Supplementary Material

A Related Works

A.1 Tackle Non-Stationarity in MARL

Many works have been proposed to tackle non-stationarity in MARL. These methods range from using a modification of standard RL training schemes to computing and sharing additional other agents' information.

A.1.1 Modification of Standard RL Training Schemes

One modification of the standard RL training schemes is the centralized critic and decentralized actor (CCDA) architecture. Since the centralized critics can access the information of all other agents during training, the dynamics of the environment remain stable for the agent. Lowe et al. (2017) combined DDPG (Lillicrap et al., 2016) with CCDA architecture and proposed MADDPG algorithm. Foerster et al. (2018b) proposed a counterfactual baseline in the advantage estimation and used the REINFORCE (Williams, 1992) algorithm as the backbone in the CCDA architecture. Similar with Lowe et al. (2017), Iqbal & Sha (2019) combined SAC (Haarnoja et al., 2018) with CCDA architecture and introduced attention mechanism into the centralized critic design. In addition, Iqbal & Sha (2019) also introduced the counterfactual baseline proposed by Foerster et al. (2018b). CCDA alleviated non-stationarity problems indirectly makes it unstable and ineffective, and our experiments have also verified this. Specifically, on the one hand, the centralized critic does not directly affect the agent's policy but only influences the update direction of the policy through the gradient. The policy modeling does not take into account the actions of other agents like centralized critics (considering the decisions of other agents in centralized critics is the crucial improvement of this type of algorithm) but only based on local observations. On the other hand, and more importantly, even if centralized critics explicitly consider the actions of other agents, they cannot mitigate the adverse effects of non-stationarity. Centralized critics only consider the sample of other agent's policy distribution. Once the other agent's policy change drastically and frequently (corresponding to more severe non-stationarity), they need more samples to "implicit" modeling the outside environment, which leads to higher sample complexity.

Another modification to handle non-stationarity in MARL is self-play for competitive tasks or population-based training for cooperative tasks (Papoudakis et al., 2019). Tesauro (1995) used self-play to train the TD-Gammon, which managed to win against the human champion in Backgammon. Baker et al. (2019) extended self-play to more complex environments with continuous state and action space. Liu et al. (2019) and Jaderberg et al. (2019) combined population-based training with self-play to solve complex team competition tasks, a popular 3D multiplayer first-person video game, *Quake III Arena Capture the Flag*, and *MuJoCo Soccer*, respectively. However, such methods require a lot of hardware resources and a well-designed parallelization platform.

A.1.2 COMPUTING AND SHARING ADDITIONAL INFORMATION

In addition to modifying the standard RL training scheme, there are also methods to solve non-stationarity problems by computing and sharing additional information. One naive approach is parameter sharing and use agents' aggregated trajectories to conduct policy optimization at every iteration (Gupta et al., 2017; Terry et al., 2020). Unfortunately, this simple approach has significant drawbacks. An obvious demerit is that parameter sharing requires that all agents have identical action spaces, *i.e.*, $\mathcal{A}^i = \mathcal{A}^j, \forall i, j \in \mathcal{N}$, which limits the class of MARL problems to solve. Importantly, enforcing parameter sharing is equivalent to putting a constraint $\theta^i = \theta^j, \forall i, j \in \mathcal{N}$ on the joint policy space. In principle, this can lead to a suboptimal solution. To elaborate, we have following proposition proposed by Kuba et al. (2021):

Proposition 1 (suboptimal)**.** *Let's 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 $r(a^{1:n}) = 0$ for all other joint actions. Let $\mathcal{J}^*$ be the optimal joint reward, and $\mathcal{J}^*_{share}$ be the optimal joint reward under the shared policy constraint. Then*

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

This proposition shows that parameter sharing can lead to a suboptimal outcome that is exponentially-worse with the increasing number of agents.

In addition, there are many other ways to share or compute additional information among agents. Foerster et al. (2017) proposed importance sampling corrections to adjust the weight of previous experience to the current environment dynamic to stabilize multi-agent experience replay. Raileanu et al. (2018) and Rabinowitz et al. (2018) used additional networks to predict the actions or goals of other agents and input them as additional information into the policy network to assist decision-making. Foerster et al. (2018a) accessed the optimized trajectory of other agents by explicitly predicting the parameter update of other agents when calculating the policy gradient, thereby alleviating the non-stationarity problem. These explicitly considering other agents' information are also called *modeling of others*. Recently, Al-Shedivat et al. (2018) transformed non-stationarity problems into meta-learning problems, and extended MAML (Finn et al., 2017) to MAS to find an initialization policy that can quickly adapt to non-stationarity. However, due to the unique training mechanism of the above methods, they are difficult to extend to the tasks of more than 2 agents.

A.2 TRUST-REGION METHODS

A.2.1 TRUST-REGION METHODS IN SINGLE-AGENT RL

trust-region or *proximity-based* methods, resonating the fact they make the new policy lie within a trust-region around the old one. Such methods include traditional dynamic programming-based conservative policy iteration (CPI) algorithm (Kakade & Langford, 2002), as well as deep RL methods, such as trust-region policy optimization (TRPO) (Schulman et al., 2015) and proximal policy optimization (PPO) (Schulman et al., 2017). TRPO used line-search to ensure that the KL divergence between the new policy and the old policy is below a certain threshold. PPO is to solve a more relaxed unconstrained optimization problem, in which the ratio of the old and new policy is clipped to a specific bound. Wu et al. (2017) (ACKTR) extended the framework of natural policy gradient and proposed to

optimize both the actor and the critic using Kronecker-factored approximate curvature (K-FAC) with trust-region. Nachum et al. (2018) proposed an off-policy trust-region method, Trust-PCL, which introduced relative entropy regularization to maintain optimization stability while exploiting off-policy data. Recently, Tomar et al. (2020) used mirror decent to solve a relaxed unconstrained optimization problem and achieved strong performance.

A.2.2 TRUST-REGION METHODS IN MARL

Extending trust-region methods to MARL is highly non-trivial. Despite empirical successes, none of them managed to propose a theoretically-justified trust-region protocol in multi-agent learning. 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) assume homogeneity of action spaces for all agents and enforce parameter sharing which is discussed above. Yu et al. (2021) 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. Wen et al. (2021) 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. Hu & Hu (2021) developed Noisy-MAPPO that targets to address the sub-optimality issue; however, it still lacks theoretical insights for the modification made on MAPPO. Recently, Li & He (2020) tried to implement TRPO for MARL through distributed consensus optimization; however, they enforced the same trust-region for all agents (see their Equation (7)) which, similar to parameter sharing, largely limits the policy space for optimization, and this also will make the algorithm face the trust-region decomposition dilemma. The method HATRPO (Kuba et al., 2021) based on sequential update scheme is proposed from the perspective of monotonic improvement guarantee. But at the same time, sequential update scheme also limit the scalability of the algorithm.

In addition, Jiang & Lu (2021) proposes a MARL algorithm to adjust the learning rates of agents adaptively. Limiting the size of the trust-region of each agent's local policy is related to limiting the learning rate of its local policy. They focus on speeding up the learning speed, instead of solving non-stationarity problems, However, Jiang & Lu (2021) has no theoretical support, and matching the direction of adjusting the learning rates with the directions of maximizing the Q values may cause overestimation problems.

B PRELIMINARIES

Cooperative POSG. POSG (Hansen et al., 2004) is denoted as a seven-tuple via the stochastic game (or Markov game)

$$\langle \mathcal{I}, \mathcal{S}, \{\mathcal{A}_i\}_{i=1}^n, \{\mathcal{O}_i\}_{i=1}^n, \mathcal{P}, \mathcal{E}, \{\mathcal{R}_i\}_{i=1}^n \rangle,$$

where n denotes the number of agents; $\mathcal{I}$ represents the agent space; $\mathcal{S}$ represents the finite set of states; $\mathcal{A}_i$, $\mathcal{O}_i$ denote a finite action set and a finite observation set of agent $i \in \mathcal{I}$ respectively; $\boldsymbol{\mathcal{A}} = \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_n$ is the finite set of joint actions; $\mathcal{P}(s'|s, \boldsymbol{a})$ denotes the Markovian state transition probability function, where $s, s' \in \mathcal{S}$ represent states of environment and $\boldsymbol{a} = \{a_i\}_{i=1}^n, a_i \in \mathcal{A}_i$ represents the action of agent i; $\boldsymbol{\mathcal{O}} = \mathcal{O}_1 \times \mathcal{O}_2 \times \cdots \times \mathcal{O}_n$ is the finite set of joint observations; $\mathcal{E}(\boldsymbol{o}|s)$ is the Markovian observation emission probability function, where $\boldsymbol{o} = \{o_i\}_{i=1}^n, o_i \in \mathcal{O}_i$ represents the local observation of agent i; $\mathcal{R}_i : \mathcal{S} \times \boldsymbol{\mathcal{A}} \times \mathcal{S} \to \mathcal{R}$ denotes the reward function of agent i and $r_i \in \mathcal{R}_i$ is the reward of agent i. The game in POSG unfolds over a finite or infinite sequence of stages

(or timesteps), where the number of stages is called *horizon*. In this paper, we consider the finite horizon case. The objective for each agent is to maximize the expected cumulative reward received during the game. For a cooperative POSG, we quote the definition in Song et al. (2020),

$$\forall i \in \mathcal{I}, \forall i' \in \mathcal{I} \backslash \{i\}, \forall \pi_i \in \Pi_i, \forall \pi_{i'} \in \Pi_{i'}, \frac{\partial \mathcal{R}_{i'}}{\partial \mathcal{R}_i} \geqslant 0,$$

where i and i' are a pair of agents in agent space $\mathcal{I}$; π_i and $\pi_{i'}$ are the corresponding policies in the policy space Π_i and $\Pi_{i'}$ respectively. Intuitively, this definition means that there is no conflict of interest for any pair of agents.

Mirror descent method in RL. The mirror descent method (Beck & Teboulle, 2003) is a typical first-order optimization method, which can be considered an extension of the classical proximal gradient method. In order to minimize the objective function $f(\mathbf{x})$ under a constraint set $\mathbf{x} \in \mathcal{C} \subseteq \mathbb{R}^n$, the basic iterative scheme at iteration k+1 can be written as

$$\mathbf{x}^{k+1} \in \arg\min_{\mathbf{x} \in \mathcal{C}} \left\langle \nabla f\left(\mathbf{x}^k\right), \mathbf{x} - \mathbf{x}^k \right\rangle + \gamma^k B_\psi\left(\mathbf{x}, \mathbf{x}^k\right), \tag{10}$$

where $B_\psi(\mathbf{x}, \mathbf{y}) := \psi(\mathbf{x}) - \psi(\mathbf{x}) - \langle \nabla \psi(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle$ denotes the Bregman divergence associated with a strongly convex function ϕ and γ^k is the step size (or learning rate). Each reinforcement learning problem can be formulated as optimization problems from two distinct perspectives, i.e.,

$$\pi^*(\cdot \mid s) \in \arg\max_\pi V^\pi(s), \quad \forall s \in \mathcal{S}; \tag{11a}$$

$$\pi^* \in \arg\max_\pi \mathbb{E}_{s \sim \mu}\left[V^\pi(s)\right]. \tag{11b}$$

Geist et al. (2019) and Shani et al. (2020) have utilized the mirror descent scheme (10) and update the policy iteratively as follows

$$\pi^{k+1}(\cdot|s) \leftarrow \arg\max_\pi \mathbb{E}_{a \sim \pi}\left[A^{\pi^k}(s,a)\right] - \gamma^k \operatorname{KL}\left(\pi, \pi^k\right);$$

$$\pi^{k+1} \leftarrow \arg\max_\pi \mathbb{E}_{s \sim \rho_{\pi^k}}\left[\mathbb{E}_{a \sim \pi}\left[A^{\pi^k}(s,a)\right] - \gamma^k \operatorname{KL}\left(\pi, \pi^k\right)\right],$$

where $\operatorname{KL}(\cdot, \cdot)$ denotes the Bregman divergence corresponding to negative entropy function.

C PSEUDO-CODE OF MAMT

This section gives the pseudo-code of the MAMT algorithm (see Algorithm 1[‖]) and the MAMT algorithm without trust-region decomposition network (see Algorithm 2). For convenience, we named the latter MAMD. In the MAMD algorithm, the trust-region constraint is equally distributed to the local policies of all agents.

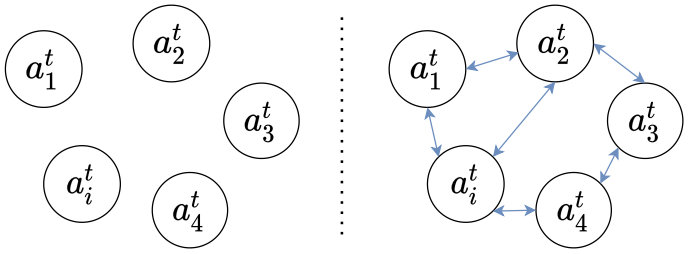


Figure 6: The probabilistic graphical models of two different modelings of the joint policy. **Left:** Modeling the joint policy with the mean-field variation family; **Right:** Modeling the joint policy as a pairwise Markov random field. Each node represents the action of agent i at timestep t.

D TRUST-REGION DECOMPOSITION DILEMMA

Algorithm 1: MAMT

Input : initial behavior policy π_{ψ_i} and main centralized critic Q_{ζ_i} for each agent i, empty replay buffer $\mathcal{D}$, trust-region decomposition network $f_{\theta-}$ and g_{w+}, local trust-region δ_i^0

Set target critic $Q_{\bar{\zeta}_i}$ equal to main critic;
Set $\pi_{\bar{\psi}_i}$ and $\pi_{\psi_i^{\text{old}}}$ equal to behavior policy;
while *not convergence* **do**
  Observe local observation o_i and select action $a_i \sim \pi_\delta(\cdot \mid o_i)$ for each agent i;
  Execute joint action a in the environment;
  Observe next local observation o_i', local reward r_i and local done signal d_i of each agent i;
  Store $(\{o_i\}, \{a_i\}, \{r_i\}, \{o_i^{'}\}, \{d_i\})$ in replay buffer $\mathcal{D}$;
  if *it's time to update* **then**
    for *j in range(however many updates)* **do**
      Sample a batch of transitions $\mathcal{B}$ from $\mathcal{D}$;
      Compute $\{\mathcal{C}_{i,\setminus i}^t\}$ with Eq. 7;
      Compute $\{\mathcal{D}_{i,\text{ns}}^t\}$ with Eq. 5;
      for *slow update* **do**
        Update δ_i^t with Eq. 8;
      for *fast update* **do**
        Update $f_{\theta-}, g_{w+}$ with Eq. 8;
      Update centralized critic of all agents by Eq. 9;
      Update individual policy of all agents by Eq. 10;
      Update target networks $Q_{\bar{\zeta}_i}$ and $\pi_{\bar{\psi}_i}$;
      if *it's time to update* **then**
        Update old policy $\pi_{\psi_i^{\text{old}}}$;

In order to verify the existence of the trust-region decomposition dilemma, we defined a simple coordination environment. We extend the *Spread* environment of Section 4 to 3 agents and 3 landmarks, labeled **Spread-3**. We define different Markov random fields (see Figure 7) of the joint policy by changing the reward function to *influence the transition function of each agent indirectly*.

Specifically, the reward function of each agent is composed of two parts: the minimum distance between all agents and the landmarks, and the other is the collision. For the

[‖]The source code is available at https://anonymous.4open.science/r/MAMT.

Algorithm 2: MAMD.

Input : initial behavior policy π_{ψ_i} and main centralized critic Q_{ζ_i} for each agent i, empty replay buffer $\mathcal{D}$

Set target critic $Q_{\bar{\zeta}_i}$ equal to main critic;
Set target policy $\pi_{\bar{\psi}_i}$ equal to behavior policy;
Set old policy $\pi_{\psi_i^{\text{old}}}$ equal to behavior policy;
while *not convergence* **do**
  Observe local observation o_i and select action $a_i \sim \pi_\delta(\cdot \mid o_i)$ for each agent i;
  Execute joint action a in the environment;
  Observe next local observation o_i', local reward r_i and local done signal d_i of each agent i;
  Store $(\{o_i\}, \{a_i\}, \{r_i\}, \{o_i^{'}\}, \{d_i\})$ in replay buffer $\mathcal{D}$;
  if *All d_i are terminal* **then**
    Reset the environment;
  if *it's time to update* **then**
    for *j in range(however many updates)* **do**
      Sample a batch of transitions $\mathcal{B}$ from $\mathcal{D}$;
      Update centralized critic of all agents by Eq. 9;
      Update individual policy of all agents by Eq. 10;
      Update target networks $Q_{\bar{\zeta}_i}$ and $\pi_{\bar{\psi}_i}$;
      if *it's time to update* **then**
        Update old policy $\pi_{\psi_i^{\text{old}}}$;

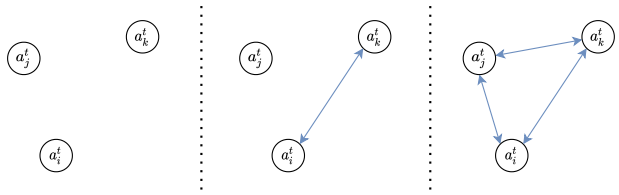
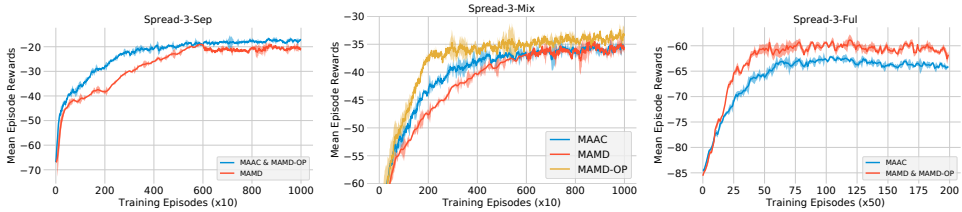


Figure 7: In the Spread-3 environment, 3 different Markov random fields are generated due to the different definitions of the reward function of each agent. Note that these are not all possible Markov random fields, but three typical cases.

leftmost MRF in Figure 7, all agents are independent. The reward function of each agent is only related to itself, only related to the minimum distance between itself and a specific landmark, and will not collide with other agents. We labeled this situation as **Spread-3-Sep**. For the middle MRF in Figure 7, the reward function of agent j is the same as that of *Spread-3-Sep*, which is only related to itself; but agent i and k are interdependent. For agent i, its reward function consists of the minimum distance between i and k and the landmark and whether the two collide. The reward function of agent k is similar. We labeled this situation as **Spread-3-Mix**. Finally, the rightmost MRF is consistent with the standard environment settings, labeled **Spread-3-Ful**.

To verify the existence of the trust-region decomposition dilemma, we compare the performance of three different algorithms. First, we select MAAC without any trust-region constraints as the baseline, labeled **MAAC**. Secondly, we choose MAMD based on mean-field approximation and naive trust-region decomposition as one of the algorithms to be compared, labeled **MAMD**. Finally, we optimally assign trust-region based on prior knowledge. For *Spread-3-Sep*, we do not impose any trust-region constraints, same as *MAAC*; for *Spread-3-Mix*, we only impose equal size constraints on agent i and k; and for *Spread-3-Ful*, we impose equal size constraints on all agents, same as *MAMD*. We labeled these optimally decomposition as **MAMD-OP**. The performance is shown in Figure 8.

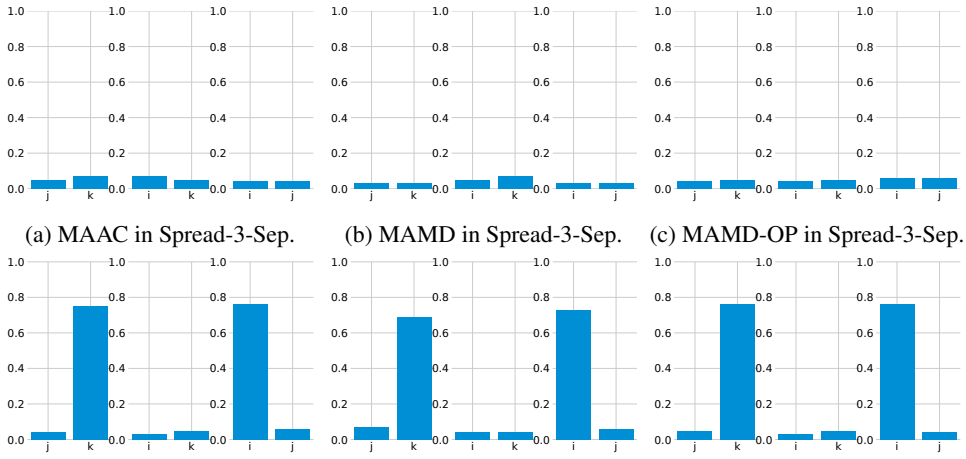


(a) Spread-3-Sep. (b) Spread-3-Mix. (c) Spread-3-Ful.

Figure 8: The performance of different trust-region decompositions in different scenarios. These results indicate the existence of a trust-region decomposition dilemma.

It can be seen from the figure that inappropriate decomposition of the trust-region will negatively affect the convergence speed and performance of the algorithm. The optimal decomposition method can make the algorithm performance and convergence speed steadily exceed baselines.

Note that the three algorithms compared here are all based on the MAAC with centralized critics. After we change the reward function of the agent, the information received by these centralized critics has more redundancy in some scenarios (for example, in *Spread-3-Sep* and *Spread-3-Mix*). To exclude the algorithm's performance from being affected by this redundant information, we output the attention weights in MAAC, as shown in Figure 9. It can be seen from the figure that different algorithms can filter redundant information well, thus eliminating the influence of redundant information on the convergence speed and performance of the algorithm.



(a) MAAC in Spread-3-Sep. (b) MAMD in Spread-3-Sep. (c) MAMD-OP in Spread-3-Sep.

(d) MAAC in Spread-3-Mix. (e) MAMD in Spread-3-Mix. (f) MAMD-OP in Spread-3-Mix.

Figure 9: The attention weights of the different agents of different trust-region decomposition in different scenarios. These results indicate the existence of a trust-region decomposition dilemma.

E PROOFS

E.1 PROOF OF LEMMA 1

Proof.

$$\begin{aligned}
D_{\mathrm{TV}}(p^t(\cdot|\boldsymbol{o},a_i),p^{t+1}(\cdot|\boldsymbol{o},a_i)) &= \max_{\boldsymbol{o},a_i}|p^{t+1}(\cdot|\boldsymbol{o},a_i)-p^t(\cdot|\boldsymbol{o},a_i)| \\
&= \max_{\boldsymbol{o},a_i}|\int p(\cdot|\boldsymbol{o},\boldsymbol{a})\cdot\left(\pi_{-i}^{t+1}(a_{-i}|\boldsymbol{o})-\pi_{-i}^{t}(a_{-i}|\boldsymbol{o})\right)\mathrm{d}a_{-i}| \\
&\leq \max_{\boldsymbol{o}}\int\left|\pi_{-i}^{t+1}(a_{-i}|\boldsymbol{o})-\pi_{-i}^{t}(a_{-i}|\boldsymbol{o})\right|\mathrm{d}a_{-i} \\
&= \max_{\boldsymbol{o}}\left\|\pi_{-i}^{t+1}(\boldsymbol{o})-\pi_{-i}^{t}(\boldsymbol{o})\right\|_1 \\
&\leq 2\ln 2\cdot\max_{\boldsymbol{o}}D_{\mathrm{KL}}^{1/2}(\pi_{-i}^{t}(\boldsymbol{o})\|\pi_{-i}^{t+1}(\boldsymbol{o})) \\
&\leq 2\ln 2\cdot\delta_i^{1/2}.
\end{aligned}$$

□

E.2 PROOF OF LEMMA 2

Proof.

$$\begin{aligned}
D_{\mathrm{TV}}(r^t(\boldsymbol{o},a_i),r^{t+1}(\boldsymbol{o},a_i)) &= \max_{\boldsymbol{o},a_i}|r^{t+1}(\boldsymbol{o},a_i)-r^t(\boldsymbol{o},a_i)| \\
&= \max_{\boldsymbol{o},a_i}|\int r(\boldsymbol{o},\boldsymbol{a})\cdot\left(\pi_{-i}^{t+1}(a_{-i}|\boldsymbol{o})-\pi_{-i}^{t}(a_{-i}|\boldsymbol{o})\right)\mathrm{d}a_{-i}| \\
&\leq \max_{\boldsymbol{o}}\int\left|\pi_{-i}^{t+1}(a_{-i}|\boldsymbol{o})-\pi_{-i}^{t}(a_{-i}|\boldsymbol{o})\right|\mathrm{d}a_{-i} \\
&= \max_{\boldsymbol{o}}\left\|\pi_{-i}^{t+1}(\boldsymbol{o})-\pi_{-i}^{t}(\boldsymbol{o})\right\|_1 \\
&\leq 2\ln 2\cdot\max_{\boldsymbol{o}}D_{\mathrm{KL}}^{1/2}(\pi_{-i}^{t}(\boldsymbol{o})\|\pi_{-i}^{t+1}(\boldsymbol{o})) \\
&\leq 2\ln 2\cdot\delta_i^{1/2}.
\end{aligned}$$

□

E.3 PROOF OF THEOREM 1

Proof. In this paper, we model the learning procedure of each agent in a multi-agent system as a dynamic non-stationary MDP. From each agent's perspective, the quantities $r_t(\boldsymbol{o},a_i)$'s and $p_t(\cdot|\boldsymbol{o},a_i)$'s of each agent i vary across different t's in general. Following Besbes et al. (2014), Cheung et al. (2019) and Mao et al. (2021), we quantify the variations on $r_t(\boldsymbol{o},a_i)$'s and $p_t(\cdot|\boldsymbol{o},a_i)$'s in terms of their respective *variation budgets* B_r, B_p (>0):

$$B_r=\sum_{t=1}^{T-1}B_{r,t},\ \text{where}\ B_{r,t}=\max_{o\in\mathcal{O},a_i\in\mathcal{A}_i}\left|r_{t+1}(\boldsymbol{o},a_i)-r_t(\boldsymbol{o},a_i)\right|,$$

$$B_p=\sum_{t=1}^{T-1}B_{p,t},\ \text{where}\ B_{p,t}=\max_{o\in\mathcal{O},a_i\in\mathcal{A}_i}\left\|p_{t+1}(\cdot\mid\boldsymbol{o},a_i)-p_t(\cdot\mid\boldsymbol{o},a_i)\right\|_1.$$

To measure the convergence to the best-response from each agent's perspective, we consider an objective of minimizing the *dynamic regret* (Jaksch et al., 2010; Besbes et al.,

2014; Cheung et al., 2019; Mao et al., 2021)

$$\text{Dyn-Reg}_T(\Pi) = \sum_{t=1}^{T} \left\{ \rho_t^* - \mathbb{E}\left[r_{i,t}\left(\boldsymbol{o}_t, \boldsymbol{a}_t \right) \right] \right\}.$$

In the oracle $\sum_{t=1}^{T} \rho_t^*$, the summand ρ_t^* is the optimal long-term average reward of the stationary MDP, i.e., other agents follow the fixed optimal policies, with state transition distribution $p_{i,t}$ and mean reward $r_{i,t}$. Below we give a definition and an assumption.

Definition 4 (Communicating MDPs and Diameter)**.** *Consider a set of states $\mathcal{S}$, a collection $\mathcal{A} = \{\mathcal{A}_s\}_{s\in\mathcal{S}}$ of action sets, and a state transition distribution $\bar{p} = \{\bar{p}(\cdot \mid s,a)\}_{s\in\mathcal{S}, a\in\mathcal{A}_s}$. For any $s, s' \in \mathcal{S}$ and stationary policy π, the hitting time from s to s' under π is the random variable $\Lambda(s' \mid \pi, s) := \min\{t : s_{t+1} = s', s_1 = s, s_{\tau+1} \sim \bar{p}(\cdot \mid s_\tau, \pi(s_\tau)) \forall \tau\}$, which can be infinite. We say that is a communicating MDP iff $D := \max_{s,s'\in\mathcal{S}} \min_{\text{stationary } \pi} \mathbb{E}[\Lambda(s' \mid \pi, s)]$ is finite. The quantity D is the diameter (Jaksch et al., 2010) associated with $(\mathcal{S}, \mathcal{A}, \bar{p})$.*

Assumption 2 (Bounded Diameters)**.** *For each $t \in [T]$, the tuple $(\mathcal{S}, \mathcal{A}, p_t)$ constitutes a communicating MDP with diameter at most D_t. We denote the maximum diameter as $D_{max} = \max_{t\in\{1,\ldots,T\}} D_t$.*

Then we have following proposition (Cheung et al., 2019) from each agent's perspective:

Proposition 2. *Consider an instance $(\mathcal{S}, \mathcal{A}, T, p, r)$ from each agent's perspective that satisfies Assumption 2 with maximum diameter D_{max} and has variation budgets B_r, B_p for rewards and transition distributions respectively. In addition, suppose that $T \geq B_r + 2D_{\max}B_p > 0$, then it holds that*

$$\sum_{t=1}^{T} \rho_t^* \geq \max_{\Pi} \left\{ \mathbb{E}\left[\sum_{t=1}^{T} r_t\left(s_t^\Pi, a_t^\Pi \right) \right] \right\} - 4\left(D_{\max} + 1 \right) \sqrt{\left(B_r + 2D_{\max}B_p \right) T}.$$

The maximum is taken over all non-anticipatory policies Π's. We denote $\left\{ \left(s_t^\Pi, a_t^\Pi \right) \right\}_{t=1}^{T}$ as the trajectory under policy Π, where $a_t^\Pi \in \mathcal{A}_{s_t^\Pi}$ is determined based on Π and $\mathcal{H}_{t-1} \cup \left\{ s_t^\Pi \right\}$, and $s_{t+1}^\Pi \sim p_t\left(\cdot \mid s_t^\Pi, a_t^\Pi \right)$ for each t.

The proof of Proposition 2 is shown in (Cheung et al., 2019). Based on Lemma 1 and Lemaa 2, we can easily obtain $B_r \leq 2\ln 2 \cdot \delta_i^{1/2} \cdot T$ and $B_p \leq 2\ln 2 \cdot \delta_i^{1/2} \cdot T \cdot |\mathcal{O}|$. Then we have following corollary:

Corollary 1. *Consider an instance $(\mathcal{S}, \mathcal{A}, T, p, r)$ from each agent's perspective that satisfies Assumption 2 with maximum diameter D_{max} and has variation budgets B_r, B_p for rewards and transition distributions respectively. In addition, suppose that $T \geq B_r + 2D_{\max}B_p > 0$, then it holds that*

$$\sum_{t=1}^{T} \rho_t^* \geq \max_{\Pi} \left\{ \mathbb{E}\left[\sum_{t=1}^{T} r_t\left(s_t^\Pi, a_t^\Pi \right) \right] \right\} - 4\left(D_{\max} + 1 \right) \cdot T \sqrt{\left(1 + 2 \cdot D_{max}|\mathcal{O}| \right) 2\ln 2 \cdot \delta_i^{1/2}}.$$

According to the Definition 4, Theorem 1 is proved.

□

E.4 PROOF OF THEOREM 2

Proof.

$$
\begin{aligned}
\mathrm{KL}\left[\pi \mid \pi'\right] &= \int\int \pi(a_i, a_{-i}|\boldsymbol{o}) \log \frac{\pi(a_i, a_{-i}|\boldsymbol{o})}{\pi'(a_i, a_{-i}|\boldsymbol{o})} \mathrm{d}a_i \mathrm{d}a_{-i} \\
&= \int\int \pi(a_i|a_{-i}, \boldsymbol{o})\pi(a_{-i}|\boldsymbol{o}) \\
&\quad \log \frac{\pi(a_i|a_{-i}, \boldsymbol{o})\pi(a_{-i}|\boldsymbol{o})}{\pi'(a_i|a_{-i}, \boldsymbol{o})\pi(a_{-i}|s)} \mathrm{d}a_i \mathrm{d}a_{-i} \\
&= \int\int \pi(a_i|a_{-i}, \boldsymbol{o})\pi(a_{-i}|\boldsymbol{o}) \log \frac{\pi(a_i|a_{-i}, \boldsymbol{o})}{\pi'(a_i|a_{-i}, \boldsymbol{o})} \mathrm{d}a_i \mathrm{d}a_{-i} \\
&\quad + \int\int \pi(a_i|a_{-i}, \boldsymbol{o})\pi(a_{-i}|\boldsymbol{o}) \log \frac{\pi(a_{-i}|\boldsymbol{o})}{\pi'(a_{-i}|\boldsymbol{o})} \mathrm{d}a_i \mathrm{d}a_{-i} \\
&= \int\int \pi(a_i|a_{-i}, \boldsymbol{o})\pi(a_{-i}|\boldsymbol{o}) \log \frac{\pi(a_i|a_{-i}, \boldsymbol{o})}{\pi'(a_i|a_{-i}, \boldsymbol{o})} \mathrm{d}a_i \mathrm{d}a_{-i} + \\
&\quad \mathrm{KL}\left[\pi_{-i} \mid \pi'_{-i}\right] \\
&= \int \mathrm{KL}\left[\pi_i(a_{-i}, \boldsymbol{o}) \mid \pi'_i(a_{-i}, \boldsymbol{o})\right] \pi(a_{-i}|\boldsymbol{o}) \mathrm{d}a_{-i} \\
&\quad + \mathrm{KL}\left[\pi_{-i} \mid \pi'_{-i}\right] \\
&\geq \mathrm{KL}\left[\pi_{-i} \mid \pi'_{-i}\right].
\end{aligned}
$$

So we have

$$
\mathrm{KL}\left[\pi \mid \pi'\right] \geq \frac{1}{n} \sum_i \mathrm{KL}\left[\pi_{-i} \mid \pi'_{-i}\right].
$$

Take the maximum value on both sides of the inequality, Theorem 2 is proved.

□

Since a similar conclusion is reached in Li & He (2020), we will make a brief comparison with it here. Theorem 2 establishes the relationship between the *maximum* KL-divergence of the consecutive joint policies of all agents and the *maximum* KL-divergence of the consecutive joint policies of other agents. It establishes the theoretical connection between the KL-divergence of the consecutive joint policies of all agents and environmental non-stationarity. The Equation (8) of Li & He (2020) extends the KL divergence constraint of the TRPO algorithm to the multi-agent scenario and establishes the connection between the divergence of the joint policy of all agents and the divergence of local policy of each agent.

E.5 PROOF OF THEOREM 3

Proof.

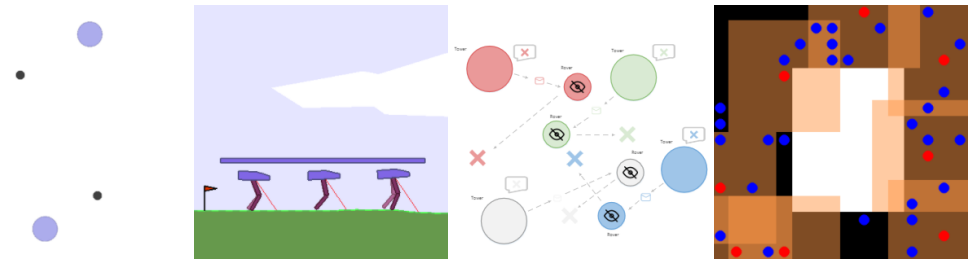
$$
\begin{aligned}
&\mathbb{E}_{\boldsymbol{o}\sim\mu}\left[\mathrm{KL}(\boldsymbol{\pi}(\cdot|\boldsymbol{o}),\boldsymbol{\pi}^k(\cdot|\boldsymbol{o}))\right] < \delta \\
\iff &\mathbb{E}_{\boldsymbol{o}\sim\mu}\left[\int_a \boldsymbol{\pi}(\cdot|\boldsymbol{o})\log\frac{\boldsymbol{\pi}(\cdot|\boldsymbol{o})}{\boldsymbol{\pi}^k(\cdot|\boldsymbol{o}))}\mathrm{d}a\right] < \delta \\
\iff &\mathbb{E}_{\boldsymbol{o}\sim\mu}\left[\int_a \boldsymbol{\pi}(\cdot|\boldsymbol{o})\left(\sum_{i=1}^n \log\frac{\boldsymbol{\pi}_i(\cdot|o_i)}{\boldsymbol{\pi}_i^k(\cdot|o_i))}\right)\mathrm{d}a\right] < \delta \\
\iff &\mathbb{E}_{\boldsymbol{o}\sim\mu}\left[\sum_{i=1}^n\int_a \boldsymbol{\pi}(\cdot|\boldsymbol{o})\log\frac{\boldsymbol{\pi}_i(\cdot|o_i)}{\boldsymbol{\pi}_i^k(\cdot|o_i))}\mathrm{d}a\right] < \delta \\
\iff &\sum_{i=1}^n\mathbb{E}_{\boldsymbol{o}\sim\mu}\left[\int_a \boldsymbol{\pi}(\cdot|\boldsymbol{o})\log\frac{\boldsymbol{\pi}_i(\cdot|o_i)}{\boldsymbol{\pi}_i^k(\cdot|o_i))}\mathrm{d}a\right] < \delta \\
\iff &\sum_{i=1}^n\left[\int_{\boldsymbol{o}}\mu(\boldsymbol{o})\int_a \boldsymbol{\pi}(\cdot|\boldsymbol{o})\log\frac{\boldsymbol{\pi}_i(\cdot|o_i)}{\boldsymbol{\pi}_i^k(\cdot|o_i))}\mathrm{d}a\mathrm{d}\boldsymbol{o}\right] < \delta.
\end{aligned}
$$

We first simplify the integral term of the inner layer

$$
\begin{aligned}
&\int_a \boldsymbol{\pi}(\cdot|\boldsymbol{o})\log\frac{\boldsymbol{\pi}_i(\cdot|o_i)}{\boldsymbol{\pi}_i^k(\cdot|o_i))}\mathrm{d}a \\
\iff &\int_{a_{\setminus i}\times a_i}\boldsymbol{\pi}_{\setminus i}(\cdot|o_{\setminus i})\pi_i(\cdot|o_i)\log\frac{\boldsymbol{\pi}_i(\cdot|o_i)}{\boldsymbol{\pi}_i^k(\cdot|o_i))}\mathrm{d}a_{\setminus i}\mathrm{d}a_i \\
\iff &\int_{a_i}\pi_i(\cdot|o_i)\log\frac{\boldsymbol{\pi}_i(\cdot|o_i)}{\boldsymbol{\pi}_i^k(\cdot|o_i))}\left[\int_{a_{\setminus i}}\boldsymbol{\pi}_{\setminus i}(\cdot|o_{\setminus i})\mathrm{d}a_{\setminus i}\right]\mathrm{d}a_i \\
\iff &\int_{a_i}\pi_i(\cdot|o_i)\log\frac{\boldsymbol{\pi}_i(\cdot|o_i)}{\boldsymbol{\pi}_i^k(\cdot|o_i))}\mathrm{d}a_i \\
\iff &\mathrm{KL}\left(\pi_i(\cdot|o_i),\pi_i^k(\cdot|o_i)\right).
\end{aligned}
$$

We replace the original formula with the simplified one

$$
\begin{aligned}
&\sum_{i=1}^n\left[\int_{\boldsymbol{o}}\mu(\boldsymbol{o})\int_a \boldsymbol{\pi}(\cdot|\boldsymbol{o})\log\frac{\boldsymbol{\pi}_i(\cdot|o_i)}{\boldsymbol{\pi}_i^k(\cdot|o_i))}\mathrm{d}a\mathrm{d}\boldsymbol{o}\right] < \delta \\
\iff &\sum_{i=1}^n\left[\int_{\boldsymbol{o}}\mu(\boldsymbol{o})\mathrm{KL}\left(\pi_i(\cdot|o_i),\pi_i^k(\cdot|o_i)\right)\mathrm{d}\boldsymbol{o}\right] < \delta \\
\iff &\sum_{i=1}^n\left[\int_{\boldsymbol{o}}\mu(\boldsymbol{o})\delta(o_i)\mathrm{d}\boldsymbol{o}\right] < \delta.
\end{aligned}
$$



(a) (b) (c) (d)

Figure 10: Coordination environments with increasing complexity.(a) Spread; (b) Multi-Walker; (c) Rover-tower; (d) Pursuit.

where we denote $\mathrm{KL}\left(\pi_i(\cdot|o_i), \pi_i^k(\cdot|o_i)\right)$ as $\delta(o_i)$. For the outer integral term, we have

$$\begin{aligned}
&\int_{\boldsymbol{o}} \mu(\boldsymbol{o})\delta(o_i)\mathrm{d}\boldsymbol{o} \\
\iff &\int_{o_{\setminus i}\times o_i} \mu(o_{\setminus i})\mu(o_i)\delta(o_i)\mathrm{d}o_{\setminus i}\mathrm{d}o_i \\
\iff &\int_{o_i} \mu(o_i)\delta(o_i)\left[\int_{o_{\setminus i}} \mu(o_{\setminus i})\mathrm{d}o_{\setminus i}\right]\mathrm{d}o_i \\
\iff &\int_{o_i} \mu(o_i)\delta(o_i)\mathrm{d}o_i \\
\iff &\mathbb{E}_{o_i\sim u_i}\left[\delta(o_i)\right].
\end{aligned}$$

Overall, we have

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{o}\sim\mu}\left[\mathrm{KL}(\boldsymbol{\pi}(\cdot|\boldsymbol{o}), \boldsymbol{\pi}^k(\cdot|\boldsymbol{o}))\right] < \delta \\
\iff &\sum_{i=1}^{n}\left[\int_{\boldsymbol{o}} \mu(\boldsymbol{o}) \int_a \boldsymbol{\pi}(\cdot|\boldsymbol{o}) \log \frac{\boldsymbol{\pi}_i(\cdot|o_i)}{\boldsymbol{\pi}_i^k(\cdot|o_i))}\mathrm{d}a\mathrm{d}\boldsymbol{o}\right] < \delta \\
\iff &\sum_{i=1}^{n}\left[\int_{\boldsymbol{o}} \mu(\boldsymbol{o})\delta(o_i)\mathrm{d}\boldsymbol{o}\right] < \delta \\
\iff &\sum_{i=1}^{n}\mathbb{E}_{o_i\sim u_i}\left[\delta(o_i)\right] < \delta \\
\iff &\sum_{i=1}^{n}\mathbb{E}_{o_i\sim u_i}\left[\mathrm{KL}\left(\pi_i(\cdot|o_i), \pi_i^k(\cdot|o_i)\right)\right] < \delta.
\end{aligned}$$

□

F Experimental Details

F.1 Environments

Spread. This environment (Lowe et al., 2017) has 2 agents, 2 landmarks. Each agent is globally rewarded based on how far the closest agent is to each landmark (sum of the minimum distances). Locally, the agents are penalized if they collide with other agents (-1 for each collision).

Multi-Walker. In this environment (Terry et al., 2020a), bipedal robots attempt to carry a

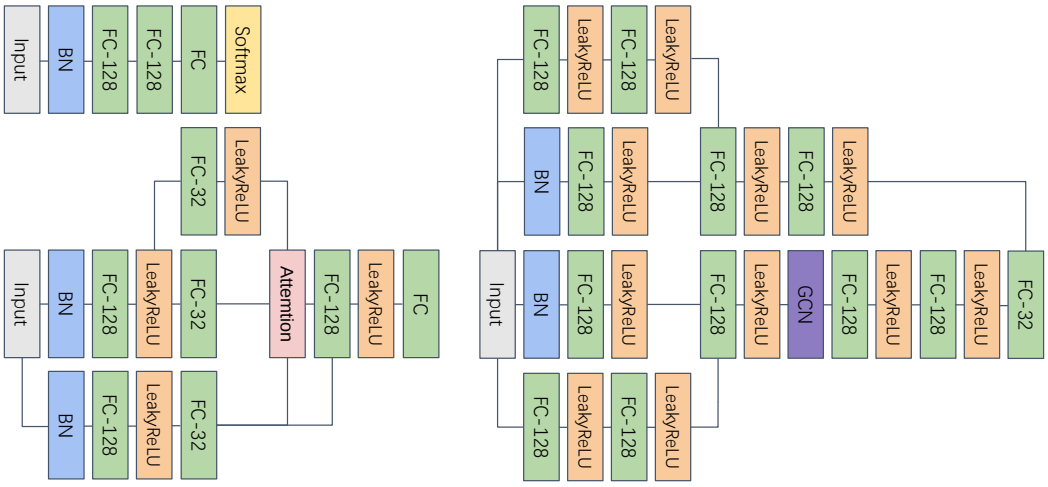


Figure 11: There are actor-network (i.e., policy and modeling policy) architecture, critic-network architecture, and trust-region decomposition network architecture from left to right and from top to bottom.

package as far right as possible. A package is placed on top of 3 bipedal robots. A positive reward is awarded to each walker, which is the change in the package distance.

Rover-Tower. This environment (Iqbal & Sha, 2019) involves 8 agents, 4 of which are "rovers" and another 4 which are "towers". In each episode, rovers and towers are randomly paired. The pair is negatively rewarded by the distance of the rover to its goal. The rovers are unable to see in their surroundings and must rely on communication from the towers, sending one of 5 discrete messages.

Pursuit. 30 blue evaders and 8 red pursuer agents are placed in a grid with an obstacle. The evaders move randomly, and the pursuers are controlled (Terry et al., 2020a). Every time the pursuers surround an evader, each of the surrounding agents receives a reward of 5, and the evader is removed from the environment. Pursuers also receive a reward of 0.01 every time they touch an evader.

F.2 Other Details

Random seeds. All experiments were run for 8 random seeds each. Graphs show the average (solid line) and std dev (shaded) performance over random seed throughout training.

Performance metric. Performance for the on-policy (MA-PPO) algorithms is measured as the average reward across the batch collected at each epoch. Performance for the off-policy algorithms (MAMT, MAMD, LOLA, MADDPG, and MAAC) is measured by running the deterministic policy (or, in the case of SAC, the mean policy) without action noise for 10 trajectories and reporting the average reward over those test trajectories.

Network Architecture. Figure 11 shows the detailed parameters of the three main networks in the MAMT algorithm.

Hyperparameters. Table 1 shows the default configuration used for all the experiments of our methods (MAMD and MAMT) and baselines in this paper. We do not fine-tune the hyperparameters of baselines and use the default setting as same as original papers, see Table 2,3,4 and 5. The hyperparameter fine-tune range of our methods are shown in Table 6 and 7. Considering the long training time of the MARL algorithm, we did not train all

hyperparameter combinations to the pre-defined maximum number of episodes for MAMT. We first train all hyperparameter combinations to one-sixth of the maximum number of episodes and select the top-sixth hyperparameter combinations with the best performance (defined in **Performance metric.**). Then we train the selected one-sixth combination to one-third of the maximum number of episodes and select the best-performing one-sixth combination. Finally, we train the remaining combinations to the maximum number of episodes and select the best hyperparameter combination.

Hardware. The hardware used in the experiment is a server with 128G memory and 4 NVIDIA 1080Ti graphics cards with 11G video memory.

The Code of Baselines. The code and license of baselines are shown in following list:

- MADDPG (Lowe et al., 2017): https://github.com/shariqiqbal2810/maddpg-pytorch, MIT License;
- MAAC (Iqbal & Sha, 2019): https://github.com/shariqiqbal2810/MAAC, MIT License;
- MA-PPO: https://github.com/zoeyuchao/mappo, MIT License;
- LOLA (Foerster et al., 2018a): https://github.com/alexis-jacq/LOLA_DiCE and https://github.com/geek-ai/MAgent, MIT License.

Learning curves are smoothed by averaging over a window of 11 epochs. Source code is available at https://anonymous.4open.science/r/MAMT.

Table 1: Default settings of our methods used in experiments.

Name	Default value
num parallel envs	12
step size	from 10,000 to 50,000
num epochs per step	4
steps per update	100
buffer size	1,000,000
batch size	1024
batch handling	Shuffle transitions
num critic attention heads	4
value loss	MSE
modeling policy loss	CrossEntropyLoss
discount	0.99
optimizer	Adam
adam lr	1e-3
adam mom	0.9
adam eps	1e-7
lr decay	0.0
policy regularization type	L2
policy regularization coefficient	0.001
modeling policy regularization type	L2
modeling policy regularization coefficient	0.001
critic regularization type	L2
critic regularization coefficient	1.0
critic clip grad	10 * num of agents
policy clip grad	0.5
soft reward scale	100
modeling policy clip grad	0.5
trust-region decomposition network clip grad	10 * num of agents
trust-region clip	from 0.01 to 100
num of iteration delay in mirror descent	100
tsallis q in mirror descent	0.2
δ in coordination coefficient	0.2

G MORE RESULTS

Figure 12 to Figure 22 show the performance indicators of all agents in 4 environments under different random seeds.

Table 2: Default settings of MAAC used in experiments.

Name	Default value
num parallel envs	12
step size	from 10,000 to 50,000
num epochs per step	4
steps per update	100
buffer size	1,000,000
batch size	1024
batch handling	Shuffle transitions
num critic attention heads	4
value loss	MSE
discount	0.99
optimizer	Adam
adam lr	1e-3
adam mom	0.9
adam eps	1e-7
lr decay	0.0
policy regularization type	L2
policy regularization coefficient	0.001
critic regularization type	L2
critic regularization coefficient	1.0
critic clip grad	10 * num of agents
policy clip grad	0.5
soft reward scale	100

Table 3: Default settings of MADDPG used in experiments.

Name	Default value
num parallel envs	12
step size	from 10,000 to 50,000
num epochs per step	4
steps per update	100
buffer size	1,000,000
batch size	1024
batch handling	Shuffle transitions
value loss	MSE
discount	0.99
optimizer	Adam
adam lr	1e-3
adam mom	0.9
adam eps	1e-7
lr decay	0.0
policy regularization type	L2
policy regularization coefficient	0.001
critic regularization type	L2
critic regularization coefficient	1.0
critic clip grad	10 * num of agents
policy clip grad	0.5

Table 4: Default settings of MAPPO used in experiments.

Name	Default value
num parallel envs	12
step size	from 10,000 to 50,000
batch size	4096
num critic attention heads	4
value loss	Huber Loss
Huber delta	10.0
GAE lambda	0.95
discount	0.99
optimizer	Adam
adam lr	1e-3
adam mom	0.9
adam eps	1e-7
lr decay	0.0
policy regularization type	L2
policy regularization coefficient	0.0
critic regularization type	L2
critic regularization coefficient	0.0
critic clip grad	10
policy clip grad	10
use reward normalization	TRUE
use feature normalization	TRUE

Table 5: Default settings of LOLA(+DQN) used in experiments.

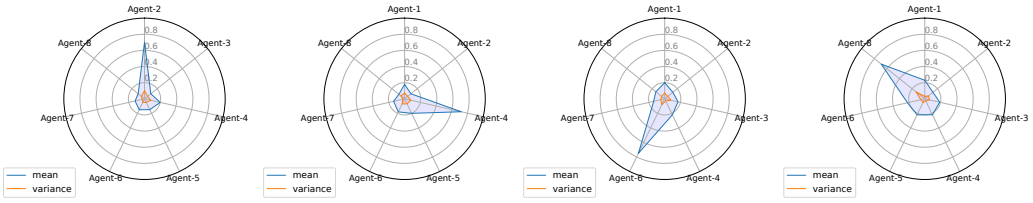
Name	Default value
num parallel envs	12
step size	from 10,000 to 50,000
num epochs per step	4
steps per update	100
buffer size	1,000,000
batch size	1024
batch handling	Shuffle transitions
value loss	MSE
discount	0.99
optimizer	Adam
adam lr	1e-3
adam mom	0.9
adam eps	1e-7
lr decay	0.0
regularization type	L2
regularization coefficient	1.0
clip grad	10 * num of agents

Table 6: Tuning ranges of key hyperparameters of MAMD in experiments.

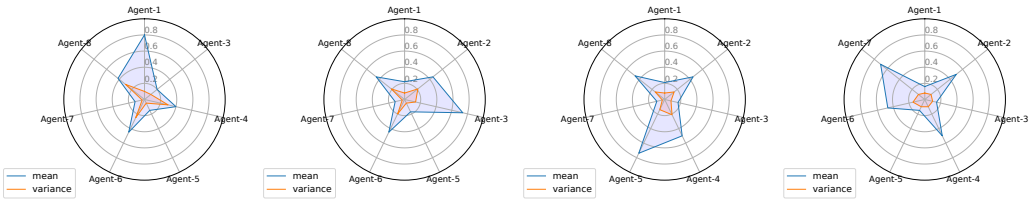
Name	Range
num epochs per step	{1, 4, 8}
adam lr	{0.0003, 0.001}
soft reward scale	{10, 100}
num of iteration delay in mirror descent	{100, 1000}

Table 7: Tuning ranges of key hyperparameters of MAMT in experiments.

Name	Range
`num epochs per step`	$\{1, 4, 8\}$
`adam lr`	$\{0.0003, 0.001\}$
`soft reward scale`	$\{10, 100\}$
`trust-region clip`	$\{0.01, 1, 100\}$
`num of iteration delay in mirror descent`	$\{100, 1000\}$
δ `in coordination coefficient`	$\{0.002, 0.02, 0.2\}$

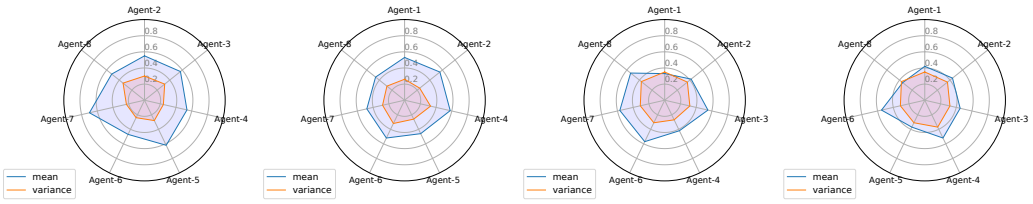


(a) Agent 1. (b) Agent 3. (c) Agent 5. (d) Agent 7.

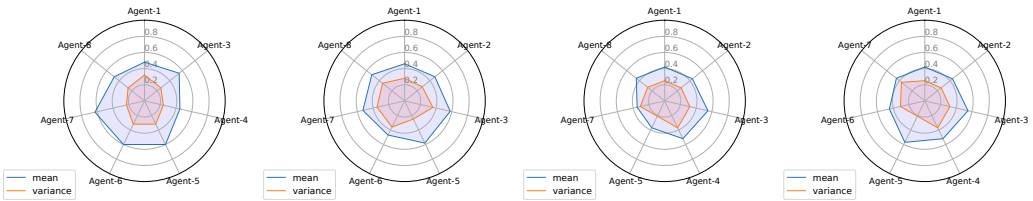


(e) Agent 2. (f) Agent 4. (g) Agent 6. (h) Agent 8.

Figure 12: The mean and variance of coordination coefficient of each agent in *Rover-Tower* environment.

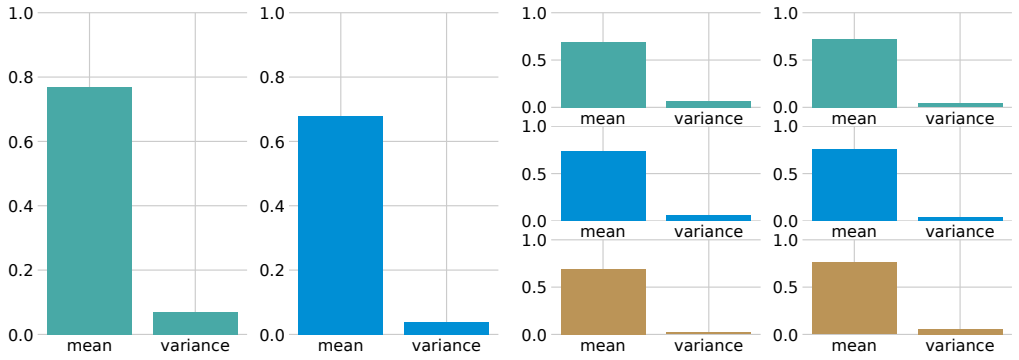


(a) Agent 1. (b) Agent 3. (c) Agent 5. (d) Agent 7.



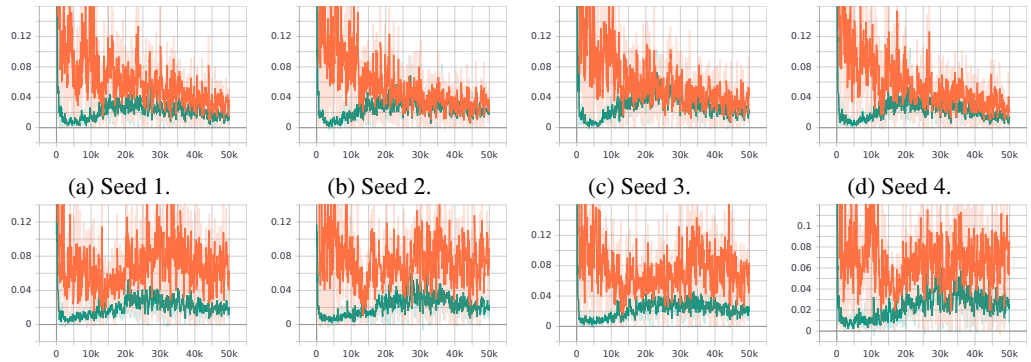
(e) Agent 2. (f) Agent 4. (g) Agent 6. (h) Agent 8.

Figure 13: The mean and variance of coordination coefficient of each agent in *Pursuit* environment.



(a) Spread environment. (b) Multi-Walker environment.

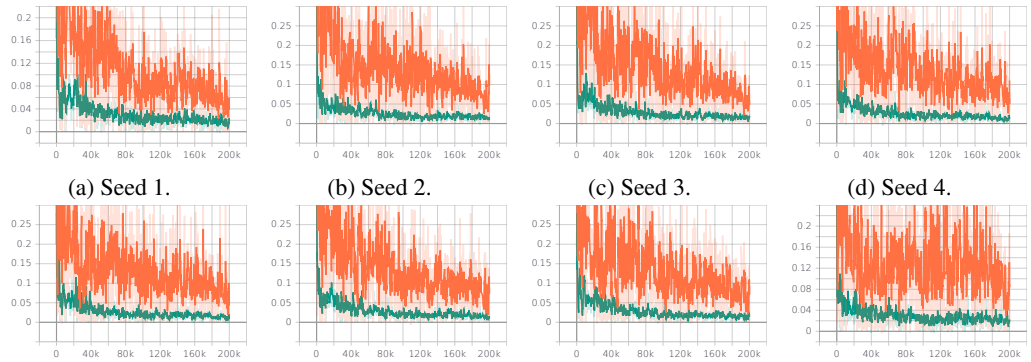
Figure 14: The mean and variance of coordination coefficient of each agent in *Spread* environment and *Multi-Walker* environment. Different color represents different agent.



(a) Seed 1. (b) Seed 2. (c) Seed 3. (d) Seed 4.

(e) Seed 5. (f) Seed 6. (g) Seed 7. (h) Seed 8.

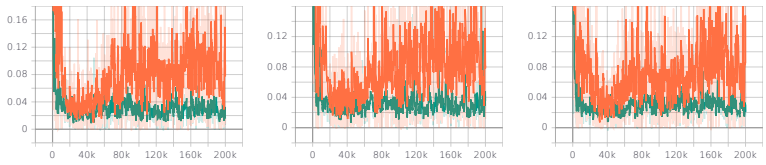
Figure 15: The averaged KL-divergence of each agent in *Rover-Tower* environments. Red line represents MAAC and green line represents MAMT.



(a) Seed 1. (b) Seed 2. (c) Seed 3. (d) Seed 4.

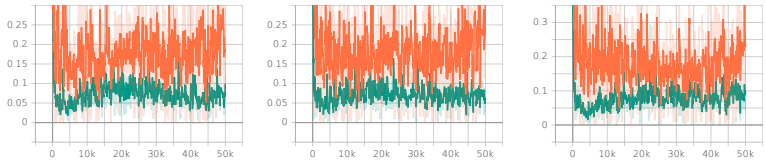
(e) Seed 5. (f) Seed 6. (g) Seed 7. (h) Seed 8.

Figure 16: The averaged KL-divergence of each agent in *Pursuit* environments. Red line represents MAAC and green line represents MAMT.



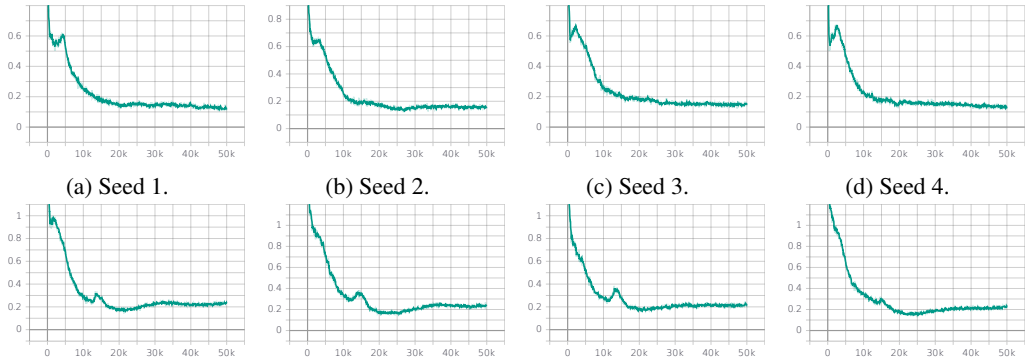
(a) Seed 1. (b) Seed 2. (c) Seed 3.

Figure 17: The averaged KL-divergence of each agent in *Multi-Walker* environments. Red line represents MAAC and green line represents MAMT.



(a) Seed 1. (b) Seed 2. (c) Seed 3.

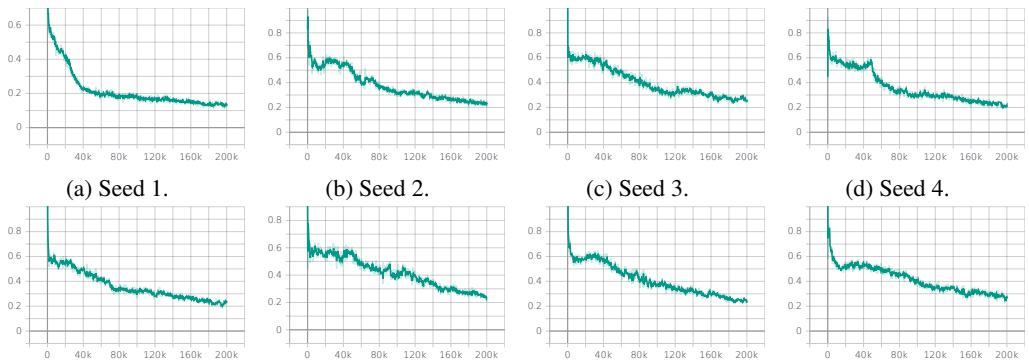
Figure 18: The averaged KL-divergence of each agent in *Spread* environments. Red line represents MAAC and green line represents MAMT.



(a) Seed 1. (b) Seed 2. (c) Seed 3. (d) Seed 4.

(e) Seed 5. (f) Seed 6. (g) Seed 7. (h) Seed 8.

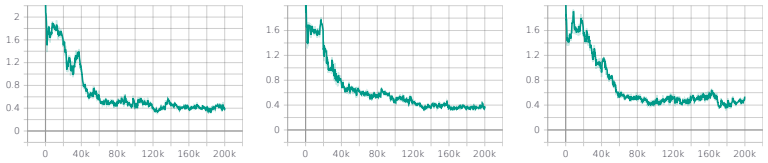
Figure 19: The averaged $\hat{\mathrm{KL}}$ of each agent in *Rover-Tower* environments.



(a) Seed 1. (b) Seed 2. (c) Seed 3. (d) Seed 4.

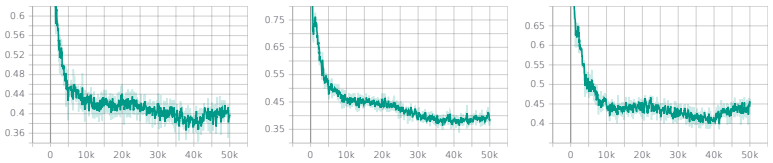
(e) Seed 5. (f) Seed 6. (g) Seed 7. (h) Seed 8.

Figure 20: The averaged $\hat{\mathrm{KL}}$ of each agent in *Pursuit* environments.



(a) Seed 1. (b) Seed 2. (c) Seed 3.

Figure 21: The averaged $\hat{\mathrm{KL}}$ of each agent in *Multi-Walker* environments.



(a) Seed 1. (b) Seed 2. (c) Seed 3.

Figure 22: The averaged $\hat{\mathrm{KL}}$ of each agent in *Spread* environments.

REFERENCES FOR SUPPLEMENTARY MATERIAL

Maruan Al-Shedivat, Trapit Bansal, Yura Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. Continuous adaptation via meta-learning in nonstationary and competitive environments. In *ICLR*, 2018.

Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. In *ICLR*, 2019.

Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

Omar Besbes, Yonatan Gur, and Assaf Zeevi. Stochastic multi-armed-bandit problem with non-stationary rewards. In *NeurIPS*, 2014.

Wang Chi Cheung, David Simchi-Levi, and Ruihao Zhu. Non-stationary reinforcement learning: The blessing of (more) optimism. *Machine Learning eJournal*, 2019.

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.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

Jakob Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. In *AAMAS*, 2018a.

Jakob N. Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip H. S. Torr, Pushmeet Kohli, and Shimon Whiteson. Stabilising experience replay for deep multi-agent reinforcement learning. In *ICML*, 2017.

Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *AAAI*, 2018b.

Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. A theory of regularized markov decision processes. In *ICML*, 2019.

Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *AAMAS*, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018.

Eric A Hansen, Daniel S Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI*, 2004.

Siyue Hu and Jian Hu. Noisy-MAPPO: Noisy advantage values for cooperative multi-agent actor-critic methods. *arXiv preprint arXiv:2106.14334*, 2021.

Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. In *ICML*, 2019.

Max Jaderberg, Wojciech M. Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio García Castañeda, Charlie Beattie, Neil C. Rabinowitz, Ari S. Morcos, Avraham Ruderman, Nicolas Sonnerat, Tim Green, Louise Deason, Joel Z. Leibo, David Silver, Demis Hassabis, Koray Kavukcuoglu, and Thore Graepel. Human-level performance in 3D multiplayer games with population-based reinforcement learning. *Science*, 364:859 – 865, 2019.

Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(51):1563–1600, 2010.

Jiechuan Jiang and Zongqing Lu. Adaptive learning rates for multi-agent reinforcement learning, 2021. URL https://openreview.net/forum?id=yN18f9V1Onp.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *ICML*, 2002.

Jakub Grudzien Kuba, Ruiqing Chen, Munning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. Trust region policy optimisation in multi-agent reinforcement learning. *arXiv preprint arXiv:2109.11251*, 2021.

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

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 *ICLR*, 2016.

Siqi Liu, Guy Lever, Josh Merel, Saran Tunyasuvunakool, Nicolas Heess, and Thore Graepel. Emergent coordination through competition. In *ICLR*, 2019.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *NeurIPS*, 2017.

Weichao Mao, Kaiqing Zhang, Ruihao Zhu, David Simchi-Levi, and Tamer Başar. Near-optimal model-free reinforcement learning in non-stationary episodic mdps. In *ICML*, 2021.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Trust-pcl: An off-policy trust region method for continuous control. In *ICLR*, 2018.

Georgios Papoudakis, Filippos Christianos, A. Rahman, and Stefano V. Albrecht. Dealing with non-stationarity in multi-agent deep reinforcement learning. *ArXiv*, abs/1906.04737, 2019.

Neil Rabinowitz, Frank Perbet, Francis Song, Chiyuan Zhang, SM Ali Eslami, and Matthew Botvinick. Machine theory of mind. In *ICML*, 2018.

Roberta Raileanu, Emily Denton, Arthur Szlam, and Rob Fergus. Modeling others using oneself in multi-agent reinforcement learning. In *ICML*, 2018.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *ICML*, 2015.

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

Lior Shani, Yonathan Efroni, and Shie Mannor. Adaptive trust region policy optimization: Global convergence and faster rates for regularized mdps. In *AAAI*, 2020.

Yuhang Song, Jianyi Wang, Thomas Lukasiewicz, Zhenghua Xu, Mai Xu, Zihan Ding, and Lianlong Wu. Arena: A general evaluation platform and building toolkit for multi-agent intelligence. In *AAAI*, 2020.

Justin K Terry, Benjamin Black, Mario Jayakumar, Ananth Hari, Luis Santos, Clemens Dieffendahl, Niall L Williams, Yashas Lokesh, Ryan Sullivan, Caroline Horsch, and Praveen Ravi. PettingZoo: Gym for multi-agent reinforcement learning. *arXiv preprint arXiv:2009.14471*, 2020a.

Justin K Terry, Nathaniel Grammel, Ananth Hari, Luis Santos, and Benjamin Black. Revisiting parameter sharing in multi-agent deep reinforcement learning. *arXiv preprint arXiv:2005.13625*, 2020b.

Gerald Tesauro. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3):58–68, 1995.

Manan Tomar, Lior Shani, Yonathan Efroni, and Mohammad Ghavamzadeh. Mirror descent policy optimization. *arXiv preprint arXiv:2005.09814*, 2020.

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

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *NeurIPS*, 2017.

Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre M. Bayen, and Yi Wu. The surprising effectiveness of MAPPO in cooperative, multi-agent games. *ArXiv*, abs/2103.01955, 2021.