META PROXIMAL POLICY OPTIMIZATION FOR COOPERATIVE MULTI-AGENT CONTINUOUS CONTROL

Boli Fang

Submitted to the faculty of the University Graduate School

in partial fulfillment of the requirements

for the degree

Master of Science

in the Luddy School of Informatics, Computing and Engineering,

Indiana University

May 2022

Accepted by the Graduate Faculty, Indiana University, in partial fulfillment of the requirements for the degree of Master of Science.

Master's Thesis Committee

David Crandall, Ph.D

Qin Zhang, Ph.D

Date of Defense: 05/15/2022

To my parents and my friends.

ACKNOWLEDGEMENTS

First of all, I'd like to express my most sincere gratitude towards my advisors Prof David Crandall and Prof Qin Zhang for this Master thesis. Without their unwavering support, continuous encouragement and pertinent advice, it would have been impossible for me to embark on any kind of research. The working ethic and dedication to research I have learned will always exert influence over my career and life in general, and I am forever grateful.

I'd also like to thank my collaborators Zhenghao Peng (PhD student at Chinese University of Hong Kong) and Hao Sun (PhD student at University of Cambridge, UK), for the fruitful and inspirational collaboration that leads to this thesis. I'd also like to extend my gratitude to Prof Bolei Zhou (now Assistant Professor at University of California, Los Angeles) for his suggestions on the presentation of different sections, as well as his insight over Artificial Intelligence.

I'd like to thank my parents for their continuous encouragement in the course of my studies, as well as the motivation to overcome the challenges in life. I also appreciate all my fellow friends I've met during my journey at IU for their inspiration.

Finally, I'd also like to acknowledge and thank the following sources of funding which have supported me in my academic journey: CCF-1844234, IIS-1633215 and CCF-2006591.

Boli Fang

Meta Proximal Policy Optimization for Cooperative Multi-Agent Continuous Control

In this thesis we propose Multi-Agent Proxy Proximal Policy Optimization (MA3PO), a novel multi-agent deep reinforcement learning algorithm that tackles the challenge of cooperative continuous multi-agent control. Our method is driven by the observation that most existing multi-agent reinforcement learning algorithms mainly focus on discrete state/action spaces and are thus computationally infeasible when extended to environments with continuous state/action spaces.

To address the issue of computational complexity and to better model intra-agent collaboration, we make use of the recently successful Proximal Policy Optimization algorithm that effectively explores of continuous action spaces, and incorporate the notion of *intrinsic motivation* via *meta-gradient methods* so as to stimulate the behavior of individual agents in cooperative multi-agent settings. Towards these ends, we design proxy rewards to quantify the effect of individual agent-level intrinsic motivation onto the team-level reward, and apply meta-gradient methods to leverage such an addition with a learning-to-learning optimization paradigm so that our algorithm can learn the team-level cumulative reward effectively.

Furthermore, we have also conducted experiments on various open multi-agent reinforcement learning benchmark environments with continuous action spaces. Our results demonstrate that our meta proximal policy optimization algorithm is not only comparable with other existing state-of-the-art algorithmic benchmarks in terms of performances, but also significantly reduces training time complexity as compared to existing techniques.

David Crandall, Ph.D

Qin Zhang, Ph.D

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

We investigate in this paper the problem of continuous control in cooperative multi-agent games. Our motivation comes from the observations that many complex real-world scenarios such as multi-player video games [1], vehicle/robotics control [2, 3] and network routing [4] can naturally be modeled as cooperative multi-agent games, under which the objective is to maximize team-level cumulative returns. Naive extensions of single-agent RL algorithms that apply a single actor to operate on joint action spaces are inherently infeasible due to poor scalability. Alternatively, when the agents are independently trained with little or no communication, the non-stationary global environment incurred by simultaneous exploration without coordination also lead to suboptimal learning. Thus, ways to reduce action space complexity would entail individual agents deploying *decentralized* policies that act only on their individual observations, while allowing for coordination within the team via a centralized controller. Towards this goal, a standard paradigm called *centralized training with decentralized execution* (CTDE) [5] is widely adopted. Under CTDE, individual agents are mainly concerned with policy execution, with team-level policy inference and coordination handled by a centralized decision maker. Numerous designs from recent developments in Multi-Agent Reinforcement Learning [6, 7, 8, 9, 10] have demonstrated the effectiveness of the CTDE structure.

Despite these successes, several challenges remain notable. First of all, most existing CTDE algorithms are designed for *discrete* action spaces and involves extensive action enumeration, making these algorithms inapplicable to continuous action spaces. Additionally, relatively few prior work achieves feasible cooperative control, since existing strategies involve either explicit action enumeration [11] or pre-specified oracle [12] for value function computation, both of which are often impossible to obtain in practice.

To address these shortcomings, we propose in this work multi-agent proxy proximal policy optimization (MA3PO), a multi-agent algorithm motivated by proximal policy optimization (PPO) [13], as well as *intrinsic motivation* in general MARL. We incorporate proxy rewards to model how local observations and the current actions indirectly influence agent impact on team performance, and apply meta-gradient methods to learn the optimal team-level policy by first finding the optimal parameters for intrinsic reward and use these parameters to optimize policy networks. Following the CTDE framework, our algorithm collects episodes from individual decentralized agent actors and uses a centralized critic that takes these episodes as input for policy optimization, using proximal policy search to reduce runtime complexity. In sum, our contributions can be summarized as follows:

1. We propose MA3PO, a multi-agent reinforcement learning algorithm which effectively utilizes episodes with a centralized critic and decentralized actors to achieve collaboration amongst the agents in case of continuous control.

2. Inspired by recent developments of intrinsic motivation in MARL, our MA3PO algorithm constructs for each agent an *intrinsic reward* from original rewards and state-action pairs. Such a design indirectly facilitates collaboration amongst agents, as it better reflects the role of individual agents in group level rewards. Such design facilitates implicit credit assignment, as demonstrated by ablation studies.

3. We conduct extensive experiments on novel benchmarks in continuous multi-agent reinforcement learning settings. Our results suggest that MA3PO works favorably compared to other benchmarks across different environments, while reducing runtime substantially.

CHAPTER 2

RELATED WORK

2.0.1 Continuous Control in Reinforcement Learning

Much work has been done in single-agent continuous control, with most involving policy gradient methods. For deterministic methods, prior examples include Deterministic Policy Gradient(DPG) [14], Deep DPG(DDPG) [15], Twin-Delayed DDPG(TD3) [16]. These methods seek to output actions with fixed policy gradients during objective optimization, at the expense of slow/unstable training. For stochastic methods, existing methods such as A3C [17], TRPO [18], PPO [13] use advantage functions to estimate the Q function with only state as input. The PPO algorithm, in particular, has achieved notable successes recently by limiting policy search into a clipped ‘trust-region’ to reduce runtime and increase robustness. Other developments such as PPG [19] and SAC [20] require additional requirements such as auxiliary tasks (for PPG) and carefully designed reward function(for SAC) to achieve satisfactory performances.

2.0.2 Cooperative Multi-Agent RL

Our work is also related to existing literature in cooperative multi-agent reinforcement learning. Prior work on achieving cooperative in MARL can be divided into two categories: explicit credit assignment and implicit credit assignment. Explicit credit assignment methods directly attribute to each agent its contribution to the overall team behavior, with provable optimal guarantees but low feasiblity in practice. Examples include COMA [11], which applies a centralized critic to estimate the counterfactual advantage of given actions, and SQDDPG [12], which determines contribution of individual agents from an oracle. Implicit credit assignment methods, on the contrary, assign credit indirectly by learning objective decomposition from team-level reward signal. Examples include value-based methods such as VDN [6], QMIX [7], QTRAN [8] and W-QMIX [9], and policy-gradient methods such as LICA [21], MADDPG [10], LIIR [22], MAAC [23]. These works all suffer from varying degrees of constraints in value/policy representation as well as low runtime efficiency. Recent works such as MAMBPO [24] and MAPPO [25] have investigated the possibility of multi-agent policy optimization for inter-agent collaboration on multi-agent discrete settings, with MAPPO [25] being the closest to our paper. Despite these efforts, designing efficient RL algorithms for multi-agent continuous control remains an open question, as pointed out by [25].

2.0.3 Intrinsic Reward Design in RL

Our work is also related to intrinsic reward design [26] in reinforcement learning. Prior works such as [27] have applied heuristics to compute the intrinsic reward. More recently, [26] proposes to parametrize intrinsic reward and update the parameters of intrinsic reward and policy alternately, and [22] suggests that individual intrinsic reward is a helpful indicator to measure agent contribution to the overall team during learning.

CHAPTER 3
BACKGROUND

3.0.1 Problem Setting and Notations

Following the standard notations in prior work on multi-agent reinforcement learning, we model the interactions and exploration amongst the agents as a Decentralized Partially Observable Markov Decision Process(DEC-POMDP) [28]. We describe a DEC-POMDP by a tuple $G =< S, U, P, r, Z, O, n, \gamma >$, where S stands for the set of states of environment, U the set of actions, P the transition probability, r the reward function, Z the set of observations that an agent can observe, O the observation function which describes what an agent actually observes while operating in environment, n the number of agents, and $\gamma \in [0, 1)$ the discount factor.

At each time step t, given the true state of the environment $s_t \in S$, each agent $a \in A = \{1, 2, ..., n\}$ chooses an action $u_t^a \in U$, and the actions undertaken by all agents form a joint action $\mathbf{u}_t \in U^n$. This $\mathbf{u}_t$ causes a transition on the environment according to the state transition function $P(s|s_t, \mathbf{u}_t) : S \times U^n \times S \to [0, 1]$. All agents share the same reward function $r(s_t, \mathbf{u}_t) : S \times U^n \to \mathbf{R}$ at all times t. The state of the external environment s_t is only partially observable to the agents: for each agent a operating under state s_t, agent a obtains an individual observation $z_t^a \in Z$ by an observation function $O(s_t, a) : S \times A \to Z$. Each agent a has an action-observation history(also called a *trajectory*):

$$\tau^a = \{(u_1^a, z_1^a), (u_2^a, z_2^a), (u_3^a, z_3^a), ...(u_t^a, z_t^a)\},$$

on which it conditions a stochastic policy $\pi^a(u_t^a|z_t^a)$ that takes value in $[0, 1]$. All the π^a's form a joint policy π. To facilitate our subsequent discussion, we also make use of the following standard definitions of the state-action value function Q_{π^a}, the value function

V_{π^a}, and the advantage function A_{π^a}, all defined below with respect to the current policy π^a undertaken by agent a:

$$\begin{aligned}
Q_{\pi^a}(s_t, u_t^a) &= \mathbb{E}_{s_{t+1:\infty}, u^a_{t+1:\infty}}[R_t | s_t, u_t^a], \\
V_{\pi^a}(s_t) &= \mathbb{E}_{s_{t+1:\infty}, u^a_{t:\infty}}[R_t | s_t, u_t^a], \\
A_{\pi^a}(s_t, u_t^a) &= Q_{\pi^a}(s_t, u_t^a) - V_{\pi^a}(s_t), \\
\text{where } u_t^a &\sim \pi^a(u_t^a | s_t), s_{t+1} \sim P(s_{t+1} | s_t, u_t^a), t \geq 0.
\end{aligned}$$

Here $R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$ is the discounted return and the expectation of R_t is taken with respect to all future states of environments and future joint actions starting from the time step $t+1$. Similarly, at every time step t, we also denote $Q_\pi(s_t, \mathbf{u}_t), V_\pi(s_t), A_\pi(s_t, \mathbf{u}_t)$ as the respective corresponding value functions for the joint policy π and joint action $\mathbf{u}_t$.

3.0.2 Proximal Policy Optimization (PPO)

Policy gradient methods [29] optimize policies directly through policy parametrization into the optimization objective. They are highly useful in continuous space/action settings, since they avoid expensive action enumeration involved in their value-based counterparts. Prior policy gradient methods [29, 30] mostly center on directly optimizing expected cumulative reward via gradient estimates, but as suggested by [18, 13], additional considerations such as robustness and stability in training are also important. Towards these goals, a general rule is to restrict policy parameter updates to a bounded region called *trust region* so as to ameliorate the instability caused by the asynchronicity between the rollout process(the process for episode collection) and the optimization process (the process for policy update). In particular, *Trust Region Policy Optimization*(TRPO) [18], by restricting policy parameter updates to a bounded *trust region*, has been demonstrated to achieve stable training, with theoretical guarantees for improvement over policy iteration. Towards such goals, *Proximal*

Policy Optimization(PPO) [13] further improves TRPO by 'clipping' the trust region in the policy parameter search to avoid potentially large parameter changes. Given an agent a, its partial observation z_t^a, the old known behavior policy parameters θ_{old}^a before update, the current policy parameters θ^a being learnt and the corresponding old/new policy ratio $c_t(\theta^a) = \frac{\pi_{\theta^a}(u_t^a|z_t^a)}{\pi_{\theta_{\text{old}}^a}(u_t^a|z_t^A)}$, the objective function of *Proximal Policy Optimization* (PPO) [13] can be written as

$$J^{\text{PPO}}(\theta^a) = \mathbb{E}[\min(c_t(\theta^a)\hat{A}_{\theta_{\text{old}}^a}, \text{clip}(c_t(\theta^a), 1-\epsilon, 1+\epsilon)\hat{A}_{\theta_{\text{old}}^a})], \tag{3.1}$$

in which $c_t(\theta^a)$ is clipped into $(1-\epsilon, 1+\epsilon)$, with ϵ a hyperparameter. The PPO objective function also uses minimum between the original TRPO objective and the clipped objective, thus penalizing extreme values of $c_t(\theta^a)$ at any time t.

Furthermore, the PPO algorithm is implemented in an actor-critic framework: a policy network actor θ and a value network critic ϕ. The value network ϕ estimates the advantage $\hat{A}_{\theta_{\text{old}}^a}$, and the policy network θ learns policies via policy gradient. The PPO objective could be further regularized by value-estimator errors and policy-level entropy [13], so as to properly represent the value functions and encourage exploration during the learning process.

3.0.3 Proxy Reward: Quantifying Intrinsic Motivation

To better take the contributions of individual agents into account of team-level coordination, *parametrized intrinsic reward*, first proposed by [22], quantifies individual agent's intrinsic motivation as a function of global state and current agent action. For each agent a at time t, given the local observation-action pair (z_t^a, u_t^a), agent a is assigned a *proxy* reward $(r^{\text{proxy}})_t^a$ consisting of the original external team-level r_t and a parameterized intrinsic reward $r_t^{\text{in}} = \eta^a(z_t^a, u_t^a)$ with parameters η^a:

$$(r^{\text{proxy}})_t^a = r_t + \lambda r_t^{\text{in}} = r_t + \lambda\eta^a(z_t^a, u_t^a), \tag{3.2}$$

where λ is a hyperparameter to balances intrinsic motivation and external awards during learning. The motivation is that the parameterized intrinsic reward η_a corresponding to each agent a will be updated in a way that the corresponding expected PPO objective will be maximized. Similar to the cases in [26, 22], one can accordingly define for each agent a the discounted cumulative proxy reward, the proxy objective functions, the proxy value function, with respect to proper time horizons and based on local observations, as follows:

$$\begin{aligned} R_t^{\text{proxy}} &= \sum_{i=0}^{\infty} \gamma^i (r_{t+i} + \lambda r_{t+i}^{\text{in}}), \\ V_a^{\text{proxy}}(z_t^a) &= \mathbb{E}_{z_{t+1:\infty}^a, u_{t:\infty}^a}[R_t^{\text{proxy}} | z_t^a, u_t^a], \\ J_a^{\text{proxy}} &= \mathbb{E}_{z_{0:\infty}^a, u_{0:\infty}^a}[R_t^{\text{proxy}}]. \end{aligned} \tag{3.3}$$

CHAPTER 4

MA3PO: MULTI-AGENT ADAPTATION OF PPO WITH PROXY REWARD

In this section we will introduce the details of our MA3PO algorithm. To incorporate the proxy reward into learning, We start off by defining our optimization objectives accordingly. The parameters of our PPO objective $J^{\text{PPO}}(\theta^a)$ should be updated to include intrinsic reward parameters η^a and policy parameters θ^a. We hence consider a bi-level optimization program as in [22], where for agent a, η^a is updated to maximize the PPO objective J, and θ^a is chosen such that the proxy objective functions are maximized:

$$\max_{\theta^a, \eta^a} J^{\text{PPO}}(\eta^a) \qquad s.t. \qquad \theta^a = \underset{\theta}{\text{argmax}} J_a^{\text{proxy}}(\theta, \eta^a). \tag{4.1}$$

Both functions are optimized by policy gradient methods, with different learning rates for the original PPO objective and the proxy objectives. Details of training are described in the following section.

We now describe the algorithm for the bi-level optimization. As in the case with general actor-critic algorithms, our training process consists of two components: centralized value network with parameters ϕ and decentralized agent-level policy networks with parameters θ^a for each agent a. During each training iteration, all θ^a's are first updated by their own policy gradient algorithms with their own proxy critics. For each agent a, given a trajectory from policy π_{θ^a}, we fix η^a, and update the policy gradient for θ^a as follows by the policy gradient theorem [29]:

$$\nabla_{\theta^a} J_a^{\text{proxy}} = \nabla_{\theta^a} \log \pi_{\theta^a}(u_t^a | z_t^a) A_a^{\text{proxy}}(z_t^a, u_t^a), \tag{4.2}$$

where

$$A_a^{\text{proxy}}(z_t^a, u_t^a) = (r^{\text{proxy}})_t^a + V_a^{\text{proxy}}(z_t'^a) - V_a^{\text{proxy}}(z_t^a) \tag{4.3}$$

is the proxy advantage function computed from the proxy value function $(V^{\text{proxy}})^a$, with $z_t'^a$ denoting the updated observation in the one-step rollout trajectory. Once θ^a is updated with learning rate α:

$$\theta^a_{\text{upd}} = \theta^a + \alpha \nabla_{\theta^a} J_a^{\text{proxy}}(\theta^a), \tag{4.4}$$

we apply chain rule and build the connection between η and the original PPO objective J as follows:

$$\nabla_{\eta^a} J^{\text{PPO}} = \nabla_{\theta^a_{\text{upd}}} J^{\text{PPO}} \nabla_{\eta^a} \theta^a_{\text{upd}}. \tag{4.5}$$

Notice that (4.5) can be seen as an instance of meta-gradient methods [31, 32] in which the η^a-gradient on J^{PPO} is modeled by the η^a-gradient on the updated parameters θ^a_{upd}. The gradient $\nabla_{\theta^a_{\text{upd}}} J^{\text{PPO}}$ can be estimated as follows [33]:

$$\begin{aligned}\nabla_{\theta^a_{\text{upd}}} J^{\text{PPO}} =& (\mathbf{1}_{|c_t(\theta^a_{\text{upd}})-1|<\epsilon} \vee \mathbf{1}_{\text{sgn}(c_t(\theta^a_{\text{upd}})-1)\neq \text{sgn}(\hat{A}_{\theta^a_{\text{old}}})}) \\ & c_t(\theta^a_{\text{upd}}) \nabla_{\theta^a} \log \pi_{\theta^a_{\text{upd}}}(u_t^a | z_t^a) \hat{A}_{\theta^a},\end{aligned} \tag{4.6}$$

where $\hat{A}_{\theta^a}$ is the General Advantage Estimates(GAE) [34] of each agent a corresponding to rollout trajectory of π_{θ^a}. The other term $\nabla_{\eta^a} \theta^a_{\text{upd}}$ can be estimated as [22]:

$$\begin{aligned}\nabla_{\eta^a} \theta^a_{\text{upd}} &= \nabla_{\eta^a} (\theta^a + \alpha \nabla_{\theta^a} J_a^{\text{proxy}}(\theta^a)) \\ &= \alpha \lambda \nabla_{\theta^a} \log \pi_{\theta^a}(u_t^a | z_t^a) \nabla_{\eta^a} r_a^{\text{proxy}}(z_t^a, u_t^a),\end{aligned} \tag{4.7}$$

either with samples generated by θ^a_{upd}, or with importance sampling(IS) as in [22] so that computation of $\nabla_{\eta^a} J^{\text{PPO}}$ does not require new samples generated from θ^a_{upd}. η^a is then updated with learning rate β similar to the case in (4.5). We also adopt standard training techniques in previous PPO literature [13, 25, 33]. In each iteration, each of the n actors collects the same number T of episodes. We construct J^{PPO} from these nT episodes, and alternatively update the policy and intrinsic reward parameters with corresponding gradients. Algorithm 1 illustrates the whole optimization process.

```
Algorithm 1 MA3PO
Input: Policy learning rate $\alpha$, intrinsic reward learning rate $\beta$.
Output: Policy parameters $\theta^a$'s and intrinsic reward parameters $\eta^a$'s.
Initialize $\theta^a$'s and $\eta^a$'s for each agent $a$.
while termination not reached do
    for each actor $a$: 1,2,...$n$ do
        Run $\pi_{\theta^a}$ for $T$ episodes to obtain rollout trajectories;
        Compute $A_a^{\text{proxy}}$ from (4.3);
        Obtain General Advantage Estimates(GAE) $\hat{A}_{\theta^a}$;
        Obtain $\theta^a_{\text{upd}}$ from $\theta^a$ with (4.4);
        Compute $\nabla_{\theta^a_{\text{upd}}} J^{\text{PPO}}$ from (4.6) on $\pi_{\theta^a_{\text{upd}}}$ trajectories;
        Compute $\nabla_{\eta^a} \theta^a_{\text{upd}}$ from (4.7) via $\pi_{\theta^a_{\text{upd}}}$ trajectories or $\pi_{\theta^a_{\text{old}}}$ trajectories with IS;
        Obtain $\eta^a_{\text{upd}}$ from $\eta^a$ with (4.5);
    end
end
```

4.0.1 Centralized Critic and Decentralized Actors

Our MA3PO algorithm is implemented with a multi-agent actor-critic framework in which agent-level policies are learned via decentralized actors and team-level value functions are learned via a centralized critic. More specifically, for each agent a, there is a decentralized actor that learns policy $\pi_{\theta^a}(u_t^a|z_t^a)$, intrinsic rewards $\eta^a(z_t^a, u_t^a)$ via a local proxy critic φ^a. On the team level, there is a centralized critic ϕ that learns shared values such as the General Advantage Estimate(GAE) $\hat{A}_{\theta^a}$, used by all agents during their own policy. Updates on intrinsic reward is done via back propagation from ϕ to η^a. Figure 4.1 illustrates the different components within the overall MA3PO architecture.

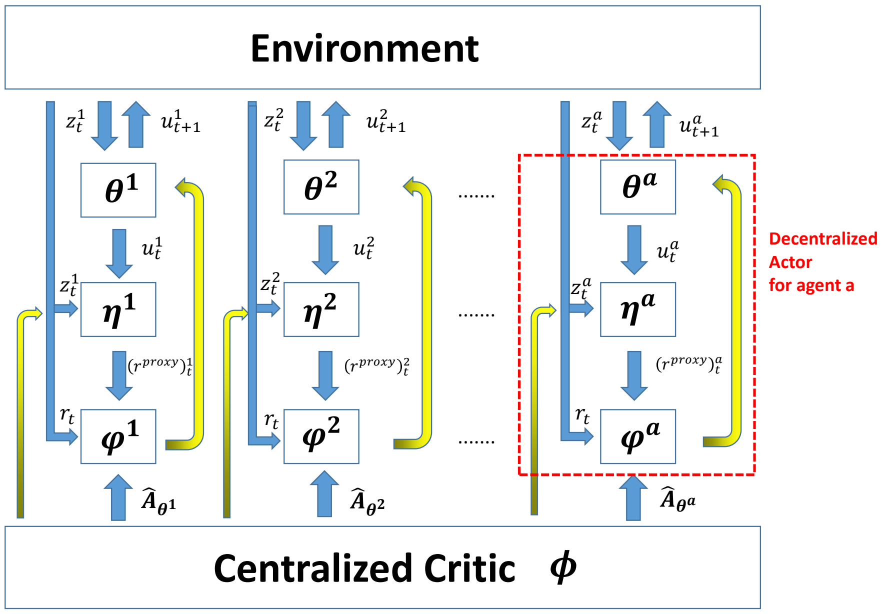

Figure 4.1: MA3PO architecture. The yellow arrows indicate directions of back propagation during training.

CHAPTER 5
EXPERIMENTS

In this section we present the results of our MA3PO algorithm on two recently proposed multi-agent continuous reinforcement learning environment benchmarks with continuous action spaces: Continuous Predator-Prey and Multi-Agent Mujoco(MaMujoco) [35]. Overall, the experiment results demonstrate the effectiveness of our MA3PO algorithm, with parameterized intrinsic motivation effectively stimulating team-level cooperative behavior.

5.0.1 Environment descriptions

Continuous Partially-Observable Predator-Prey

The first environment we consider is the *simple tag* environment introduced by [36], a partially-observable variant of the classical predator-prey game described in [10]. Three slower cooperating agents, each with continuous movement, aim to catch a faster circular prey on a 2D plane with two large landmarks blocking the way. To make the environment purely cooperative, the prey's policy is represented by a hard-coded heuristic that moves the prey to the sampled position with the largest distance to the closest predator, and reward is given every time a predator agent collides with the prey. In addition, a *view radius* is introduced to restrict the agents from receiving any kind of information(landmarks, other agents, prey) outside the range defined by the radius. More detailed descriptions of the environment can be found in [35].

We fix the number of training episodes at 10^6 for all the algorithms, and plot cumulative rewards of each algorithm against the number of episodes and wall time.

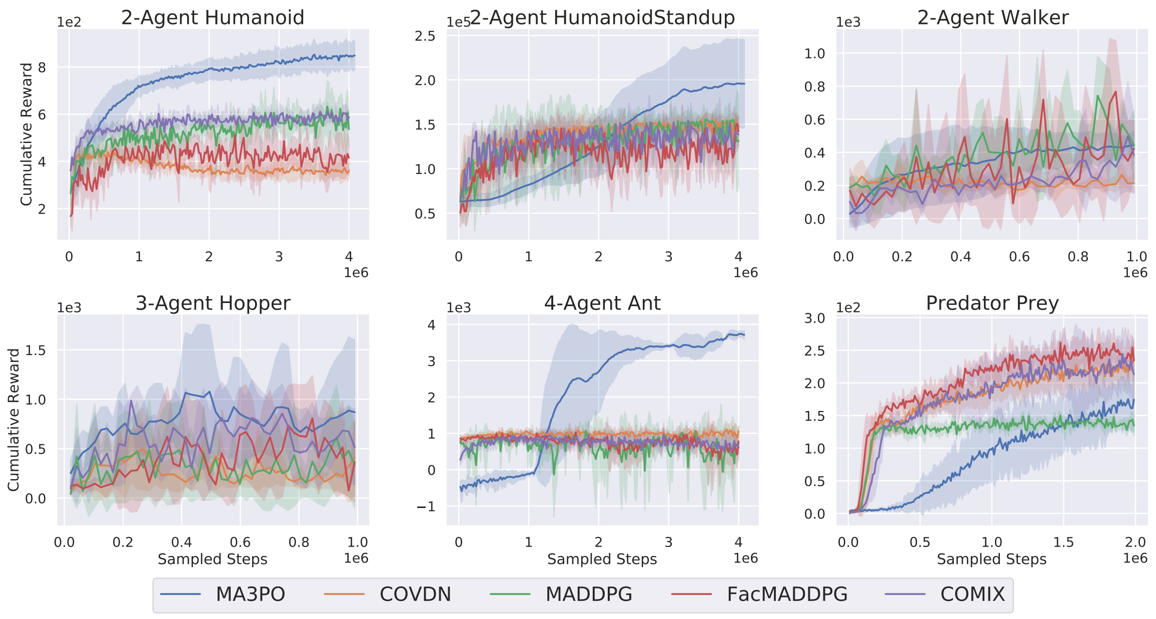

Figure 5.1: Cumulative Rewards vs Sampled Steps for all environments. The mean across 5 seeds is plotted, with the shaded areas denoting the 95% confidence interval for each curve.

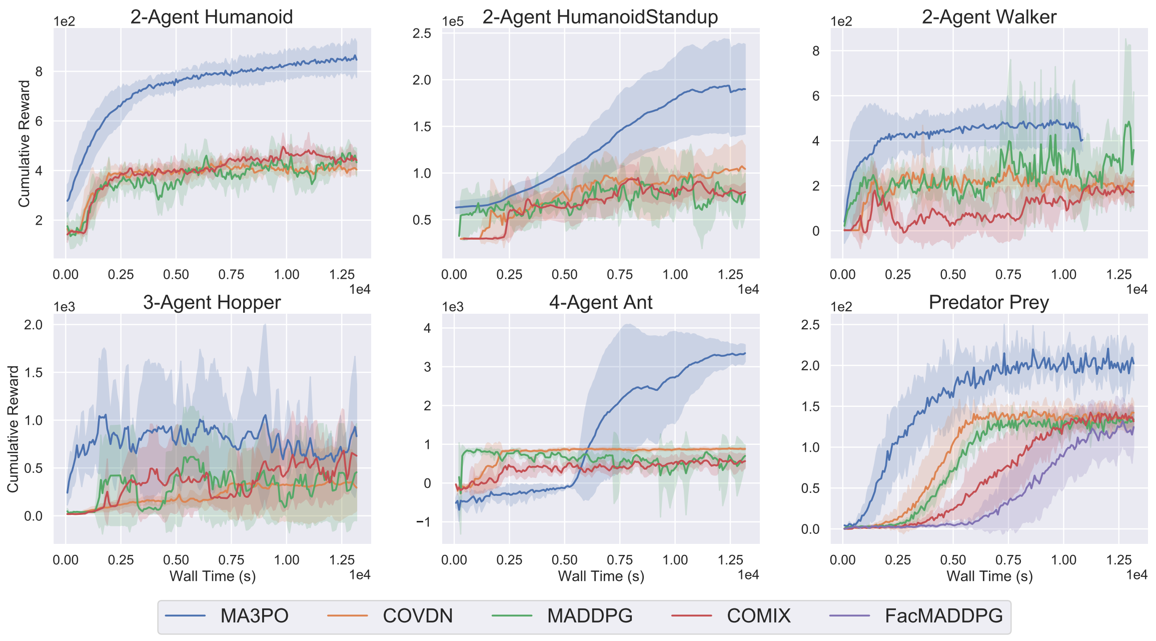

Figure 5.2: Cumulative Rewards vs Wall time(s) for all environments. The mean across 5 seeds is plotted, with the shaded areas denoting the 95% confidence interval for each curve.

MaMujoco: Multi-agent Mujoco

Multi-Agent Mujoco(MaMujoco) [35] is a new benchmark for decentralised cooperative continuous multi-agent robotic control. Based on the fully observable single-agent robotic control benchmark Mujoco [37], MaMujoco includes novel scenarios in which multiple agents within a single robot solve cooperative tasks. This setting is implemented via a *body graph*, under which vertices(joints) are connected by adjacent edges(body segments). The motivation of multi-agent setup in a single robot arises from partial observability incurred due to latency, bandwidth and noisy sensors within a single robot. The resulting lack of access to global state from each joint makes it particularly important to introduce separate agents for local decision making, as such a design makes the overall system more robust and flexible. Such a setup also facilitates comparisons with existing MARL algorithms, both in fully observable settings and in partially observable/low-bandwidth settings [35]. More detailed descriptions of the environment can be found in [35]. For better illustration and easier visualization, we mainly focus on the following multi-agent MaMujoco environments in our subsequent experiments: 2-Agent Humanoid, 2-Agent HumanoidStandup, 2-Agent Walker, 3-Agent Hopper and 4-Agent Ant.

Similar to the case of the Continuous Predator-Prey environment, we fix the number of episodes at 4×10^6 to compare the performances of different algorithms. The benchmarks we compared MA3PO with are identical to those in predator-prey environment.

5.0.2 Experiment Setup

We compare MA3PO against the following state-of-the-art algorithms for cooperative multi-agent continuous control problems:

1. COVDN [35]: COVDN is a novel variant of the established VDN method. The joint action-value Q function is factored as an additive sum of agent-level action-value Q function. Actions are greedily selected with the *cross-entropy method*(CEM) [38] to

approximate optimality in continuous settings.

2. COMIX [35]: COMIX is a novel variant of the established QMIX method onto continuous settings. As in the case of COVDN, CEM is used to approximate greedy action selection, but the action-value function is factored as a deep neural network mixture that increases monotonically with respect to each agent-level action-value Q function.

3. MADDPG [10]: Multi-agent deep deterministic policy gradient(MADDPG) is an extension of the DDPG algorithm onto multiple agents. It learns a centralized critic and a separate actor for each agent, and is applicable to arbitrary reward functions.

4. FacMADDPG [35]: FacMADDPG is a variant of MADDPG with all agents sharing a single centralized critic that is factored as a monotonic network taking agent-level action-value Q functions and global state s as input.

We implement all the algorithms from the RLLib [39] library, a distributed reinforcement learning framework that allows for easy scaling through parallelization and distributed computing. For each set of experiments we have thus described, we host 5 concurrent trials with an NVIDIA GeForce RTX 2080Ti GPU. Each trial consumes 1 CPU with 2 parallel rollout workers, and these trials are then averaged across all trials to produce the cumulative reward (with respect to running mean of the latest 100 episodes) curves for comparison. To reduce influence of possible randomization, we plotted the running mean across 5 random seeds, with the shaded areas denoting the 95% confidence interval for each curve.

To facilitate visual comparison between different algorithms, we fix the number of sampled steps(interactions with the environment) at 2×10^6 for the Continuous Predator-Prey environment, at 4×10^6 for the 2-Agent Humanoid, 2-Agent Humanoidstandup, 4-Agent Ant environments, and at 10^6 for 2-Agent Walker and 3-Agent Hopper in consideration. To produce the most robust curves for detailed analysis, we also set the regularizer of intrinsic reward λ as 10^{-2}. The intrinsic reward learning rate β is set as 10^{-3}, and the policy learning

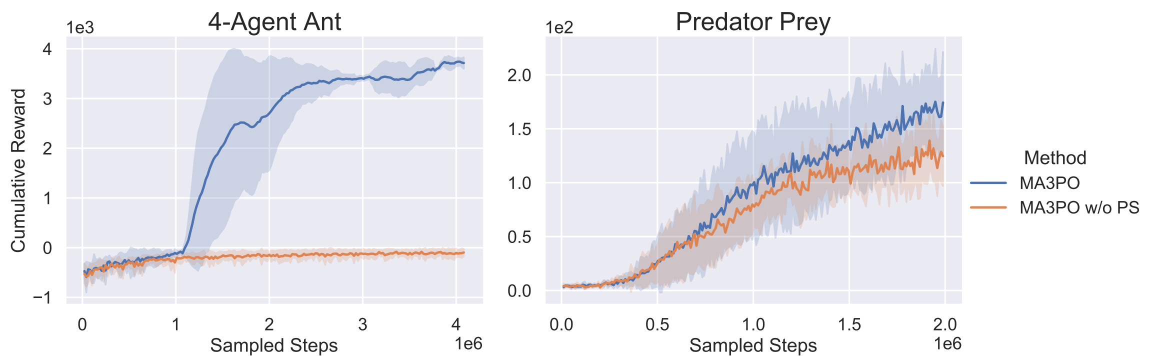

Figure 5.3: Example Ablation studies on Parameter Sharing

rate α is set as 10^{-2} for Predator Prey and 10^{-3} for MaMujoco tasks. For all experiments, we leverage neural networks with ReLU activation and one hidden layer with 64 neurons for policy networks and value networks. The hyperparameters of all other benchmark algorithms follow those described in [35].

5.0.3 Results

We plot cumulative rewards against the sampled steps(interactions with the environment) and wall time(the raw runtime for which the algorithms are run). Figures 5.1 and 5.2 summarize the results for continuous Predator-Prey and Mamujoco environments.

As measured by cumulative reward against sampled steps in Figure 5.1, our MA3PO algorithm consistently performs the best in the 2-Agent Humanoid, 2-Agent Humanoid-Standup, 3-Agent Hopper and 4-Agent Ant environments, with less volatile learning curves going monotonically upwards as compared to other existing benchmarks. For 2-Agent Walker and Predator-Prey, MA3PO yields comparatively less optimal outcome due to lack of episode usage in on-policy training, but the levels of volatility are lower than other benchmark algorithms. Moreover, as demonstrated by the cumulative reward vs runtime plot in Figure 5.2, MA3PO, with its efficient use of episodes, reaches substantially higher levels of cumulative rewards given fixed runtime. In sum, our results suggest that MA3PO notably improves team-level performances both in terms of episode usage and runtime. For

3-Agent Hopper, MA3PO fluctuates more due to the inherent noise in the environment, yet still outperforms existing benchmarks with considerable success. In sum, our results show that learning intrinsic motivation will also aid in obtaining improved team-level policies.

5.1 Ablation Studies

To better analyze the effectiveness of our MARL structure design, we conduct ablation studies over the different components in our MA3PO algorithm, including the use of parameter sharing and the introduction of parameterized intrinsic motivation in the learning process. As in previous sections, we fix the sampled steps, but plot the graph between sampled steps and the cumulative reward so as to better visualize how different components of MA3PO contribute to episodes utilization.

5.1.1 Parameter Sharing

In this section we analyze parameter sharing, a mechanism in which centralized critic ϕ shares identical network parameters with each actor policy network θ^a. We setup control experiments using parameter sharing as the control variable, and compare performances accordingly. Some example results are listed in Figure 5.3. These results clearly indicate a substantial increase of cumulative reward value when parameter sharing is introduced. This observation is also consistent with prior work on single-agent [13] and multi-agent PPO [25], in which parameter-sharing yields better performances than non-parameter-sharing.

5.1.2 Proxy Reward

We likewise set up control experiments on parametrized intrinsic motivation to see how individual agents influence team performance. Notice that MA3PO without intrinsic motivation is similar to MAPPO [25], in which a centralized critic learns a value function used by decentralized actors without proxy intrinsic reward. Sample results suggest that the extent to which parametrized intrinsic motivation help team-level performances de-

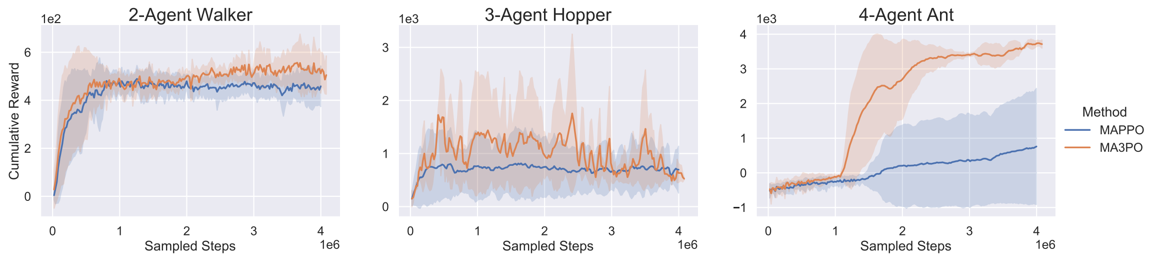

Figure 5.4: Envs where MA3PO outperforms MAPPO

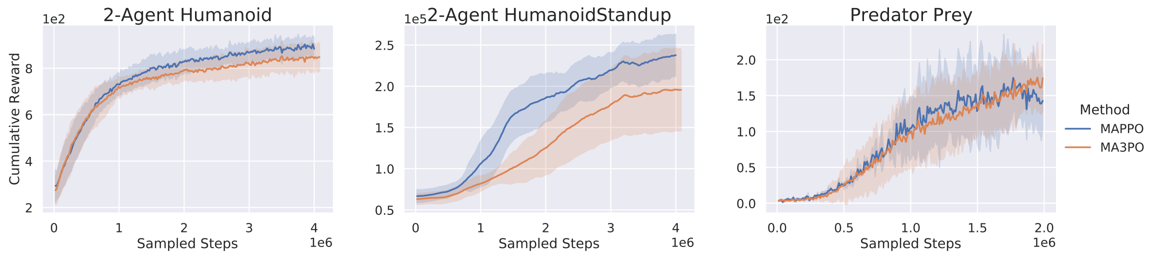

Figure 5.5: Envs where MAPPO outperforms MA3PO

pends on the complexity of the multi-agent environments. We list the result comparisons between MA3PO and the established MAPPO [25] in Figures 5.4 and 5.5, with Figure 5.4 showcasing the environments where MA3PO outperforms MAPPO and Figure 5.5 vice versa. These two plots suggest that MA3PO tends to work better in environments with more agents, possibly because fewer agents imply that fewer parameters suffice to model intra-agent interactions, and that intrinsic rewards are unnecessary to model interactions.

Furthermore, to visualize the relationship between proxy reward and overall performances, we plot the average intrinsic reward across agents against Sampled Steps for the environments where MA3PO outcompetes MAPPO in Fiture 5.6. We notice that MA3PO learns intrinsic reward more actively where the performance improvement becomes more noticeable, as demonstrated by the relative fluctuations in the learning curves. Since intrinsic rewards are independent from other components of the original actor network, our results indicate that parameterized intrinsic rewards correlates with overall team performance improvement.

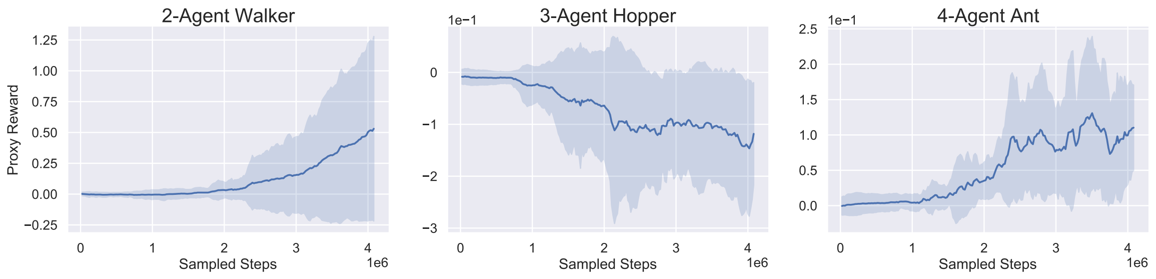

Figure 5.6: Example Visualization of Proxy Reward Learning

CHAPTER 6
CONCLUSIONS AND FUTURE WORK

In this paper we present MA3PO, a novel MARL algorithm that tackles continuous control. Consisting of a centralized critic and decentralized agent-level actors, our algorithm makes use of proxy rewards to integrate individual-agent intrinsic motivation to overall team learning. Experiments on diverse continuous multi-agent benchmarks against existing MARL algorithms demonstrate the power of MA3PO.

There are several directions for possible future research. To begin with, meta-learning principles can be extended to other policy-gradient algorithms such as the SAC [20] and PPG [19]. These algorithms perform favorably, and preliminary investigations on their multi-agent extensions [35] are promising. Alternatively, other possible intrinsic reward designs [27] may also be helpful. New findings over relationship between agents and the team capture alternative aspects of improvement in strengthening corporation, and proper methods to integrate these definitions into multi-agent framework will be interesting additions to Cooperative Reinforcement Learning.

REFERENCES

[1] Michał Kempka et al. “ViZDoom: A Doom-based AI research platform for visual reinforcement learning”. In: *2016 IEEE Conference on Computational Intelligence and Games (CIG)*. 2016.

[2] B Ravi Kiran et al. “Deep Reinforcement Learning for Autonomous Driving: A Survey”. In: *IEEE Transactions on Intelligent Transportation Systems* (2021).

[3] Maximilian Hüttenrauch, Adrian Šošić, and Gerhard Neumann. “Deep Reinforcement Learning for Swarm Systems”. In: *Journal of Machine Learning Research* (2019).

[4] Dayong Ye, M. Zhang, and Yun Yang. “A Multi-Agent Framework for Packet Routing in Wireless Sensor Networks”. In: *Sensors (Basel, Switzerland)* (2015).

[5] Frans A. Oliehoek, Matthijs T. J. Spaan, and Nikos Vlassis. “Optimal and Approximate Q-Value Functions for Decentralized POMDPs”. In: 2008.

[6] Peter Sunehag et al. “Value-Decomposition Networks For Cooperative Multi-Agent Learning”. In: *AAMAS*. 2018.

[7] Tabish Rashid et al. “QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning”. In: *Proceedings of the 35th International Conference on Machine Learning*. 2018.

[8] Kyunghwan Son et al. “QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning”. In: *Proceedings of the 36th International Conference on Machine Learning*. 2019.

[9] Tabish Rashid et al. “Weighted QMIX: Expanding Monotonic Value Function Factorisation”. In: *Advances in Neural Information Processing Systems*. 2020.

[10] Ryan Lowe et al. “Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS’17. Long Beach, California, USA, 2017, 6382–6393.

[11] Jakob N. Foerster et al. “Counterfactual Multi-Agent Policy Gradients”. In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[12] Jianhong Wang et al. “Shapley Q-Value: A Local Reward Approach to Solve Global Reward Games”. In: *AAAI*. 2020.

[13] John Schulman et al. “Proximal Policy Optimization Algorithms”. In: (). arXiv: `1707.06347`.

[14] David Silver et al. “Deterministic Policy Gradient Algorithms”. In: *Proceedings of the 31st International Conference on Machine Learning*. 2014.

[15] Timothy P. Lillicrap et al. “Continuous control with deep reinforcement learning”. In: *International Conference on Learning Representations*. 2016.

[16] Scott Fujimoto, Herke van Hoof, and David Meger. “Addressing Function Approximation Error in Actor-Critic Methods”. In: *Proceedings of the 35th International Conference on Machine Learning*. 2018.

[17] Volodymyr Mnih et al. “Asynchronous Methods for Deep Reinforcement Learning”. In: *Proceedings of The 33rd International Conference on Machine Learning*. 2016.

[18] John Schulman et al. “Trust Region Policy Optimization”. In: *Proceedings of the 32nd International Conference on Machine Learning*. 2015.

[19] Karl Cobbe et al. “Phasic Policy Gradient”. In: abs/2009.04416 (2020).

[20] Tuomas Haarnoja et al. “Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor”. In: *Proceedings of the 35th International Conference on Machine Learning*. 2018.

[21] Meng Zhou et al. “Learning Implicit Credit Assignment for Cooperative Multi-Agent Reinforcement Learning”. In: *Advances in Neural Information Processing Systems*. 2020.

[22] Yali Du et al. “LIIR: Learning Individual Intrinsic Reward in Multi-Agent Reinforcement Learning”. In: *NeurIPS*. 2019.

[23] Shariq Iqbal and Fei Sha. “Actor-Attention-Critic for Multi-Agent Reinforcement Learning”. In: *Proceedings of the 36th International Conference on Machine Learning*. 2019.

[24] Daniël Willemsen, Mario Coppola, and Guido C. H. E. de Croon. “MAMBPO: Sample-efficient multi-robot reinforcement learning using learned world models”. In: abs/2103.03662 (2021).

[25] Chao Yu et al. “The Surprising Effectiveness of MAPPO in Cooperative, Multi-Agent Games”. In: *CoRR* abs/2103.01955 (2021). arXiv: `2103.01955`.

[26] Zeyu Zheng, Junhyuk Oh, and Satinder Singh. “On Learning Intrinsic Rewards for Policy Gradient Methods”. In: *Advances in Neural Information Processing Systems*. 2018.

[27] Tejas D. Kulkarni et al. “Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation”. In: *NIPS*. 2016.

[28] Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. 1st. Springer Publishing Company, Incorporated, 2016.

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

[30] Ivo Grondman et al. “A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* (2012).

[31] Zhongwen Xu, Hado van Hasselt, and David Silver. “Meta-Gradient Reinforcement Learning”. In: *NeurIPS*. 2018.

[32] Zhongwen Xu et al. “Meta-Gradient Reinforcement Learning with an Objective Discovered Online”. In: *Advances in Neural Information Processing Systems*. 2020.

[33] Chloe Ching-Yun Hsu, Celestine Mendler-Dünner, and Moritz Hardt. “Revisiting Design Choices in Proximal Policy Optimization”. In: abs/2009.10897 (2020).

[34] John Schulman et al. “High-Dimensional Continuous Control Using Generalized Advantage Estimation”. In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2016.

[35] Christian Schröder de Witt et al. “Deep Multi-Agent Reinforcement Learning for Decentralized Continuous Cooperative Control”. In: abs/2003.06709 (2020).

[36] Joel Z. Leibo et al. “Multi-Agent Reinforcement Learning in Sequential Social Dilemmas”. In: *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. AAMAS ’17. 2017.

[37] Emanuel Todorov, Tom Erez, and Yuval Tassa. “MuJoCo: A physics engine for model-based control”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012.

[38] Shie Mannor, Reuven Rubinstein, and Yohai Gat. “The Cross Entropy Method for Fast Policy Search”. In: ICML’03. 2003.

[39] Eric Liang et al. “RLlib: Abstractions for distributed reinforcement learning”. In: *International Conference on Machine Learning*. 2018.