new}}$ as being the argmax of Inequality (25), as $\hat{\pi}^{i_m}$ is strictly better. The contradiction finishes the proof. ■

Theorem 14 (The Fundamental Theorem of Heterogeneous-Agent Mirror Learning) *Let, for every agent $i \in \mathcal{N}$, $\mathfrak{D}^i$ be a HADF, $\mathcal{U}^i$ be a neighbourhood operator, and let the sampling distributions $\beta_{\boldsymbol{\pi}}$ depend continuously on $\boldsymbol{\pi}$. Let $\boldsymbol{\pi}_0 \in \boldsymbol{\Pi}$, and the sequence of joint policies $(\boldsymbol{\pi}_k)^{\infty}_{k=0}$ be obtained by a HAML algorithm induced by $\mathfrak{D}^i$,$\mathcal{U}^i$,$\forall i \in \mathcal{N}$, and $\beta_{\boldsymbol{\pi}}$. Then, the joint policies induced by the algorithm enjoy the following list of properties*

1. *Attain the monotonic improvement property,*

$$J(\boldsymbol{\pi}_{k+1}) \;\geq\; J(\boldsymbol{\pi}_k),$$

2. *Their value functions converge to a Nash value function V^{NE}*

$$\lim_{k\to\infty} V_{\boldsymbol{\pi}_k} = V^{NE},$$

3. *Their expected returns converge to a Nash return,*

$$\lim_{k\to\infty} J(\boldsymbol{\pi}_k) = J^{NE},$$

4. *Their ω-limit set consists of Nash equilibria.*

Proof

Proof of Property 1. It follows from combining Lemmas 13 & 23.

Proof of Properties 2, 3 & 4.

Step 1: convergence of the value function. By Lemma 13, we have that $V_{\boldsymbol{\pi}_k}(s) \leq V_{\boldsymbol{\pi}_{k+1}}(s),\ \forall s \in \mathcal{S}$, and that the value function is upper-bounded by $V_{\max}$. Hence, the sequence of value functions $(V_{\boldsymbol{\pi}_k})_{k\in\mathbb{N}}$ converges. We denote its limit by V.

Step 2: characterisation of limit points. As the joint policy space $\boldsymbol{\Pi}$ is bounded, by Bolzano-Weierstrass theorem, we know that the sequence $(\boldsymbol{\pi}_k)_{k\in\mathbb{N}}$ has a convergent subsequence. Therefore, it has at least one limit point policy. Let $\bar{\boldsymbol{\pi}}$ be such a limit point. We introduce an auxiliary notation: for a joint policy $\boldsymbol{\pi}$ and a permutation $i_{1:n}$, let $\mathrm{HU}(\boldsymbol{\pi}, i_{1:n})$ be a joint policy obtained by a HAML update from $\boldsymbol{\pi}$ along the permutation $i_{1:n}$.

Claim: For any permutation $z_{1:n} \in \mathrm{Sym}(n)$,

$$\bar{\boldsymbol{\pi}} = \mathrm{HU}(\bar{\boldsymbol{\pi}}, z_{1:n}). \tag{26}$$

Proof of Claim. Let $\hat{\boldsymbol{\pi}} = \mathrm{HU}(\bar{\boldsymbol{\pi}}, z_{1:n}) \neq \bar{\boldsymbol{\pi}}$ and $(\boldsymbol{\pi}_{k_r})_{r\in\mathbb{N}}$ be a subsequence converging to $\bar{\boldsymbol{\pi}}$. Let us recall that the limit value function is unique and denoted as V. Writing $\mathbb{E}_{i^{0:\infty}_{1:n}}[\cdot]$ for the expectation operator under the stochastic process $(i^k_{1:n})_{k\in\mathbb{N}}$ of update orders, for a state $s \in \mathcal{S}$, we have

$$\begin{aligned}
0 &= \lim_{r\to\infty} \mathbb{E}_{i^{0:\infty}_{1:n}}\big[V_{\boldsymbol{\pi}_{k_r+1}}(s) - V_{\boldsymbol{\pi}_{k_r}}(s)\big] \\
&\quad \text{as every choice of permutation improves the value function} \\
&\geq \lim_{r\to\infty} \mathrm{P}(i^{k_r}_{1:n} = z_{1:n})\big[V_{\mathrm{HU}(\boldsymbol{\pi}_{k_r}, z_{1:n})}(s) - V_{\boldsymbol{\pi}_{k_r}}(s)\big] \\
&= p(z_{1:n}) \lim_{r\to\infty} \big[V_{\mathrm{HU}(\boldsymbol{\pi}_{k_r}, z_{1:n})}(s) - V_{\boldsymbol{\pi}_{k_r}}(s)\big].
\end{aligned}$$

By the continuity of the expected HAMO , we obtain that the first component of $\mathrm{HU}(\boldsymbol{\pi}_{k_r}, z_{1:n})$, which is $\pi^{z_1}_{k_r+1}$, is continuous in $\boldsymbol{\pi}_{k_r}$ by Berge's Maximum Theorem (Ausubel and Deneckere, 1993). Applying this argument recursively for $z_2, \ldots, z_n$, we have that $\mathrm{HU}(\boldsymbol{\pi}_{k_r}, z_{1:n})$ is continuous in $\boldsymbol{\pi}_{k_r}$. Hence, as $\boldsymbol{\pi}_{k_r}$ converges to $\bar{\boldsymbol{\pi}}$, its HU converges to the HU of $\bar{\boldsymbol{\pi}}$, which is $\hat{\boldsymbol{\pi}}$. Hence, we continue writing the above derivation as

$$= p(z_{1:n})\big[V_{\hat{\boldsymbol{\pi}}}(s) - V_{\bar{\boldsymbol{\pi}}}(s)\big] \geq 0, \text{ by Lemma 13.}$$

As s was arbitrary, the state-value function of $\hat{\boldsymbol{\pi}}$ is the same as that of $\boldsymbol{\pi}$: $V_{\hat{\boldsymbol{\pi}}} = V_{\boldsymbol{\pi}}$, by the Bellman equation (Sutton and Barto, 2018): $Q(s, \boldsymbol{a}) = r(s, \boldsymbol{a}) + \gamma \mathbb{E} V(\mathrm{s}')$, this also implies that their state-value and advantage functions are the same: $Q_{\hat{\boldsymbol{\pi}}} = Q_{\bar{\boldsymbol{\pi}}}$ and $A_{\hat{\boldsymbol{\pi}}} = A_{\bar{\boldsymbol{\pi}}}$. Let m be the smallest integer such that $\hat{\pi}^{z_m} \neq \bar{\pi}^{z_m}$. This means that $\hat{\pi}^{z_m}$ achieves a greater

expected HAMO than $\bar{\pi}^{z_m}$, for which it is zero. Hence,

$$
\begin{aligned}
0 &< \mathbb{E}_{\mathrm{s}\sim\beta_{\boldsymbol{\pi}}}\Big[\big[\mathcal{M}^{(\hat{\pi}^{z_m})}_{\mathfrak{D}^{z_m},\bar{\boldsymbol{\pi}}^{z_{1:m-1}}} A_{\bar{\boldsymbol{\pi}}}\big](s)\Big] \\
&= \mathbb{E}_{\mathrm{s}\sim\beta_{\boldsymbol{\pi}}}\Big[\mathbb{E}_{\mathbf{a}^{z_{1:m}}\sim\bar{\boldsymbol{\pi}}^{z_{1:m-1}},\mathrm{a}^{z_m}\sim\hat{\pi}^{z_m}}\big[A^{z_m}_{\bar{\boldsymbol{\pi}}}(s,\mathbf{a}^{z_{1:m-1}},\mathrm{a}^{z_m})\big] - \mathfrak{D}^{z_m}_{\boldsymbol{\pi}}(\hat{\pi}^{z_m}|s,\bar{\boldsymbol{\pi}}^{z_{1:m-1}})\Big] \\
&= \mathbb{E}_{\mathrm{s}\sim\beta_{\boldsymbol{\pi}}}\Big[\mathbb{E}_{\mathbf{a}^{z_{1:m}}\sim\bar{\boldsymbol{\pi}}^{z_{1:m-1}},\mathrm{a}^{z_m}\sim\hat{\pi}^{z_m}}\big[A^{z_m}_{\hat{\boldsymbol{\pi}}}(s,\mathbf{a}^{z_{1:m-1}},\mathrm{a}^{z_m})\big] - \mathfrak{D}^{z_m}_{\boldsymbol{\pi}}(\hat{\pi}^{z_m}|s,\bar{\boldsymbol{\pi}}^{z_{1:m-1}})\Big] \\
&\text{and as the expected value of the multi-agent advantage function is zero} \\
&= \mathbb{E}_{\mathrm{s}\sim\beta_{\boldsymbol{\pi}}}\Big[-\mathfrak{D}^{z_m}_{\boldsymbol{\pi}}(\hat{\pi}^{z_m}|s,\bar{\boldsymbol{\pi}}^{z_{1:m-1}})\Big] \leq 0.
\end{aligned}
$$

This is a contradiction, and so the claim in Equation (26) is proved, and the Step 2 is finished.

Step 3: dropping the HADF. Consider an arbitrary limit point joint policy $\bar{\boldsymbol{\pi}}$. By Step 2, for any permutation $i_{1:n}$, considering the first component of the HU,

$$
\begin{aligned}
\bar{\pi}^{i_1} &= \underset{\pi^{i_1}\in\mathcal{U}^{i_1}_{\bar{\boldsymbol{\pi}}}(\bar{\pi}^{i_1})}{\arg\max}\ \mathbb{E}_{\mathrm{s}\sim\beta_{\bar{\boldsymbol{\pi}}}}\Big[\big[\mathcal{M}^{(\pi^{i_1})}_{\mathfrak{D}^{i_1}} A_{\bar{\boldsymbol{\pi}}}\big](\mathrm{s})\Big] \\
&= \underset{\pi^{i_1}\in\mathcal{U}^{i_1}_{\bar{\boldsymbol{\pi}}}(\bar{\pi}^{i_1})}{\arg\max}\ \mathbb{E}_{\mathrm{s}\sim\beta_{\bar{\boldsymbol{\pi}}}}\Big[\mathbb{E}_{\mathrm{a}^{i_1}\sim\pi^{i_1}}\big[A^{i_1}_{\bar{\boldsymbol{\pi}}}(\mathrm{s},\mathrm{a}^{i_1})\big] - \mathfrak{D}^{i_1}_{\bar{\boldsymbol{\pi}}}(\pi^{i_1}|\mathrm{s})\Big].
\end{aligned} \tag{27}
$$

As the HADF is non-negative, and at $\pi^{i_1}=\bar{\pi}^{i_1}$ its value and of its all Gâteaux derivatives are zero, it follows by Step 3 of Theorem 1 of Kuba et al. (2022b) that for every $s\in\mathcal{S}$,

$$
\bar{\pi}^{i_1}(\cdot^{i_1}|s) = \underset{\pi^{i_1}\in\mathcal{P}(\mathcal{A}^{i_1})}{\arg\max}\ \mathbb{E}_{\mathrm{a}^{i_1}\sim\pi^{i_1}}\big[Q^{i_1}_{\bar{\boldsymbol{\pi}}}(s,\mathrm{a}^{i_1})\big].
$$

Step 4: Nash equilibrium. We have proved that $\bar{\boldsymbol{\pi}}$ satisfies

$$
\begin{aligned}
\bar{\pi}^{i}(\cdot^{i}|s) &= \underset{\pi^{i}(\cdot^{i}|s)\in\mathcal{P}(\mathcal{A}^{i})}{\arg\max}\ \mathbb{E}_{\mathrm{a}^{i}\sim\pi^{i}}\big[Q^{i}_{\bar{\boldsymbol{\pi}}}(s,\mathrm{a}^{i})\big] \\
&= \underset{\pi^{i}(\cdot^{i}|s)\in\mathcal{P}(\mathcal{A}^{i})}{\arg\max}\ \mathbb{E}_{\mathrm{a}^{i}\sim\pi^{i},\mathbf{a}^{-i}\sim\bar{\boldsymbol{\pi}}^{-i}}\big[Q_{\bar{\boldsymbol{\pi}}}(s,\mathbf{a})\big],\ \forall i\in\mathcal{N}, s\in\mathcal{S}.
\end{aligned}
$$

Hence, by considering $\bar{\boldsymbol{\pi}}^{-i}$ fixed, we see that $\bar{\pi}^{i}$ satisfies the condition for the optimal policy Sutton and Barto (2018), and hence

$$
\bar{\pi}^{i} = \underset{\pi^{i}\in\Pi^{i}}{\arg\max}\, J(\pi^{i},\bar{\boldsymbol{\pi}}^{-i}).
$$

Thus, $\bar{\boldsymbol{\pi}}$ is a Nash equilibrium. Lastly, this implies that the value function corresponds to a Nash value function V^{NE}, the return corresponds to a Nash return J^{NE}. ■

Appendix F. Casting HAPPO as HAML

The maximisation objective of agent i_m in HAPPO is

$$\mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}_{\text{old}}},\mathbf{a}^{i_{1:m-1}}\sim\boldsymbol{\pi}_{\text{new}}^{i_{1:m-1}},\mathrm{a}^{i_m}\sim\pi_{\text{old}}^{i_m}}\Big[\min\Big(\mathrm{r}(\bar{\pi}^{i_m})A^{i_{1:m}}_{\boldsymbol{\pi}_{\text{old}}}(\mathrm{s},\mathbf{a}^{i_{1:m}}),\mathrm{clip}\big(\mathrm{r}(\bar{\pi}^{i_m}),1\pm\epsilon\big)A^{i_{1:m}}_{\boldsymbol{\pi}_{\text{old}}}(\mathrm{s},\mathbf{a}^{i_{1:m}})\Big)\Big].$$

Fixing s and $\boldsymbol{a}^{i_{1:m-1}}$, we can rewrite it as

$$\begin{aligned}&\mathbb{E}_{\mathrm{a}^{i_m}\sim\bar{\pi}^{i_m}}\big[A^{i_{1:m}}_{\boldsymbol{\pi}_{\text{old}}}(\mathrm{s},\boldsymbol{a}^{i_{1:m-1}},\mathrm{a}^{i_m})\big]-\mathbb{E}_{\mathrm{a}^{i_m}\sim\pi_{\text{old}}^{i_m}}\Big[\mathrm{r}(\bar{\pi}^{i_m})A^{i_{1:m}}_{\boldsymbol{\pi}_{\text{old}}}(s,\boldsymbol{a}^{i_{1:m-1}},\mathrm{a}^{i_m})\\&-\min\Big(\mathrm{r}(\bar{\pi}^{i_m})A^{i_{1:m}}_{\boldsymbol{\pi}_{\text{old}}}(\mathrm{s},\boldsymbol{a}^{i_{1:m-1}},\mathrm{a}^{i_m}),\mathrm{clip}\big(\mathrm{r}(\bar{\pi}^{i_m}),1\pm\epsilon\big)A^{i_{1:m}}_{\boldsymbol{\pi}_{\text{old}}}(s,\boldsymbol{a}^{i_{1:m-1}},\mathrm{a}^{i_m})\Big)\Big].\end{aligned}$$

By the multi-agent advantage decomposition,

$$\begin{aligned}&\mathbb{E}_{\mathrm{a}^{i_m}\sim\bar{\pi}^{i_m}}\big[A^{i_{1:m}}_{\boldsymbol{\pi}_{\text{old}}}(\mathrm{s},\boldsymbol{a}^{i_{1:m-1}},\mathrm{a}^{i_m})\big]\\&=A^{i_{1:m-1}}_{\boldsymbol{\pi}_{\text{old}}}(s,\boldsymbol{a}^{i_{1:m-1}})+\mathbb{E}_{\mathrm{a}^{i_m}\sim\bar{\pi}^{i_m}}\big[A^{i_m}_{\boldsymbol{\pi}_{\text{old}}}(\mathrm{s},\boldsymbol{a}^{i_{1:m-1}},\mathrm{a}^{i_m})\big].\end{aligned}$$

Hence, the presence of the joint advantage of agents $i_{1:m}$ is equivalent to the multi-agent advantage of i_m given $\boldsymbol{a}^{i_{1:m-1}}$ that appears in HAMO, since the term $A^{i_{1:m-1}}_{\boldsymbol{\pi}_{\text{old}}}(s,\boldsymbol{a}^{i_{1:m-1}})$ cancels out with $-1\cdot M^{i_{1:m}}(\mathrm{s},\mathbf{a})$ of Equation 10 that we drop due to its zero gradient. Hence, we only need to show that that the subtracted term is an HADF. Firstly, we change min into max with the identity $-\min f(x)=\max[-f(x)]$.

$$\begin{aligned}&\mathbb{E}_{\mathrm{a}^{i_m}\sim\pi_{\text{old}}^{i_m}}\Big[\mathrm{r}(\bar{\pi}^{i_m})A^{i_{1:m}}_{\boldsymbol{\pi}_{\text{old}}}(s,\boldsymbol{a}^{i_{1:m-1}},\mathrm{a}^{i_m})\\&+\max\Big(-\mathrm{r}(\bar{\pi}^{i_m})A^{i_{1:m}}_{\boldsymbol{\pi}_{\text{old}}}(\mathrm{s},\boldsymbol{a}^{i_{1:m-1}},\mathrm{a}^{i_m}),-\mathrm{clip}\big(\mathrm{r}(\bar{\pi}^{i_m}),1\pm\epsilon\big)A^{i_{1:m}}_{\boldsymbol{\pi}_{\text{old}}}(s,\boldsymbol{a}^{i_{1:m-1}},\mathrm{a}^{i_m})\Big)\Big]\end{aligned}$$

which we then simplify

$$\begin{aligned}&\mathbb{E}_{\mathrm{a}^{i_m}\sim\pi_{\text{old}}^{i_m}}\Big[\max\Big(0,\big[\mathrm{r}(\bar{\pi}^{i_m})-\mathrm{clip}\big(\mathrm{r}(\bar{\pi}^{i_m}),1\pm\epsilon\big)\big]A^{i_{1:m}}_{\boldsymbol{\pi}_{\text{old}}}(s,\boldsymbol{a}^{i_{1:m-1}},\mathrm{a}^{i_m})\Big)\Big]\\&=\mathbb{E}_{\mathrm{a}^{i_m}\sim\pi_{\text{old}}^{i_m}}\Big[\mathrm{ReLU}\Big(\big[\mathrm{r}(\bar{\pi}^{i_m})-\mathrm{clip}\big(\mathrm{r}(\bar{\pi}^{i_m}),1\pm\epsilon\big)\big]A^{i_{1:m}}_{\boldsymbol{\pi}_{\text{old}}}(s,\boldsymbol{a}^{i_{1:m-1}},\mathrm{a}^{i_m})\Big)\Big].\end{aligned}$$

As discussed in the main body of the paper, this is an HADF.

Appendix G. Algorithms

Algorithm 5: HAA2C

Input: stepsize α, batch size B, number of: agents n, episodes K, steps per episode T, mini-epochs e;

Initialize: the critic network: ϕ, the policy networks: $\{\theta^i\}_{i\in\mathcal{N}}$, replay buffer $\mathcal{B}$;

for $k = 0, 1, \ldots, K-1$ **do**

Collect a set of trajectories by letting the agents act according to their policies, $\mathrm{a}^i \sim \pi^i_{\theta^i}(\cdot^i|\mathrm{o}^i)$;

Push transitions $\{(s_t, o^i_t, a^i_t, r_t, s_{t+1}, o^i_{t+1}), \forall i \in \mathcal{N}, t \in T\}$ into $\mathcal{B}$;

Sample a random minibatch of B transitions from $\mathcal{B}$;

Estimate the returns R and the advantage function, $\hat{A}(\mathrm{s}, \mathbf{a})$, using $\hat{V}_\phi$ and GAE;

Draw a permutation of agents $i_{1:n}$ at random;

Set $M^{i_1}(\mathrm{s}, \mathbf{a}) = \hat{A}(\mathrm{s}, \mathbf{a})$;

for *agent* $i_m = i_1, \ldots, i_n$ **do**

Set $\pi_0^{i_m}(\mathrm{a}^{i_m}|\mathrm{o}^{i_m}) = \pi^{i_m}_{\theta^{i_m}}(\mathrm{a}^{i_m}|\mathrm{o}^{i_m})$;

for *mini-epoch*$= 1, \ldots, e$ **do**

Compute agent i_m's policy gradient

$$\mathbf{g}^{i_m} = \nabla_{\theta^{i_m}} \frac{1}{B} \sum_{b=1}^{B} M^{i_m}(s_b, \boldsymbol{a}_b) \frac{\pi^{i_m}_{\theta^{i_m}}(a^{i_m}_b|o^{i_m}_b)}{\pi^{i_m}_0(a^{i_m}_b|o^{i_m}_b)}.$$

Update agent i_m's policy by

$$\theta^{i_m} = \theta^{i_m} + \alpha \mathbf{g}^{i_m}.$$

Compute $M^{i_{m+1}}(\mathrm{s}, \mathbf{a}) = \frac{\pi^{i_m}_{\theta^{i_m}}(\mathrm{a}^{i_m}|\mathrm{o}^{i_m})}{\pi^{i_m}_0(\mathrm{a}^{i_m}|\mathrm{o}^{i_m})} M^{i_m}(\mathrm{s}, \mathbf{a})$ //Unless $m = n$.

Update the critic by gradient descent on

$$\frac{1}{B} \sum_b \left(\hat{V}_\phi(s_b) - R_b\right)^2.$$

Discard ϕ. Deploy $\{\theta^i\}_{i\in\mathcal{N}}$ in execution;

Algorithm 6: HADDPG

Input: stepsize α, Polyak coefficient τ, batch size B, number of: agents n, episodes K, steps per episode T, mini-epochs e;

Initialize: the critic networks: ϕ and $\hat{\phi}$ and policy networks: $\{\theta^i\}_{i\in\mathcal{N}}$ and $\{\hat{\theta}^i\}_{i\in\mathcal{N}}$, replay buffer $\mathcal{B}$, random processes $\{\mathcal{X}^i\}_{i\in\mathcal{N}}$ for exploration;

for $k = 0, 1, \ldots, K-1$ **do**

Collect a set of transitions by letting the agents act according to their deterministic policies with the exploratory noise

$$a_t^i = \mu_{\theta^i}^i(o_t^i) + \mathcal{X}_t^i.$$

Push transitions $\{(s_t, o_t^i, a_t^i, r_t, s_{t+1}, o_{t+1}^i), \forall i \in \mathcal{N}, t \in T\}$ into $\mathcal{B}$;

Sample a random minibatch of B transitions from $\mathcal{B}$;

Compute the critic targets

$$y_t = r_t + \gamma Q_{\hat{\phi}}(s_{t+1}, \hat{\boldsymbol{a}}_{t+1}), \text{ where } \hat{\boldsymbol{a}}_{t+1} \text{ is sampled by } \{\hat{\theta}^i\}_{i\in\mathcal{N}}.$$

Update the critic by minimising the loss

$$\phi = \arg\min_\phi \tfrac{1}{B} \textstyle\sum_t \big(y_t - Q_\phi(s_t, \boldsymbol{a}_t)\big)^2.$$

Draw a permutation of agents $i_{1:n}$ at random;

for *agent* $i_m = i_1, \ldots, i_n$ **do**

Update agent i_m by solving

$$\theta_{\text{new}}^{i_m} = \arg\max_{\theta^{i_m}} \frac{1}{B} \sum_t Q_\phi\big(s_t, \boldsymbol{\mu}_{\boldsymbol{\theta}_{\text{new}}^{i_{1:m-1}}}^{i_{1:m-1}}(\boldsymbol{o}_t^{i_{1:m-1}}), \mu_{\theta^{i_m}}^{i_m}(o_t^{i_m}), \boldsymbol{\mu}_{\boldsymbol{\theta}_{\text{old}}^{i_{m+1:n}}}^{i_{m+1:n}}(\boldsymbol{o}_t^{i_{m+1:n}})\big).$$

with e mini-epochs of deterministic policy gradient ascent;

Update the target networks smoothly

$$\hat{\phi} = \tau\phi + (1-\tau)\hat{\phi}.$$

$$\hat{\theta}^i = \tau\theta^i + (1-\tau)\hat{\theta}^i.$$

Discard $\phi, \hat{\phi}$, and $\hat{\theta}^i, \forall i \in \mathcal{N}$. Deploy $\theta^i, \forall i \in \mathcal{N}$ in execution.

Algorithm 7: HATD3

Input: stepsize α, Polyak coefficient τ, batch size B, number of: agents n, episodes K, steps per episode T, mini-epochs e, target noise range c;

Initialize: the critic networks: ϕ_1, ϕ_2 and $\hat{\phi}_1, \hat{\phi}_2$ and policy networks: $\{\theta^i\}_{i\in\mathcal{N}}$ and $\{\hat{\theta}^i\}_{i\in\mathcal{N}}$, replay buffer $\mathcal{B}$, random processes $\{\mathcal{X}^i\}_{i\in\mathcal{N}}$ for exploration;

for $k = 0, 1, \ldots, K-1$ **do**

Collect a set of transitions by letting the agents act according to their deterministic policies with the exploratory noise

$$a_t^i = \mu_{\theta^i}^i(o_t^i) + \mathcal{X}_t^i.$$

Push transitions $\{(s_t, o_t^i, a_t^i, r_t, s_{t+1}, o_{t+1}^i), \forall i \in \mathcal{N}, t \in T\}$ into $\mathcal{B}$;

Sample a random minibatch of B transitions from $\mathcal{B}$;

Compute the critic targets

$$y_t = r_t + \gamma \min_{j=1,2} Q_{\hat{\phi}_j}(s_{t+1}, \hat{\boldsymbol{a}}_{t+1}), \text{ where}$$

$$\hat{a}_{t+1}^i = \text{clip}(\mu_{\hat{\theta}^i}^i(o_{t+1}^i) + \epsilon, a_{\text{Low}}^i, a_{\text{High}}^i),\ \epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c). \triangleright \text{Here } \mathcal{N} \text{ denotes Normal distribution.}$$

Update the critic by minimising the loss

$$\phi_j = \arg\min_{\phi_j} \frac{1}{B} \sum_t \left(y_t - Q_{\phi_j}(s_t, \boldsymbol{a}_t)\right)^2, j = 1, 2.$$

if $k \mod \textit{policy\_delay} = 0$ **then**

Draw a permutation of agents $i_{1:n}$ at random;

for *agent* $i_m = i_1, \ldots, i_n$ **do**

Update agent i_m by solving

$$\theta_{\text{new}}^{i_m} = \arg\max_{\tilde{\theta}^{i_m}} \frac{1}{B} \sum_t Q_{\phi_1}\left(s_t, \boldsymbol{\mu}_{\boldsymbol{\theta}_{\text{new}}^{i_{1:m-1}}}^{i_{1:m-1}}(\boldsymbol{o}_t^{i_{1:m-1}}), \mu_{\tilde{\theta}^{i_m}}^{i_m}(o_t^{i_m}), \boldsymbol{\mu}_{\boldsymbol{\theta}_{\text{old}}^{i_{m+1:n}}}^{i_{m+1:n}}(\boldsymbol{o}_t^{i_{m+1:n}})\right).$$

with e mini-epochs of deterministic policy gradient ascent;

Update the target networks smoothly

$$\hat{\phi}_1 = \tau\phi_1 + (1-\tau)\hat{\phi}_1.$$

$$\hat{\phi}_2 = \tau\phi_2 + (1-\tau)\hat{\phi}_2.$$

$$\hat{\theta}^i = \tau\theta^i + (1-\tau)\hat{\theta}^i.$$

Discard $\phi_1, \phi_2, \hat{\phi}_1, \hat{\phi}_2$, and $\hat{\theta}^i, \forall i \in \mathcal{N}$. Deploy $\theta^i, \forall i \in \mathcal{N}$ in execution.

Appendix H. The Summary of HARL algorithms as Instances of HAML

Recap of HAML

- Definition of HAMO:

$$\big[\mathcal{M}^{(\pi^{i_m})}_{\mathfrak{D}^{i_m},\boldsymbol{\pi}^{i_{1:m-1}}_{k+1}} A_{\boldsymbol{\pi}_k}\big](s) \triangleq \mathbb{E}_{\mathbf{a}^{i_{1:m-1}}\sim\boldsymbol{\pi}^{i_{1:m-1}}_{k+1},\mathrm{a}^{i_m}\sim\pi^{i_m}}\Big[A^{i_m}_{\boldsymbol{\pi}_k}\left(s,\mathbf{a}^{i_{1:m-1}},\mathrm{a}^{i_m}\right)\Big] - \mathfrak{D}^{i_m}_{\boldsymbol{\pi}_k}\Big(\pi^{i_m}\big|s,\boldsymbol{\pi}^{i_{1:m-1}}_{k+1}\Big).$$

- Optimisation target: $\pi^{i_m}_{k+1} = \underset{\pi^{i_m}\in\mathcal{U}^{i_m}_{\boldsymbol{\pi}_k}(\pi^{i_m}_k)}{\arg\max} \mathbb{E}_{\mathrm{s}\sim\beta_{\boldsymbol{\pi}_k}}\Big[\big[\mathcal{M}^{(\pi^{i_m})}_{\mathfrak{D}^{i_m},\boldsymbol{\pi}^{i_{1:m-1}}_{k+1}} A_{\boldsymbol{\pi}_k}\big](\mathrm{s})\Big]$

HATRPO

$$\pi^{i_m}_{k+1} = \underset{\pi^{i_m}}{\arg\max}\, \mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}_k},\mathbf{a}^{i_{1:m-1}}\sim\boldsymbol{\pi}^{i_{1:m-1}}_{k+1},\mathrm{a}^{i_m}\sim\pi^{i_m}}\left[A^{i_m}_{\boldsymbol{\pi}_k}\left(\mathrm{s},\mathbf{a}^{i_{1:m-1}},\mathrm{a}^{i_m}\right)\right], \quad \text{subject to } \bar{\mathrm{D}}_{\mathrm{KL}}\left(\pi^{i_m}_k,\pi^{i_m}\right)\le\delta. \tag{28}$$

- Drift functional: HADF $\mathfrak{D}^{i_m}_{\boldsymbol{\pi}_k}\Big(\pi^{i_m}\big|s,\boldsymbol{\pi}^{i_{1:m-1}}_{k+1}\Big)\equiv 0$.

- Neighborhood operator:

$$\mathcal{U}^{i_m}_{\boldsymbol{\pi}_k}(\pi^{i_m}_k) = \Big\{\pi^{i_m}\in\Pi^{i_m}\ \Big|\ \mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}_k}}\Big[\mathrm{D}_{\mathrm{KL}}\big(\pi^{i_m}_k(\cdot|\mathrm{s}),\pi^{i_m}(\cdot|\mathrm{s})\big)\Big]\le\delta\Big\}.$$

- Sampling distribution: $\beta_{\boldsymbol{\pi}_k}=\rho_{\boldsymbol{\pi}_k}$.

HAPPO

$$\pi^{i_m}_{k+1} = \underset{\pi^{i_m}}{\arg\max}\, \mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}_k},\mathbf{a}^{i_{1:m-1}}\sim\boldsymbol{\pi}^{i_{1:m-1}}_{k+1},\mathrm{a}^{i_m}\sim\pi^{i_m}_k} \Big[\min\Big(\mathrm{r}(\pi^{i_m})A^{i_{1:m}}_{\boldsymbol{\pi}_k}(\mathrm{s},\mathbf{a}^{i_{1:m}}),\mathrm{clip}\big(\mathrm{r}(\pi^{i_m}),1\pm\epsilon\big)A^{i_{1:m}}_{\boldsymbol{\pi}_k}(\mathrm{s},\mathbf{a}^{i_{1:m}})\Big)\Big], \quad \text{where } \mathrm{r}(\pi^{i_m}) = \frac{\pi^{i_m}(\mathrm{a}^{i_m}|\mathrm{s})}{\pi^{i_m}_k(\mathrm{a}^{i_m}|\mathrm{s})}. \tag{29}$$

- Drift functional:

$$\mathfrak{D}^{i_m}_{\boldsymbol{\pi}_k}\Big(\pi^{i_m}\big|s,\boldsymbol{\pi}^{i_{1:m-1}}_{k+1}\Big) = \mathbb{E}_{\mathbf{a}^{i_{1:m-1}}\sim\boldsymbol{\pi}^{i_{1:m-1}}_{k+1},\mathrm{a}^{i_m}\sim\pi^{i_m}_k}\Big[\mathrm{ReLU}\big(\big[\mathrm{r}(\pi^{i_m})-\mathrm{clip}\big(\mathrm{r}(\pi^{i_m}),1\pm\epsilon\big)\big]A^{i_{1:m}}_{\boldsymbol{\pi}_k}(s,\mathbf{a}^{i_{1:m}})\big)\Big] \tag{30}$$

- Neighborhood operator: $\mathcal{U}^{i_m}_{\boldsymbol{\pi}_k}(\pi^{i_m}_k)\equiv\Pi^{i_m}$.

- Sampling distribution: $\beta_{\boldsymbol{\pi}_k}=\rho_{\boldsymbol{\pi}_k}$.

HAA2C

$$\pi_{k+1}^{i_m} = \arg\max_{\pi^{i_m}} \mathbb{E}_{\mathrm{s}\sim\rho_{\boldsymbol{\pi}_k}, \mathbf{a}^{i_{1:m-1}}\sim\boldsymbol{\pi}_{k+1}^{i_{1:m-1}}, \mathrm{a}^{i_m}\sim\pi^{i_m}} \left[A_{\boldsymbol{\pi}_k}^{i_m}\left(\mathrm{s}, \mathbf{a}^{i_{1:m-1}}, \mathrm{a}^{i_m}\right)\right] \tag{31}$$

- Drift functional: HADF $\mathfrak{D}_{\boldsymbol{\pi}_k}^{i_m}\left(\pi^{i_m}\middle|s, \boldsymbol{\pi}_{k+1}^{i_{1:m-1}}\right) \equiv 0$.
- Neighborhood operator: $\mathcal{U}_{\boldsymbol{\pi}_k}^{i_m}(\pi_k^{i_m}) \equiv \Pi^{i_m}$.
- Sampling distribution: $\beta_{\boldsymbol{\pi}_k} = \rho_{\boldsymbol{\pi}_k}$.

HADDPG & HATD3

$$\mu_{k+1}^{i_m} = \arg\max_{\mu^{i_m}} \mathbb{E}_{\mathrm{s}\sim\beta_{\boldsymbol{\mu}_k}}\left[Q_{\boldsymbol{\mu}_k}^{i_{1:m}}\left(\mathrm{s}, \boldsymbol{\mu}_{k+1}^{i_{1:m-1}}(\mathrm{s}), \mu^{i_m}(\mathrm{s})\right)\right], \tag{32}$$

- Drift functional: HADF $\mathfrak{D}_{\boldsymbol{\mu}_k}^{i_m}\left(\mu^{i_m}\middle|s, \boldsymbol{\mu}_{k+1}^{i_{1:m-1}}\right) \equiv 0$.
- Neighborhood operator:

$$\mathcal{U}_{\boldsymbol{\mu}_k}^{i_m}(\mu_k^{i_m}) \equiv \Pi^{i_m} \quad \text{(the deterministic policy space)}.$$

- Sampling distribution: $\beta_{\boldsymbol{\mu}_k}$ is a uniform distribution over the states in the off-policy replay buffer.

Appendix I. HAD3QN: A Pure Value-based Approximation to HADDPG

In this section, we propose HAD3QN, which is a pure value-based approximation of HADDPG. Corresponding to HADDPG where each agent learns to maximise the joint target given the previous agents' updates, HAD3QN models decentralised agents as individual Q networks that predict the centralised critic's output. In particular, the centralised critic's output is sequentially maximised for sequential learning. During execution, for each observation each agent chooses the action that maximises its individual Q network. We provide its pseudocode in Algorithm 8.

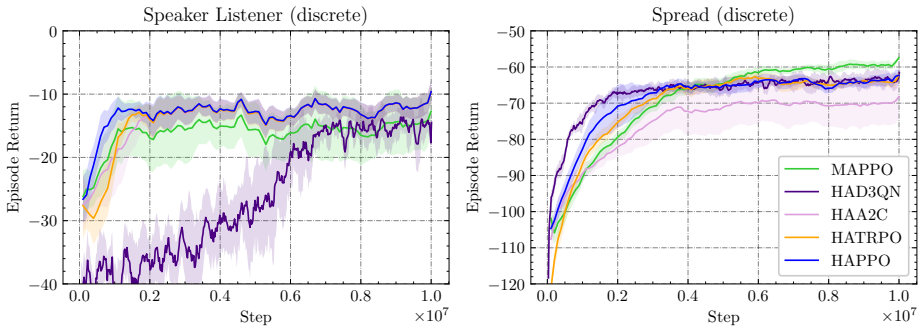


Figure 13: Average episode return of HAD3QN on Speaker Listener and Spread compared with existing methods.

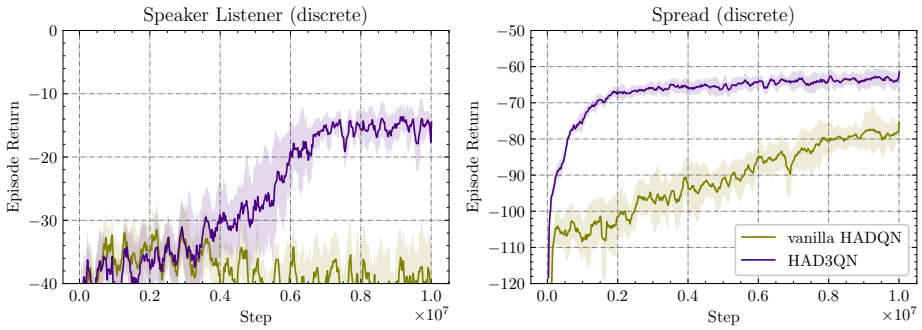


Figure 14: Ablation study on the effect of dueling network architecture in HAD3QN.

Empirically, we test it on the Speaker Listener and Spread task in MPE, and observe that HAD3QN is able to solve them within 10 million steps (Figure 13). Compared with the vanilla HADQN where dueling architecture is not utilised (Figure 14), we find that the dueling network architecture effectively improves learning efficiency and stability, and is crucial for HAD3QN to achieve higher return. The hyperparameters are reported in Section K.

However, we note that HAD3QN does not scale well as it suffers from the curse of dimensionality with the growing number of agents and increasing dimensionality of individual action space. This phenomenon is similar to what has been discussed in the DQN case in RL by Lillicrap et al. (2016). The purpose of proposing HAD3QN is not to refresh SOTA

methods, but to show that discretised approximation of HADDPG is also possible and it performs well on low-dimensional tasks. It also shows that our HARL framework allows direct extension of RL research results, in this case being the dueling network design, which is potentially powerful as the efforts to re-derive similar multi-agent results can be saved.

Appendix J. Additional Experiment Results

In this section, we present the learning curves of HAPPO, HATRPO, MAPPO, and QMIX across at least three seeds on ten SMAC maps and five SMACv2 maps in Figure 15.

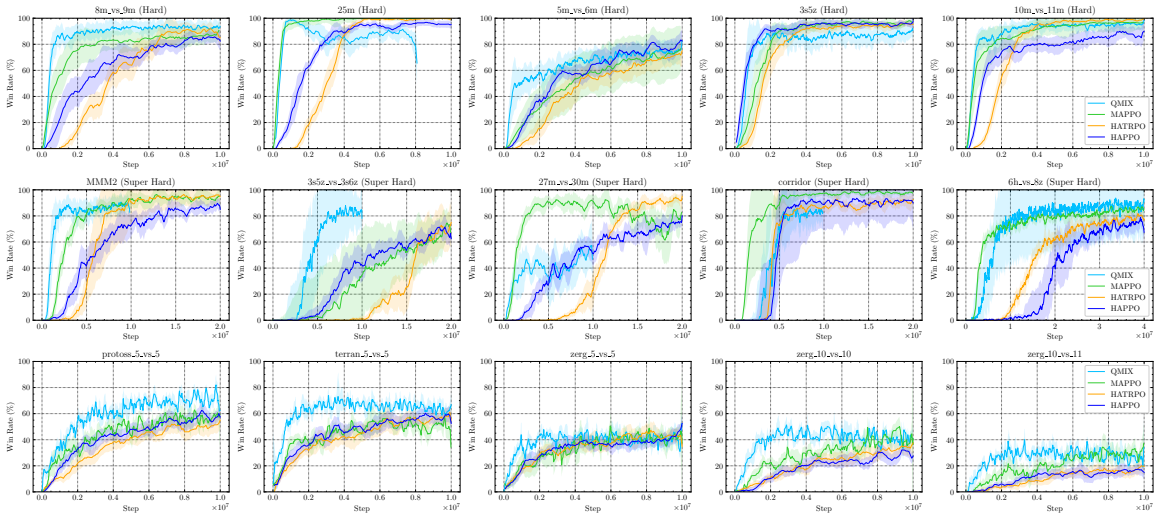


Figure 15: Comparisons of average win rate on SMAC and SMACv2. It should be noted that some of the QMIX experiments were terminated early if they had already converged, as observed in `MMM2`, `3s5z_vs_3s6z`, and `corridor`, or if the computational resources required were excessive, as observed in the case of `27m_vs_30m`. Specifically, running QMIX for a single seed for 20 million steps in `27m_vs_30m` would have necessitated more than 250 GB memory and 10 days, which exceeded the computational budget allocated for this study. Consequently, we executed the experiment for only 10 million steps.

Algorithm 8: HAD3QN

Input: stepsize α, Polyak coefficient τ, batch size B, exploration parameter ϵ, number of: agents n, episodes K, steps per episode T.

Initialize: global critic and target networks: ϕ, and $\hat{\phi}$, distributed critic and target networks: $\{\theta^i,\ \forall i \in \mathcal{N}\}$ and $\{\hat{\theta}^i,\ \forall i \in \mathcal{N}\}$, replay buffer $\mathcal{B}$.

for $k = 0, 1, \dots, K-1$ **do**

Collect a set of trajectories by letting the agents act ϵ-greedily with respect to the distributed critics

$$a_t^i = \begin{cases} \arg\max_{a^i} Q^i_{\theta^i}(o_t^i, a^i) & \text{with probability } 1-\epsilon \\ \text{random} & \text{with probability } \epsilon. \end{cases}$$

Push transitions $\{(s_t, o_t^i, a_t^i, r_t, s_{t+1}, o_{t+1}^i), \forall i \in \mathcal{N}, t \in T\}$ into $\mathcal{B}$.

Sample a random minibatch of B transitions from $\mathcal{B}$.

Compute the global target

$$y_t = r_t + \gamma \cdot Q_{\hat{\phi}}(s_{t+1}, \boldsymbol{a}_*),$$
$$\text{where } a_*^i = \arg\max_{a^i} Q^i_{\hat{\theta}^i}(o_{t+1}^i, a^i), \text{ for all } i \in \mathcal{N}.$$

Compute the global loss

$$L(\phi) = \tfrac{1}{B} \sum_{b=1}^{B} \big(Q_\phi(s_b, \boldsymbol{a}_b) - y_b\big)^2.$$

Update the critic parameters

$$\phi = \phi - \alpha \nabla_\phi L(\phi).$$

Draw a permutation of agents $i_{1:n}$ at random;

for *agent* $i_m = i_1, \dots, i_n$ **do**

Compute the local targets

$$y_t^{i_m} = Q_\phi(s_t, \boldsymbol{a}_*^{i_{1:m-1}}, \boldsymbol{a}_t^{-i_{1:m-1}}),$$
$$\text{where } a_*^{i_j} = \arg\max_{a^{i_j}} Q_\phi(s_t, \boldsymbol{a}_*^{i_{1:j-1}}, a^{i_j}, \boldsymbol{a}_t^{-i_{1:j}}), \text{ for } j < m.$$

Compute the agent's local loss

$$L(\theta^{i_m}) = \tfrac{1}{B} \sum_{b=1}^{B} \big(Q^{i_m}_{\theta^{i_m}}(o_b^{i_m}, a_b^{i_m}) - y_b^{i_m}\big)^2.$$

Update the critic parameters

$$\theta^{i_m} = \theta^{i_m} - \alpha \nabla_{\theta^{i_m}} L(\theta^{i_m}).$$

Update the target networks smoothly

$$\hat{\phi} = \tau\phi + (1-\tau)\hat{\phi}, \quad \hat{\theta}^i = \tau\theta^i + (1-\tau)\hat{\theta}^i.$$

Discard $\phi, \hat{\phi}$, and $\hat{\theta}^i, \forall i \in \mathcal{N}$. Deploy $\theta^i, \forall i \in \mathcal{N}$ in execution.

Appendix K. Hyperparameter Settings for Experiments

Before we report the hyperparameters used in the experiments, we would like to clarify the reporting conventions that we follow. Firstly, for simplicity and clarity reasons, we specify the network architecture to be MLP or RNN, but in configuration files the corresponding term is a boolean value `use_recurrent_policy` . The only difference between RNN network and MLP network is that the former has a GRU layer after the same MLP backbone, and the related configuration of this GRU layer is provided in Table 4. Secondly, the hyperparameters will only take effect when they are used. For example, the number of GRU layers is set to 1 across all environments, but it should only be considered when the network architecture is RNN; as another example, while we report `kl_threshold` in on-policy hyperparameter tables, it is only useful when HATRPO is applied. Finally, the `batch_size` reported for on-policy algorithms is calculated as the product of `n_rollout_threads` and `episode_length`.

K.1 Common Hyperparameters Across All Environments

In this part, we present the common hyperparameters used for on-policy algorithms in Table 4 and for off-policy algorithms in Table 5 across all environments.

Table 4: Common hyperparameters used for on-policy algorithms HAPPO, HATRPO, HAA2C, and MAPPO (when our MAPPO implementation is used) across all environments.

hyperparameters	value	hyperparameters	value
use valuenorm	True	use proper time limits	True
activation	ReLU	use feature normalization	True
initialization method	orthogonal	gain	0.01
use naive recurrent policy	False	num GRU layers	1
data chunk length	10	optim eps	$1e-5$
weight decay	0	std x coef	1
std y coef	0.5	use clipped value loss	True
value loss coef	1	use max grad norm	True
max grad norm	10.0	use GAE	True
GAE lambda	0.95	use huber loss	True
use policy active masks	True	huber delta	10.0
action aggregation	prod	ls step	10
accept ratio	0.5		

K.2 Multi-Agent Particle Environment (MPE)

In this part, we present the hyperparameters used in MPE tasks for HAPPO, HATRPO, HAA2C, and MAPPO in Table 6, for HADDPG, HATD3, MADDPG, and MATD3 in Table 7, and for HAD3QN in Table 8.

Table 5: Common hyperparameters used for off-policy algorithms HADDPG, HATD3, HAD3QN, MADDPG, and MATD3 across all environments.

hyperparameters	value	hyperparameters	value
proper time limits	True	warmup steps	1e4
activation	ReLU	final activation	Tanh
base activation	ReLU	dueling v activation	Hardswish
dueling a activation	Hardswish	buffer size	1e6
batch size	1000	polyak	0.005
epsilon	0.05	policy noise	0.2
noise clip	0.5		

Table 6: Common hyperparameters used for HAPPO, HATRPO, HAA2C, and MAPPO in the MPE domain.

hyperparameters	value	hyperparameters	value	hyperparameters	value
batch size	4000	linear lr decay	False	network	MLP
hidden sizes	[128, 128]	actor lr	$5e-4$	critic lr	$5e-4$
ppo epoch	5	critic epoch	5	a2c epoch	5
clip param	0.2	actor mini batch	1	critic mini batch	1
entropy coef	0.01	gamma	0.99	kl threshold	0.005
backtrack coeff	0.8				

Table 7: Common hyperparameters used for HADDPG, HATD3, MADDPG, and MATD3 in the MPE domain.

hyperparameters	value	hyperparameters	value	hyperparameters	value
rollout threads	20	train interval	50	update per train	1
linear lr decay	False	hidden sizes	[128, 128]	actor lr	$5e-4$
critic lr	$1e-3$	gamma	0.99	n step	1
policy update frequency	2				

Table 8: Common hyperparameters used for HAD3QN in the MPE domain.

hyperparameters	value	hyperparameters	value	hyperparameters	value
rollout threads	20	train interval	50	base hidden sizes	[128, 128]
linear lr decay	False	update per train	1	dueling v hidden sizes	[128]
actor lr	$5e-4$	critic lr	$1e-3$	dueling a hidden sizes	[128]
gamma	0.95	n step	1		

K.3 Multi-Agent MuJoCo (MAMuJoCo)

In this part, we report the hyperparameters used in MAMuJoCo tasks for HAPPO, HATRPO, HAA2C, and MAPPO in Table 9, 10, 11, and 12, and for HADDPG, HATD3, MADDPG, and MATD3 in Table 13, 14, 15, and 16.

Table 9: Common hyperparameters used for HAPPO, HATRPO, HAA2C, and MAPPO in the MAMuJoCo domain.

hyperparameters	value	hyperparameters	value	hyperparameters	value
batch size	4000	network	MLP	hidden sizes	[128, 128, 128]
gamma	0.99	backtrack coeff	0.8		

Table 10: Different hyperparameters used for HAPPO and MAPPO in the MAMuJoCo domain.

scenarios	linear lr decay	actor/critic lr	ppo/critic epoch	clip param	actor/critic mini batch	entropy coef
Ant 4x2	False	$5e-4$	5	0.1	1	0
HalfCheetah 2x3	False	$5e-4$	15	0.05	1	0.01
Hopper 3x1	True	$5e-4$	10	0.05	1	0
Walker 2x3	True	$1e-3$	5	0.05	2	0
Walker 6x1	False	$5e-4$	5	0.1	1	0.01
Humanoid 17x1	True	$5e-4$	5	0.1	1	0

Table 11: Different hyperparameters used for HATRPO in the MAMuJoCo domain.

scenarios	linear lr decay	critic lr	critic epoch	clip param	critic mini batch	kl threshold
Ant 4x2	False	$5e-4$	5	0.2	1	$5e-3$
HalfCheetah 2x3	False	$5e-4$	5	0.2	1	$1e-2$
Hopper 3x1	False	$5e-4$	5	0.2	1	$1e-3$
Walker 2x3	False	$5e-4$	5	0.2	1	$1e-2$
Walker 6x1	False	$5e-4$	5	0.2	1	$5e-3$

K.4 StarCraft Multi-Agent Challenge (SMAC)

In the SMAC domain, for MAPPO and QMIX baselines we adopt the implementation and tuned hyperparameters reported in the MAPPO paper. Here we report the hyperparameters for HAPPO and HATRPO in Table 17, 18, 19, and 20, which are kept comparable with the baselines for fairness purposes. The `state type` hyperparameter can take "EP" (for *Environment-Provided global state*) and "FP" (for *Featured-Pruned agent-specific global state*), as named by Yu et al. (2022).

Table 12: Different hyperparameters used for HAA2C in the MAMuJoCo domain.

scenarios	linear lr decay	actor/critic lr	a2c/critic epoch	clip param	actor/critic mini batch	entropy coef
Ant 4x2	True	$5e-4$	5	0.1	1	0
HalfCheetah 2x3	True	$5e-4$	5	0.1	1	0
Hopper 3x1	True	$1e-4$	3	0.1	1	0
Walker 2x3	True	$1e-4$	5	0.1	1	0
Walker 6x1	True	$1e-4$	5	0.1	1	0

Table 13: Common hyperparameters used for HADDPG and MADDPG in the MAMuJoCo domain.

hyperparameters	value	hyperparameters	value	hyperparameters	value
rollout threads	10	train interval	50	linear lr decay	False
hidden sizes	[256, 256]	actor lr	$5e-4$	critic lr	$1e-3$
gamma	0.99				

Table 14: Different hyperparameters used for HADDPG and MADDPG in the MAMuJoCo domain.

scenarios	update per train	exploration noise	n step
Ant 4x2	0.5	0.05	20
HalfCheetah 2x3	1	0.1	20
Hopper 3x1	1	0.1	20
Walker 2x3	1	0.1	10
Walker 6x1	1	0.1	20

Table 15: Common hyperparameters used for HATD3 and MATD3 in the MAMuJoCo domain.

hyperparameters	value	hyperparameters	value	hyperparameters	value
rollout threads	10	train interval	50	update per train	1
linear lr decay	False	hidden sizes	[256, 256]	actor lr	$5e-4$
critic lr	$1e-3$	gamma	0.99	exploration noise	0.1

Table 16: Different hyperparameters used for HATD3 and MATD3 in the MAMuJoCo domain.

scenarios	policy update frequency	n step
Ant 4x2	2	5
HalfCheetah 2x3	2	10
Hopper 3x1	2	5
Walker 2x3	8	20
Walker 6x1	2	25
Humanoid 17x1	2	5

Table 17: Common hyperparameters used for HAPPO in the SMAC domain.

hyperparameters	value	hyperparameters	value	hyperparameters	value
batch size	3200	linear lr decay	False	hidden sizes	[64, 64, 64]
actor lr	$5e-4$	critic lr	$5e-4$	entropy coef	0.01

Table 18: Different hyperparameters used for HAPPO in the SMAC domain.

Map	network	ppo/critic epoch	clip param	actor/critic mini batch	gamma	state type
8m_vs_9m	RNN	5	0.05	1	0.95	EP
25m	RNN	5	0.2	1	0.99	EP
5m_vs_6m	RNN	5	0.05	1	0.95	FP
3s5z	RNN	5	0.2	1	0.99	EP
10m_vs_11m	RNN	5	0.05	1	0.95	FP
MMM2	MLP	5	0.2	1	0.95	EP
3s5z_vs_3s6z	RNN	5	0.1	2	0.95	FP
27m_vs_30m	RNN	5	0.05	1	0.95	FP
6h_vs_8z	MLP	10	0.05	2	0.95	FP
corridor	MLP	5	0.2	1	0.99	FP

Table 19: Common hyperparameters used for HATRPO in the SMAC domain.

hyperparameters	value	hyperparameters	value	hyperparameters	value
batch size	3200	linear lr decay	False	hidden sizes	[64, 64, 64]
critic epoch	5	clip param	0.2	critic mini batch	1

Table 20: Different hyperparameters used for HATRPO in the SMAC domain.

Map	network	critic lr	gamma	kl threshold	backtrack coeff	state type
8m_vs_9m	MLP	$5e-4$	0.99	$5e-3$	0.5	FP
25m	RNN	$5e-4$	0.99	$1e-2$	0.5	EP
5m_vs_6m	RNN	$5e-4$	0.99	$1e-2$	0.5	FP
3s5z	MLP	$5e-4$	0.95	$1e-2$	0.5	EP
10m_vs_11m	MLP	$5e-4$	0.95	$5e-3$	0.5	FP
MMM2	MLP	$5e-4$	0.95	$6e-2$	0.5	EP
3s5z_vs_3s6z	MLP	$5e-4$	0.99	$5e-3$	0.5	FP
27m_vs_30m	RNN	$5e-4$	0.99	$1e-3$	0.8	FP
6h_vs_8z	MLP	$1e-3$	0.99	$1e-3$	0.8	FP
corridor	RNN	$5e-4$	0.99	$6e-2$	0.5	FP

K.5 SMACv2

In the SMACv2 domain, for MAPPO and QMIX baselines we adopt the implementation and tuned hyperparameters reported in Ellis et al. (2022). Here we report the hyperparameters for HAPPO and HATRPO in Table 21 and 22, which are kept comparable with the baselines for fairness purposes.

Table 21: Hyperparameters used for HAPPO in the SMACv2 domain.

hyperparameters	value	hyperparameters	value	hyperparameters	value
batch size	3200	linear lr decay	False	hidden sizes	[64]
network	RNN	ppo / critic epoch	5	clip param	0.05
actor lr	$5e-4$	critic lr	$5e-4$	entropy coef	0.01
gamma	0.99	actor / critic mini batch	2 for `terran_5_vs_5` and 1 otherwise		

Table 22: Hyperparameters used for HATRPO on all tasks in the SMACv2 domain.

hyperparameters	value	hyperparameters	value	hyperparameters	value
batch size	3200	linear lr decay	False	hidden sizes	[64]
network	RNN	critic lr	$5e-4$	critic epoch	5
clip param	0.2	critic mini batch	1	gamma	0.99
kl threshold	$5e-3$	backtrack coeff	0.5		

K.6 Google Research Football Environment (GRF)

In the GRF domain, for MAPPO and QMIX baselines we adopt the implementation and tuned hyperparameters reported in the MAPPO paper. Here we report the hyperparameters for HAPPO in Table 23 and 24, which are kept similar and comparable to the baselines for fairness purposes.

Table 23: Common hyperparameters used for HAPPO in the GRF domain.

hyperparameters	value	hyperparameters	value	hyperparameters	value
rollout threads	50	hidden sizes	[64, 64]	actor lr	$5e-4$
critic lr	$5e-4$	ppo epoch	15	critic epoch	15
clip param	0.2	actor mini batch	2	critic mini batch	2
entropy coef	0.01	gamma	0.99		

K.7 Bi-DexterousHands

In the Bi-DexterousHands domain, we use the PPO and MAPPO baselines implemented in the Bi-DexterousHands benchmark for comparison and for them we adopt the officially reported hyperparameters. Here we report the hyperparameters used for HAPPO in Table 25. As Bi-DexterousHands tasks are GPU-parallelised, we reload the configuration term `n_rollout_threads` with a meaning of number of parallel environments. Thus, `parallel envs` in Table 25 refers to `n_rollout_threads`.

Table 24: Different hyperparameters used for HAPPO in the GRF domain.

scenarios	network	episode length	linear lr decay
PS	RNN	200	True
RPS	MLP	200	False
3v.1	MLP	200	True
CA(easy)	MLP	200	True
CA(hard)	MLP	1000	True

Table 25: Common hyperparameters used for HAPPO in the Bi-DexterousHands domain.

hyperparameters	value	hyperparameters	value	hyperparameters	value
parallel envs	256	linear lr decay	False	network	MLP
hidden sizes	[256, 256, 256]	actor lr	$5e-4$	critic lr	$5e-4$
ppo epoch	5	critic epoch	5	clip param	0.2
actor mini batch	1	critic mini batch	1	entropy coef	0.01
gamma	0.95				

References

Johannes Ackermann, Volker Gabler, Takayuki Osa, and Masashi Sugiyama. Reducing overestimation bias in multi-agent domains using double centralized critics. *arXiv preprint arXiv:1910.01465*, 2019.

Carlos Alós-Ferrer and Nick Netzer. The logit-response dynamics. *Games and Economic Behavior*, 68(2):413–427, 2010.

Lawrence M Ausubel and Raymond J Deneckere. A generalized theorem of the maximum. *Economic Theory*, 3(1):99–107, 1993.

Tamer Başar and Geert Jan Olsder. *Dynamic noncooperative game theory*. SIAM, 1998.

Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4):819–840, 2002.

Dimitri Bertsekas. Multiagent rollout algorithms and reinforcement learning. *arXiv preprint arXiv:1910.00120*, 2019.

Jeancarlo Arguello Calvo and Ivana Dusparic. Heterogeneous multi-agent deep reinforcement learning for traffic lights control. In *AICS*, pages 2–13, 2018.

Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial informatics*, 9(1):427–438, 2012.

Yuanpei Chen, Yaodong Yang, Tianhao Wu, Shengjie Wang, Xidong Feng, Jiechuan Jiang, Zongqing Lu, Stephen Marcus McAleer, Hao Dong, and Song-Chun Zhu. Towards human-level bimanual dexterous manipulation with reinforcement learning. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL `https://openreview.net/forum?id=D29JbExncTP`.

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*, pages 1989–1998. PMLR, 2021.

Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI*, 1998(746-752):2, 1998.

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*, 2020.

Benjamin Ellis, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob N Foerster, and Shimon Whiteson. Smacv2: An improved benchmark for cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2212.07489*, 2022.

Jerzy Filar and Koos Vrieze. *Competitive Markov decision processes*. Springer Science & Business Media, 2012.

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.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

Ian Gemp, Brian McWilliams, Claire Vernade, and Thore Graepel. Eigengame: {PCA} as a nash equilibrium. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=NzTU59SYbNq`.

Siyi Hu, Chuanlong Xie, Xiaodan Liang, and Xiaojun Chang. Policy diagnosis via measuring role diversity in cooperative multi-agent rl. In *International Conference on Machine Learning*, pages 9041–9071. PMLR, 2022a.

Siyi Hu, Yifan Zhong, Minquan Gao, Weixun Wang, Hao Dong, Zhihui Li, Xiaodan Liang, Xiaojun Chang, and Yaodong Yang. Marllib: Extending rllib for multi-agent reinforcement learning. *arXiv preprint arXiv:2210.13708*, 2022b.

Maximilian Hüttenrauch, Adrian Šošić, and Gerhard Neumann. Guided deep reinforcement learning for swarm systems. *arXiv preprint arXiv:1709.06011*, 2017.

Maximilian Hüttenrauch, Sosic Adrian, Gerhard Neumann, et al. Deep reinforcement learning for swarm systems. *Journal of Machine Learning Research*, 20(54):1–31, 2019.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *In Proc. 19th International Conference on Machine Learning*. Citeseer, 2002.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Jakub Grudzien Kuba, Muning Wen, Linghui Meng, Haifeng Zhang, David Mguni, Jun Wang, Yaodong Yang, et al. Settling the variance of multi-agent policy gradients. *Advances in Neural Information Processing Systems*, 34:13458–13470, 2021.

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*, 2022a. URL `https://openreview.net/forum?id=EcGGFkNTxdJ`.

Jakub Grudzien Kuba, Christian Schroeder de Witt, and Jakob Foerster. Mirror learning: A unifying framework of policy optimisation. *ICML*, 2022b.

Karol Kurach, Anton Raichuk, Piotr Stańczyk, Michał Zając, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, et al. Google research football: A novel reinforcement learning environment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4501–4510, 2020.

Hepeng Li and Haibo He. Multiagent trust region policy optimization. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.

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

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6382–6393, 2017.

David H Mguni, Yutong Wu, Yali Du, Yaodong Yang, Ziyi Wang, Minne Li, Ying Wen, Joel Jennings, and Jun Wang. Learning in nonzero-sum stochastic games with potentials. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 7688–7699. PMLR, 18–24 Jul 2021.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

John Nash. Non-cooperative games. *Annals of mathematics*, pages 286–295, 1951.

Frans A Oliehoek and Christopher Amato. *A concise introduction to decentralized POMDPs*. Springer, 2016.

Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS)*, 2021. URL `http://arxiv.org/abs/2006.07869`.

Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. Facmac: Factored multi-agent centralised policy gradients. *Advances in Neural Information Processing Systems*, 34:12208–12221, 2021.

P Peng, Q Yuan, Y Wen, Y Yang, Z Tang, H Long, and J Wang. Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games. arxiv 2017. *arXiv preprint arXiv:1703.10069*, 2017.

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

Ray-Team. Ray rllib documentation: Multi-agent deep deterministic policy gradient (maddpg). `https://docs.ray.io/en/latest/rllib/rllib-algorithms.html#multi-agent-deep-deterministic-policy-gradient-maddpg`, accessed on 2023-03-14.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philiph H. S. Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft Multi-Agent Challenge. *CoRR*, abs/1902.04043, 2019.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

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

John Schulman, F. Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017.

Lloyd S Shapley. Stochastic games. *Proceedings of the national academy of sciences*, 39(10): 1095–1100, 1953.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. PMLR, 2014.

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

R. S. Sutton, D. Mcallester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12*, volume 12, pages 1057–1063. MIT Press, 2000.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pages 330–337, 1993.

J Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sullivan, Luis S Santos, Clemens Dieffendahl, Caroline Horsch, Rodrigo Perez-Vicente, et al. Pettingzoo: Gym for multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34:15032–15043, 2021.

Justin K Terry, Nathaniel Grammel, Sanghyun Son, and Benjamin Black. Parameter sharing for heterogeneous agents in multi-agent reinforcement learning. *arXiv preprint arXiv:2005.13625*, 2020.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Jiangxing Wang, Deheng Ye, and Zongqing Lu. More centralized training, still decentralized execution: Multi-agent conditional policy factorization. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=znLlSgN-4S0`.

Tonghan Wang, Tarun Gupta, Anuj Mahajan, Bei Peng, Shimon Whiteson, and Chongjie Zhang. Rode: Learning roles to decompose multi-agent tasks. *International Conference on Learning Representations*, 2021.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.

Ying Wen, Yaodong Yang, Rui Luo, Jun Wang, and Wei Pan. Probabilistic recursive reasoning for multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2018.

Ying Wen, Yaodong Yang, and Jun Wang. Modelling bounded rationality in multi-agent interactions by generalized recursive reasoning. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 414–421. International Joint Conferences on Artificial Intelligence Organization, 7 2020. Main track.

Ying Wen, Hui Chen, Yaodong Yang, Minne Li, Zheng Tian, Xu Chen, and Jun Wang. A game-theoretic approach to multi-agent trust region optimization. In *International Conference on Distributed Artificial Intelligence*, pages 74–87. Springer, 2022.

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.

Yaodong Yang and Jun Wang. An overview of multi-agent reinforcement learning from game theoretical perspective. *arXiv preprint arXiv:2011.00583*, 2020.

Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. Mean field multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 5571–5580. PMLR, 2018.

Yaodong Yang, Ying Wen, Jun Wang, Liheng Chen, Kun Shao, David Mguni, and Weinan Zhang. Multi-agent determinantal q-learning. In *International Conference on Machine Learning*, pages 10757–10766. PMLR, 2020.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of PPO in cooperative multi-agent games. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

Haifeng Zhang, Weizhe Chen, Zeren Huang, Minne Li, Yaodong Yang, Weinan Zhang, and Jun Wang. Bi-level actor-critic for multi-agent coordination. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7325–7332, 2020.

Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of reinforcement learning and control*, pages 321–384, 2021.

Ming Zhou, Ziyu Wan, Hanjing Wang, Muning Wen, Runzhe Wu, Ying Wen, Yaodong Yang, Yong Yu, Jun Wang, and Weinan Zhang. Malib: A parallel framework for population-based multi-agent reinforcement learning. *Journal of Machine Learning Research*, 24(150): 1–12, 2023. URL `http://jmlr.org/papers/v24/22-0169.html`.