Multiagent Model-based Credit Assignment for Continuous Control

Dongge Han[1], Chris Xiaoxuan Lu[2], Tomasz Michalak[3], Michael Wooldridge[1]
[1] University of Oxford, Oxford, United Kingdom
[2] University at Edinburgh, Edinburgh, United Kingdom
[3] University at Warsaw & IDEAS NCBR, Warsaw, Poland

ABSTRACT

Deep reinforcement learning (RL) has recently shown great promise in robotic continuous control tasks. Nevertheless, prior research in this vein center around the centralized learning setting that largely relies on the communication availability among all the components of a robot. However, agents in the real world often operate in a decentralised fashion without communication due to latency requirements, limited power budgets and safety concerns. By formulating robotic components as a system of decentralised agents, this work presents a decentralised multiagent reinforcement learning framework for continuous control. To this end, we first develop a cooperative multiagent PPO framework that allows for centralized optimisation during training and decentralised operation during execution. However, the system only receives a global reward signal which is not attributed towards each agent. To address this challenge, we further propose a generic game-theoretic credit assignment framework which computes agent-specific reward signals. Last but not least, we also incorporate a model-based RL module into our credit assignment framework, which leads to significant improvement in sample efficiency. Finally, we empirically demonstrate the effectiveness of our framework on Mujoco locomotion control tasks.

KEYWORDS

Multiagent Systems; Reinforcement Learning; Cooperative Game Theory; Locomotion

ACM Reference Format:
Dongge Han[1], Chris Xiaoxuan Lu[2], Tomasz Michalak[3], Michael Wooldridge[1] . 2022. Multiagent Model-based Credit Assignment for Continuous Control. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022,* IFAAMAS, 9 pages.

1 INTRODUCTION

Reinforcement learning (RL) has recently shown many remarkable successes, e.g., in playing Atari and Go at a superhuman level [21, 31]. Recently, there have been wide research interest and significant advances in robotics learning using RL techniques [8, 16]. The topics of RL and classical control theory are closely related: both aim to find an optimal policy that optimizes an objective function, given a system represented by states and transition dynamics. Therefore, RL algorithms have the potential of enabling robots to learn in complex real-world tasks such as locomotion, manipulation and navigation. Unlike classical RL tasks, which have discrete action spaces and underlying state spaces (e.g. Atari and Go), problems in robotics often have high-dimensional continuous states and actions, and are often limited by real-world sample budgets [15]. To this end, prior research in robotic learning have developed RL algorithms capable of performing continuous control [10, 11, 18, 29], and sample-efficient learning methods, e.g., [1, 7, 13].

Despite the success demonstrated by RL methods in continuous control, when applying RL algorithms to robotic applications, it is essential not to overlook real-life physical limits such as sensing noise and communication delays [15]. Specifically, prior RL approaches typically model a robot as a single, fully-centralised agent which observes the whole state consisting of observations from each body/joint and learns an optimal policy that outputs a combined action for all controllers. Though a centralized controller can compensate for the state of the whole system, when deployed in complex, unseen real-life settings, local observations may be noisy, and communications of the local observations between body components can be delayed due to physical limitations. Moreover, robots with a centralised controller do not scale up to large number of controllers and can easily become incapacitated if any sub-component is compromised. In such scenarios, a decentralised control system is desirable, where the robot controllers learn and execute their own policies. Fortunately, many real-life robots consist of multiple connected links which can be modelled as a multiagent system [42]. To enable coordination among agents with local observations and decentralised controllers, a standard approach is to perform centralized optimisation during training and decentralised operation during execution (CTDE) [9, 20], i.e., the system can benefit from the state of the whole system during training, while acting in a decentralised manner. Using this framework, multiagent systems with RL agents have demonstrated coordination skills in complex coordination tasks such as Starcraft [26, 27, 44].

Inspired by the above framework, we apply multiagent RL to robotics which exhibits high-dimensional, continuous state and action spaces. Specifically, we extend Proximal Policy Optimisation (PPO) [29]—an algorithm widely used in both discrete and continuous control tasks—to a fully cooperative multiagent framework with CTDE. However, as the robot interacts with the environment as a whole, *only a global reward signal for the whole system is available while a reward function which signifies the contribution of each individual agent is absent.*

To address this problem, one can adopt solution concepts from cooperative game theory [4] that perform credit assignments for each agent, using its (marginal) contributions towards all possible coalitions of other agents. Several prior multiagent RL methods have used one such well-known solution concept, the Shapley value [4]), for credit assignment in applications with discrete action spaces

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online. © 2022 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

(such as traffic junction [39], Starcraft [17] and cooperative sensing [43]). However, *no model of this kind has been proposed for continuous action spaces so far.* Our first contribution in this work is the formulation of a generic game-theoretic credit assignment framework for multiagent continuous control using semivalues [6, 12]—a wide family of solution concepts that encompass many common credit assignment methods such as the Shapley value, the Banzhaf value, etc. This formulation provides a useful tool for studying the underlying relationship and differences between the commonly used credit assignment methods in multiagent RL.

The formulation of game-theoretic credit assignment allows us to fairly evaluate the contribution of each agent. However, to compute the credit assignments, *how can we obtain the value of different coalitions of agents, when only the value of the grand coalition is available (i.e., the global reward of the whole system)?* As we will discuss in Section 3.4, if we simply apply the method to estimate values of coalitions that was proposed for models with discrete action spaces [17] to continues ones, we would suffer from highly inaccurate results. This stems from a common problem in the multiagent robotics control domain, caused by the high dimensional continuous state and action spaces. To address this, we further propose a model-based framework for accurately estimating the values of coalitions of agents. This model-based framework greatly improves credit assignment in continuous control tasks and significantly boosts the rewards and sample-efficiency. Finally, we empirically evaluate our proposed methods on MuJoCo robotic continuous control tasks. We demonstrate that our model-based game-theoretic credit assignment leads to significant improvements in rewards and sample-efficiency compared to a shared advantage function and model-free credit assignment methods.

2 BACKGROUND

In this section, we introduce our notation and main definitions (see Table 1 for a summary of notation).

2.1 Reinforcement Learning:

A (single-agent) RL task is commonly defined by a reward function [34], and the goal of an agent is to learn an optimal policy which maximizes the cumulative future rewards. This can be modelled by the Markov decision process (MDP): $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$. At each time step t, an agent in state $s_t \in \mathcal{S}$, chooses an action $a_t \in \mathcal{A}$, receives a reward $r_t = r(s_t, a_t)$, and transits to the next state $s_{t+1} \in \mathcal{S}$ according to an unknown transition dynamics $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$. The agent aims to maximise the future return $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$ where $\gamma \in [0, 1]$ is a discount factor. We denote an agent's experience tuples as $\mathcal{D}_t = \langle s_t, a_t, s_{t+1}, r_t \rangle$. The policy of an agent is a mapping from states to probabilities of selecting a possible action, denoted as $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$.

2.1.1 Value Functions. The *(state) value function*, denoted $V^{\pi}(s)$, is the expected return of an agent which starts in s and follows policy π thereafter:

$$V^{\pi}(s) := \mathbb{E}_{\pi}[G_t | s_t = s] = \mathbb{E}_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s], \forall s \in \mathcal{S}. \quad (1)$$

Table 1: Notation

i, t, k	Indices for agents, timesteps, iterations
π^i	Policy of agent i (Sec. 2.1, 3.2)
G_t	Discounted return from t onwards (Sec. 2.1)
$V^{\pi}(s), Q^{\pi}(s, \mathbf{a})$	State and action-value functions (Sec. 2.1.1)
$A^{\pi}(s, \mathbf{a})$	Advantage function (Sec. 2.1.2, 3.2)
$\omega, \theta, \phi_s, \phi_r$	parameters of V, Q, f_s, f_r (Sec. 3.2, 3.4)
$v^C(s, \mathbf{a})$	(Characteristic) value of coalition (Sec. 3.3.1)
$\tilde{\mathbf{a}}_t$	Actions $\mathbf{a}_t$ masked by coalitions (Sec. 3.3.1)
$\mathcal{MC}^i(C, s, \mathbf{a})$	Marginal contribution of agent i to coalition C
$\psi^i(v), p_c$	semivalues and its probability indices (Sec. 3.3.2)
$f_s(s, \mathbf{a}), f_r(s, \mathbf{a})$	dynamics and reward models (Sec. 3.4.1)
$\hat{s}_{t+1}, \hat{r}_t$	States/Rewards predicted by model (Sec. 3.4.1)

The *(action) value function* $Q^{\pi}(s, a)$ defines the expected return when starting from s, taking action a, and following π thereafter:

$$Q^{\pi}(s, a) := \mathbb{E}_{\pi}[G_t | s_t = s, a_t = a]$$
$$= \mathbb{E}_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a], \forall s \in \mathcal{S}. \quad (2)$$

2.1.2 Proximal Policy Optimisation (PPO). PPO [29] is an actor-critic policy gradient RL algorithm which is widely used in both discrete and continuous control tasks, due to its simplicity in implementation, sample complexity, and ease of hyperparameter tuning. Specifically, an *actor* corresponds to the policy $\pi(a|s)$, while a *critic* corresponds to the value function $V^{\pi}(s)$. The actor update is guided by an *advantage function* $A(s, a)$ estimated using the critic. The advantage function $A^{\pi}(s, a) = Q^{\pi}(s, a) - V^{\pi}(s)$ measures how much better (worse) taking action a in state s is compared with the policy's default behavior. Intuitively, if the advantage $A(s, a)$ is positive, the actor parameters move towards the direction where this action becomes more likely. To improve stability, PPO avoids parameter updates that change the policy too much by clipping the ratio between old and new policies. In practise, $A^{\pi}(s, a)$ can be estimated using methods such as general advantage estimation (GAE) [28].

2.1.3 Centralised Training, Decentralised Execution. A standard practise in training a decentralised multiagent RL system is Centralised Training, Decentralised Execution (CTDE) [9]. For example, in a multiagent system with agents $N = \{1, \ldots, n\}$, a *centralised critic* (typically, action value functions $Q(\mathbf{s}_t, a_1, \ldots, a_n)$) and decentralised actors $(\pi_1, \ldots, \pi_n)$ are learned. The centralised critic takes as input the observations and joint actions of all agents, while the decentralised actors take the agents' local observations as input and outputs an action per agent. Compared with having decentralised critics (one per agent) which only conditions on the local observations and actions, the advantage of CTDE is that the centralised critic prevents an agent from experiencing un-stationary environment as a result of treating other learning agents as part of the environment. Moreover, the critic is only used during training; hence, the agents learn coordinated behaviours which can be executed in a decentralised manner during execution.

2.2 Cooperative Game Theory

A *cooperative game* is given by a pair $G = (N, v)$, where $N = \{1, \ldots, n\}$ is the set of agents (players), and $v\colon 2^N \to \mathbb{R}$ is the *characteristic function*, which assigns a real value $v(C)$ to every coalition C reflecting its performance. We assume $v(\emptyset) = 0$. Subsets of agents $C \subseteq N$ are called *coalitions*. To measure the contribution of an individual agent to a cooperative task, we use solution concepts [4] from cooperative game theory that evaluate the contribution of each agent $i \in N$ to the game. We will denote it $\psi^i \in \mathbb{R}$. Specifically, we focus on a wide class of solution concepts called semivalues [6]. For clarity, we will introduce the general definition of semivalues in Section 3.3. Before this, let us introduce the concept of *marginal contribution* and two well-known semivalues: the Shapley and Banzhaf values. Intuitively, the marginal contribution of i to coalition C is the difference that this agent makes towards C before and after joining it, i.e., $\mathcal{MC}^i(C) \coloneqq v(C \cup \{i\}) - v(C)$.

The Shapley value [30] is the most common solution concept. It is defined as follows:

$$\psi^i_{\text{Shapley}}(N, v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|C|!(|N| - |C| - 1)!}{|N|!} \mathcal{MC}^i(C).$$

The Banzhaf value [2] is another common solution concept:

$$\psi^i_{\text{Banzhaf}}(N, v) = \frac{1}{2^{|N|-1}} \sum_{C \subseteq N \setminus \{i\}} \mathcal{MC}^i(C).$$

Intuitively, the Banzhaf value of i is simply the average marginal contribution of this agent across all possible coalitions. The Shapley value is the weighted average, where the weight of each marginal contribution depends on the size of the coalition and the total number of agents.

2.3 Kinematics Tree of Robots

A robot capable of performing continuous control can be defined via a kinematics tree [32, 40] of nested bodies, specifying the mechanical structure and physical properties of the robot. Within such a kinematics tree, there are three major types of entities: `Bodies`, `Joints and Actuators`. The `Bodies` represent the rigid bodies of the robot, e.g., foot. Physical properties such as the relative position and geometry are specified for each body. The `Joints` are moveable components which connect parent and child bodies, and creates motion degrees of freedom between them. Typical joints include ball joints (3 rotational degree of freedom), hinge joints (1 rotational degree of freedom) and slide joints (1 translational degree of freedom). For example, in a cheetah robot (Figure 1), the torso (root body) can perform translational and rotational movements through slide and hinge `joints`. The back foot (body) is attached to its parent back shin (body) through a hinge `joint`. Attached to the joints are `Actuators` which are the components that actuate the joints. Many real life robots consist of multiple connected links, we now introduce how to model the robot as a multiagent system.

3 MULTIAGENT CONTINUOUS CONTROL

3.1 Problem Formulation

3.1.1 Robot Joints as a Multiagent System. Figure 1 shows an example cheetah robot built using MuJoCo [3] (more details in Section 4.1.1). We model the robot as a multiagent system of N agents, where each agent represents: 1. a rigid body component (e.g., the back foot) of the robot, 2. the joint which attaches the body to its parent (e.g., the ankle joint which attaches the back foot to the back shin), and 3. any actuators attached to the joint. In this way, the agent receives input of the local sensor observations to its joint (e.g., positions, angles, translational and angular velocities, external forces and torques), and outputs a control action for the actuator.

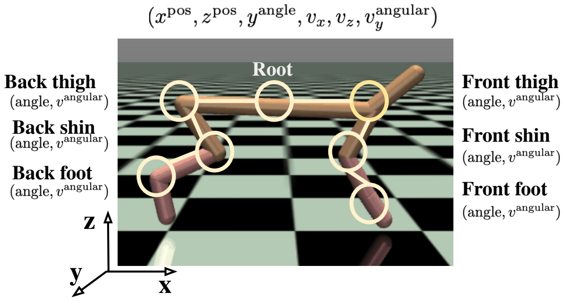

$(x^{\text{pos}}, z^{\text{pos}}, y^{\text{angle}}, v_x, v_z, v_y^{\text{angular}})$
Root
Back thigh
(angle, v^{angular})
Back shin
(angle, v^{angular})
Back foot
(angle, v^{angular})
Front thigh
(angle, v^{angular})
Front shin
(angle, v^{angular})
Front foot
(angle, v^{angular})
z
y
x

Figure 1: Multiagent MuJoCo Cheetah

3.1.2 Multi-agent based Robot Locomotion. Typically in the robotic control problems, the actuators have a continuous action space which are normalised to $[-1, 1]$ for the convenience of learning. Continuous control on its own has a broad range of applications, in that almost all real-life robotic control problems fall into the domain of continuous control, for example, formation control for a system of unmanned aerial vehicles (UAV), grasping and manipulation, locomotion of legged robots, and multi-robot exploration of unknown environments. In this work, we focus on the robot locomotion task, which is a representative continuous robot control problem in this family. The objectives of robot locomotion tasks are to train the robot to learn control policies in order to transport from place to place through walking, hopping, swimming, etc. Here we model the locomotion tasks as a fully cooperative Multiagent decision making problem, described by a Markov game [19] denoted by a tuple $\langle N, \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$, where N is the set of all agents $N = \{1, \ldots, n\}$, and $\mathcal{S}$ are global states of the environment. Each agent i has a continuous action space $\mathcal{A}_i$ and the combined action space of the robot is $\mathcal{A} = \mathcal{A}_1 \times \ldots \times \mathcal{A}_n$. At each step t, each agent i makes a local observation $s_t^i = o^i(s_t)$, and chooses an action simultaneously according to its own policy $a_t^i \sim \pi^i(s_t^i)$. Then, a combined action composed of the actions of all agents $\mathbf{a}_t = (a_t^1, \ldots, a_t^n)$ is performed on the environment (the combined policy of all agents is denoted as $\pi = (\pi^1, \ldots, \pi^n)$). Upon taking the action, all agents receive a shared reward $r_t = r(s_t, \mathbf{a}_t)$ which evaluates how well the robot performed (e.g., the distance travelled along a specified direction, the control energy consumption, the stability of the robot, etc), and the environment transitions to the next state according to the transition dynamics $P : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$. At each step, the agents aim to maximise the cumulative future discounted return $G_t = \sum_k \gamma^k r_{t+k+1}$ where $\gamma \in [0, 1]$ is a discount factor.

3.2 Multiagent PPO with Shared Advantage

Having formulated the multiagent decision problem, we next introduce an algorithm for optimising the agents' policies. To optimise the continuous control policies, we extend the standard PPO algorithm with actor-critic framework (c.f. Section 2.1.2) to cooperative multiagent PPO. In particular, we adopt the centralised training, decentralised execution (CTDE) paradigm [9], c.f. Section 2.1.3. As the agents are fully cooperative and the whole robot receives a global reward signal, we let the agents share a centralised critic which estimates the value function of the global state $V_\omega : \mathcal{S} \to \mathbb{R}$. The centralised critic is parametrised by ω, and in our case, ω is a neural network. The critic is only used during training as a guide for optimising the actors and will be dismissed in the execution phase—during the execution only the actors are used to choose the actions. The actors are decentralised, i.e., for the i−th agent, its actor $\pi_{\theta^i} : \mathcal{S}^i \to \mathcal{A}^i$ parametrised by neural network θ^i, chooses its own action given the local observations. Most locomotion tasks have continuous action space by nature; hence we model each agent's policy outputs by a Gaussian distribution. At each step, action is sampled from the Gaussian distribution. During training, each actor learns to update the mean of the Gaussian distribution, and the standard deviation can be either static or learned.

We optimise the centralised critic and decentralised actors following the PPO algorithm: At each epoch k, the agents collect a set of trajectories $\mathcal{D}_k$ by running their current policy $\pi_k = (\pi^1(\theta^1_k), \dots \pi^n(\theta^n_k))$ in the environment. Then, the critic which estimates the (state) value function $V(s)$ is updated by regression. The loss is defined as the mean squared error between the predicted value of the state s_t the empirical return $\hat{G}_t$ from t onwards:

$$\omega \leftarrow \arg\min_{\omega} \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \big(V_\omega(s_t) - \hat{G}_t\big)^2.$$

The actor of each agent is updated by optimising the PPO clipped surrogate objective:

$$\theta^i_{k+1} = \arg\max_{\theta^i} \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\Big(\frac{\pi_{\theta^i}(a^i_t | s^i_t)}{\pi_{\theta^i_k}(a^i_t | s^i_t)} A(s_t, \mathbf{a}_t), \\ \text{clip}\big(\frac{\pi_{\theta^i}(a^i_t | s^i_t)}{\pi_{\theta^i_k}(a^i_t | s^i_t)}, 1 - \epsilon, 1 + \epsilon \big) A(s_t, \mathbf{a}_t) \Big),$$

where $A(s_t, \mathbf{a}_t)$ is the shared advantage function computed using the critic through generalised advantage estimation (c.f. Sec. 2.1.2). Intuitively, if the advantage $A(s, a)$ is positive, the actor parameters update in the direction where this action becomes more likely. PPO avoids parameter updates that change the policy too much by clipping the ratio between old and new policies [29].

3.3 Multiagent PPO with Game-theoretic Agent-specific Advantage

The above Multiagent PPO with shared advantage function often yields decent performance for cooperative multiagent tasks. However, the shared advantage function $A(s_t, \mathbf{a}_t)$ only evaluates the quality of the combined action, which fails to assess each agent's individual contribution. As we will discuss in Sec. 4.2.1, failing to address per-agent contribution can result in low sample efficiency oftentimes, for example, an under-actuated agent can equally share and update its policy using the global advantage. To deal with this issues, we leverage the fair credit assignment methods from cooperative game theory, and present a generic game theoretic framework for agent-specific advantage computation in Multiagent PPO.

3.3.1 The Characteristic Function. To compute the value of an agent assigned by the game-theoretic solution concepts, we first need to define the characteristic function. At each timestep t, given the environment state s_t and agents' joint action $\mathbf{a}_t = (a^1_t, \dots, a^n_t)$, let the value of a coalition C be defined as:

$$v^C(s_t, \mathbf{a}_t) = Q^\pi(s_t, \tilde{a}^1_t, \dots, \tilde{a}^n_t), \text{ where} \tag{3}$$
$$\tilde{a}^i_t = \begin{cases} a^i_t & \text{if } i \in C \\ a_{\text{default}} & \text{if } i \in N \setminus C, \end{cases}$$

where $\tilde{a}^i_t$ denotes the action of agent i is replaced by a default one if i is outside the coalition, a widely adopted practice [17, 41]. To follow the game theoretic conventions where empty coalitions have zero value, we can normalise the characteristic value function by subtracting a baseline value $Q^\pi(s_t, \mathbf{a}_{\text{default}})$. However, the marginal contributions will stay invariant so we will use the above definition for simplicity. Given this characteristic function, agent i's marginal contributions towards coalition C:

$$\mathcal{MC}^i(C, s_t, \mathbf{a}_t) = v^{C \cup \{i\}}(s_t, \mathbf{a}_t) - v^C(s_t, \mathbf{a}_t), \tag{4}$$

represents the difference made by i's chosen action towards coalition C, compared with the default action.

3.3.2 Agent-Specific Advantage. We now introduce a generic game-theoretic credit assignment framework for multiagent continuous control using semivalues [6, 12] – a wide family of solution concepts that encompass many common credit assignment methods such as the Shapley value, Banzhaf value, etc.

$$\psi^i(v) = \sum_{C \subseteq N \setminus \{i\}} w_c \mathcal{MC}^i(C) \text{ where } c = |C|, \sum_{c=0}^{|N|-1} w_c \binom{|N|-1}{c} = 1.$$

To better understand them, we can rewrite the semivalues as:

$$\psi^i(v) = \sum_{c=0}^{|N|-1} p_c \frac{\sum_{C \subseteq N \setminus \{i\}, |C| = c} \mathcal{MC}^i(C)}{\binom{|N|-1}{c}}, \text{where } \sum_{c=0}^{|N|-1} p_c = 1. \tag{5}$$

Note that the denominator $\binom{|N|-1}{c}$ is the number of size-c coalitions excluding player i. Intuitively, the semivalue is a weighted sum of an agent's average marginal contribution towards different sized coalitions, where p_c form a probability distribution. In particular, the Shapley value places uniform weight on all coalition sizes: $p_c = \frac{1}{|N|}$; and the Banzhaf value has a bell-shaped distribution, i.e., $p_c = \frac{1}{2^{|N|-1}} \binom{|N|-1}{c}$. Any other probability distribution p_c also specifies a semivalue.

In our model, *at each timestep, the semivalue ψ^i of an agent computes the contributions of the agent's chosen action towards the global future return of the robot, and hence will be used as the agent-specific pseudo advantage for multiagent PPO updates.* We call the semivalues "pseudo" advantage because they are not defined strictly as $A(s, a) = Q(s, a) - V(s)$. Nevertheless, the semivalues provide an effective proxy of advantage as they can evaluate the individual contribution of an agent's action towards the global value.

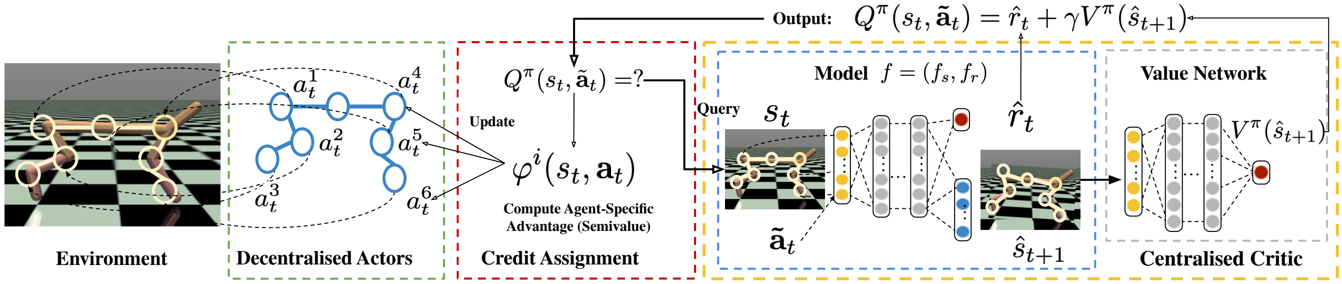


Figure 2: Model-based Multiagent Credit Assignment. Our framework consists of three modules: (yellow) the centralised critic module which consists of a dynamics/reward model $f = (f_s, f_r)$ and a centralised state-value function $V^\pi(s)$; (red) credit assignment module which queries the critic for coalition values and computes the counterfactual solution concepts assigned to each agent; (green) actors module with N agents, which are updated using the credits from the credit assignment module.

Algorithm 1 Model-based Multiagent Credit Assignment

```
Input: $N = \{1, \dots, n\}$: the set of all agents.
Initialise: model $f = (f_s, f_r)$, centralised critic $V_\omega$, decentralised actors $\pi_\theta^i$ for each agent $i \in N$
for iterations $k = 0, 1, 2, \dots$ do
    Collect sets of trajectories $\mathcal{D}_k = \{\tau\}$ by running policy $\pi_k = (\pi^1_{\theta^1_k}, \dots \pi^n_{\theta^n_k})$ in the environment. Compute returns $\hat{G}_t$
    # Fit the dynamics/reward model (in minibatches):
        $f_s \leftarrow \arg\min_{f_s} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \|s_{t+1} - (s_t + f_s(s_t, \mathbf{a}_t))\|^2$
        $f_r \leftarrow \arg\min_{f_r} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \|(r_t - f_r(s_t, \mathbf{a}_t)\|^2$
    # Compute per-agent advantage $\psi_i$ for all timesteps $t$:
        Sample coalitions $C_m$
        Compute coalition values by model $f$ and critic $V_{\omega_k}$:
            $v(C_m, s_t, \mathbf{a}_t)) = f_r(s_t, \tilde{\mathbf{a}}_t) + \gamma V_\omega^\pi(f_s(s_t, \tilde{\mathbf{a}}_t) + s_t)$
        Compute marginal contribution $\mathcal{MC}^i(C_m, s_t, \mathbf{a}_t)$
        Compute semivalues $\psi^i$ using the marginal contributions.
    # For each agent $i$, update the policy by maximising the PPO-Clip objective:
        $\theta^i_{k+1} = \arg\max_{\theta^i} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \min\Big(\frac{\pi_{\theta^i}(a_t^i|s_t^i)}{\pi_{\theta^i_k}(a_t^i|s_t^i)} \psi^i(s_t, \mathbf{a}_t),$
            $\text{clip}\big(\frac{\pi_{\theta^i}(a_t^i|s_t^i)}{\pi_{\theta^i_k}(a_t^i|s_t^i)}, 1 - \epsilon, 1 + \epsilon\big)\psi^i(s_t, \mathbf{a}_t)\Big),$
    # Fit the centralised critic by regression:
        $\omega_{k+1} = \arg\min_\omega \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \big(V_\omega(s_t) - \hat{G}_t\big)^2$
end for
```

Since RL training procedures typically require millions of timesteps, enumerating all possible coalitions and computing their characteristic value at each time step is inefficient. A simple procedure for estimating the semivalue is through Monte-Carlo sampling and output the average over the agent's marginal contribution towards all sampled coalitions. For each sample drawn, we can first sample a coalition size c_m according to the semivalue distribution p_c, then uniformly sample a coalition C_m of the size c_m.

3.4 Model-based Advantage Estimation

The semivalues (e.g., the Shapley and Banzhaf values) as agent-specific pseudo advantages, allow us to evaluate the contribution of each agent and its chosen action. *However, how can we obtain the coalitions' values, when only the value of the grand coalition is available (i.e., the global reward of the whole system)?*

An obvious approach is to perform extra simulations, where for each simulation the grand coalition is replaced by a different coalitions of agents. This, however, would require exponentially more simulations which is expensive for mutliagent RL domains in general. Another approach which does not require extra simulations is to infer the value of coalitions from the present simulations [17]. However, this approach suffers from inaccurate estimations in the multiagent robotics control domain, due to the high dimensional continuous state and action spaces.

To accurately estimate the value of different coalitions, we draw inspiration from model-based RL and propose incorporating an additional model of the environment dynamics and rewards. In this way, we obtain a better coalition value estimation through model-based simulations of different coalitions. Meanwhile, optimisation of the model is straightforward (via supervised learning) and does not require extra environment interactions.

3.4.1 Dynamics and Reward Model. We use a *model* parametrised by $\phi = (\phi_s, \phi_r)$, which consists of a transition model f_s which maps the state-action tuple to the the difference between the next and current state (alternatively, f_s can map the state-action tuple to the next state directly), and a reward model f_r which maps the state-action tuple to the reward [22]:

$$\forall s_t \in \mathcal{S}, a_t \in \mathcal{A}, \quad \hat{s}_{t+1} = f_s(s_t, a_t) + s_t, \tag{6}$$

$$\hat{r}_t = f_r(s_t, a_t). \tag{7}$$

The model can be optimised through supervised learning, by regression using mean-squared error between the predicted next states/rewards and actual ones.

$$\min_{\phi_s} \frac{1}{|\mathcal{D}|} \sum_{\langle s_t, s_{t+1}, \mathbf{a}_t, r_t \rangle \in \mathcal{D}} \|s_{t+1} - (s_t + f_s(s_t, \mathbf{a}_t))\|^2 \tag{8}$$

$$\min_{\phi_r} \frac{1}{|\mathcal{D}|} \sum_{\langle s_t, s_{t+1}, \mathbf{a}_t, r_t \rangle \in \mathcal{D}} \|(r_t - f_r(s_t, \mathbf{a}_t)\|^2. \tag{9}$$

The model is only used in (centralised) training, hence only one centralised model is needed and can compensate for the state of the whole system. During each iteration of training, we alternate between model fitting and agent updates using the data collected from environment interactions. Moreover, we only use the model for generating imaginary simulations of short-horizons, which mitigates the error-prone model predictions drifted in long horizons.

3.4.2 Estimating the coalition values using the model. Inspired by model-based value expansion [7], we use the model to perform estimation of the one-step transition dynamics and rewards. In this way, we obtain a better value estimation through imaginary simulations of different coalitions using the dynamics/reward model. Through the Bellman equation, we can estimate the action-value function through a state-value function V^{π} in the following:

$$\begin{aligned} Q^{\pi}(s_t, \tilde{\mathbf{a}}_t) &= \mathbb{E}[r_t + \gamma V^{\pi}(s_{t+1}) |\ s_t, a_t] \\ &= \mathbb{E}[\hat{r}_t + \gamma V^{\pi}(\hat{s}_{t+1}) |\ s_t, a_t] \\ &\approx f_r(s_t, \tilde{\mathbf{a}}_t) + \gamma V^{\pi}(s_t + f_s(s_t, \tilde{\mathbf{a}}_t)). \end{aligned} \tag{10}$$

A schematic of the framework is shown in Figure 2. At each training step, to produce the credit value for each agent i, the credit assignment module first samples coalitions $C_m \subseteq N \setminus \{i\}$ according to the semivalue, and computes the average over counterfactual value of i towards the sampled coalitions $\mathcal{MC}_i(C_m) = v(C_m \cup \{i\}) - v(C_m)$. To obtain the value of the coalitions, say $v^{C_m}(s_t, \tilde{\mathbf{a}}_t) = Q^{\pi}(s_t, \tilde{\mathbf{a}}_t)$, the credit assignment module queries the critic module (yellow in Figure 2): the model (blue) first produces the next state $\hat{s}_{t+1} = f_s(s_t, \tilde{\mathbf{a}}_t)$ and reward $\hat{r}_t = f_r(s_t, \tilde{\mathbf{a}}_t)$, then the centralised state-value critic (grey) produces $V^{\pi}(\hat{s}_{t+1})$. Together with the reward r_t produced by the model, the critic module outputs the value of the coalition C_m as $Q^{\pi}(s_t, \tilde{\mathbf{a}}_t) = \hat{r}_t + \gamma V^{\pi}(\hat{s}_{t+1})$. Finally, having obtained the credit value assigned to each agent, the actors module (green) can update the decentralised actors through multiagent PPO. The full algorithm is shown in Algorithm 1.

4 EXPERIMENTS

In this section, we present empirical results of our model-based multiagent RL framework on continuous tasks.

4.1 Experiment Setups

4.1.1 Experiment Settings. For the experiments we use OpenAI Gym MuJoCo [3, 36] locomotion tasks, which are the standard benchmarks for continuous control with RL [22, 40]. Two exemplar robot locomotion tasks are selected as our case studies: (1) the Cheetah movement in 2D space and (2) the Ant movement in 3D space. The cheetah model in MuJoCo has 7 joints, including one root joint and 6 joints each paired with one actuator. Following the definition in Section 3.1, we consider each joint as an agent with an actuator that applies a joint torque. For all agents, the action space is normalised to $[-1, 1]$. The root agent observes the position, angle, velocity and angular velocity and for other agents, their observation includes the relative angle with respect to the body that they are attached to, and the angular velocity. The global observation utilised by the critic and the dynamics model (c.f. Figure 2) is given by a concatenation of local observations. The objective of the agent is to move forward in the x-direction. On the other hand, the Ant model can move in a 3D space and the agents are similarly defined as in the Cheetah. Notably, the agents of an Ant model in MuJoCo can additionally observe external forces such as friction. The objective of the Ant locomotion task is also to move forward in the x-direction.

4.1.2 Models and Training Details. Our multiagent system consists of a model $f = (f_s, f_r)$ of the environment, the centralised critic, and the decentralised actors. We use PyTorch [24] for implementing and optimising the neural network models. For the dynamics model f_s, we use 4 fully-connected layers with ReLU activation function [23] and a linear output layer with the dimension of 128. For the reward model f_r, we use 3 fully-connected layers with ReLU activation function and a linear output layer with the dimension of 128. The model is optimised in minibatches of size 64 using Adam optimiser [14] with the learning rate of 10^{-3}. For the centralised critic, we use 3 fully-connected layers of the dimension of 32 with Tanh [23] activation and a linear output layer which produces the value of the input state. For the decentralised actors, each actor uses 3 fully-connected layers of the dimension of 32 with Tanh activation, and an output layer with Tanh activation. The default actions a_{default} are given by zero vectors. The control action of each agent corresponds to the torque exerted on the joint by the actuator and is normalized to the real interval [-1, +1]. Therefore, the agents playing default actions correspond to joints that are un-actuated, which is an intuitive model for the agents not contributing. Both the critic and the actor are optimised using the Adam optimiser. The learning rate for the critic is 10^{-3} and is 3×10^{-4} for the actor. All graphs plot mean and standard deviation across 5 seeds.

4.2 Overall Results

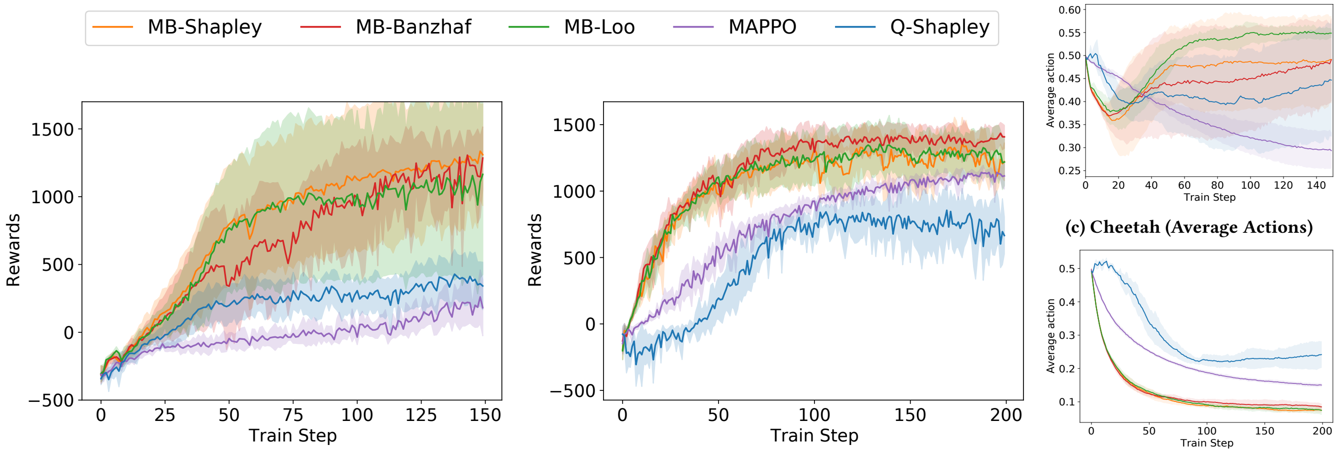
Figure 3 shows the overall performance of our algorithms and baseline algorithms on the MuJoCo locomotion tasks with cheetah and ant robots. The y-axis shows the episode rewards and x-axis shows the train steps. In particular, we follow Eq. 5 to implement the following variants of our semivalue based credit assignment algorithms. `MB` is for Model-based coalition value estimation:

- `MB-Shapley`: the Shapley value ($p_c = \frac{1}{|N|}$);
- `MB-Banzhaf`: the Banzhaf value ($p_c = \frac{1}{2^{|N|-1}}\binom{|N|-1}{c}$);
- `MB-Loo`: the Leave-one-out value ($p_c = \mathbb{1}_{c=|N|-1}$), where we compute the marginal contribution of an agent towards the grand coalition (made of all other agents);

For *baselines*, we compare with the following state-of-the-art algorithms in multiagent RL that use the Shapley value-based credit assignment and the global advantage respectively:

- `Q-Shapley` [17]: we adapt the discrete domain algorithm [17] which used Shapley value for credit assignment, to our multiagent PPO which can be used for continuous control. The coalition values are computed using a centralised critic of Q-values in a model-free manner.
- `MAPPO` (Multiagent PPO) [44]: In `MAPPO`, the centralised critic computes a *shared advantage function (GAE)* for all agents based on the global reward signal. And the agents are trained using centralised training, decentralised execution.

4.2.1 Shared advantage vs. agent-specific advantage. We are now in a position to present the results and findings. In both the cheetah (Figure 3a) and ant (Figure 3b) cases, we observe that our



(a) Cheetah (Rewards)

(b) Ant (Rewards)

(c) Cheetah (Average Actions)

(d) Ant (Average Actions)

Figure 3: Average Rewards Multiagent Cheetah and Ant

credit assignment methods that use model-based estimation (i.e., MB-Shapley, MB-Banzhaf, MB-Banzhaf) significantly outperform multiagent PPO (MAPPO) in terms of both the *average rewards* and *sample efficiency*. For example, in the Ant case, our model-based algorithm with Banzhaf credit assignment (MB-Banzhaf) quickly reaches the reward of 1000 within 50 train steps, while MAPPO needs around 150 train steps to reach the same level. MB-Banzhaf also converges to a higher average reward (∼ 1400) than MAPPO (∼ 1100). In all our results, performing credit assignment and evaluating the per-agent contribution (c.f. Sec. 3.3) is consistently better than agents using a shared advantage function. A demo video can be found on https://youtu.be/gFyVPm4svEY.

4.2.2 Model-based vs. Model-free credit assignment. We next compare our MB-Shapley with the baseline Q-Shapley [17] in order to understand the role of model-based credit assignment. In our MB-Shapley, the coalition values are estimated using the dynamics/reward model and the state-value critic V^{π}. While in Q-Shapley, the coalition values are estimated in a model-free way with action-value critic Q^{π}. As shown in Figure 3a and 3b, for both locomotion tasks, MB-Shapley outperforms Q-Shapley. This shows that when only the reward of the grand coalition is provided by the simulations, it is difficult to infer the values v^{C} of different coalitions using a model-free Q-value critic. In contrast, a better estimation of values of different coalitions is obtained with the predictions of our model-based module, which enables more effective agent-specific credit assignment and sample-efficient learning.

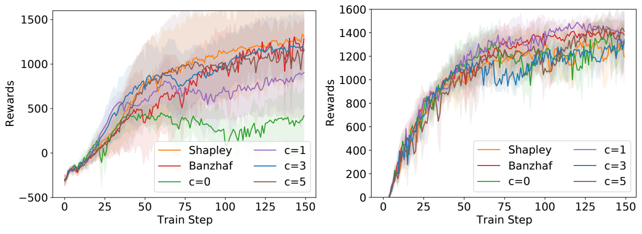
4.2.3 Robustness of Different Semivalue Variants. Our model-based coalition value estimation supports the broad class of semivalue credit assignments. Still, from Figure 3, we observe that the agents trained using different selected semivalues (MB-Shapley, MB-Banzhaf, MB-Loo) all deliver decent performance. In the case of cheetah (Figure 3a), both MB-Shapley and MB-Banzhaf yield the highest rewards, and in ant (Figure 3b), MB-Banzhaf yields the highest rewards. Other model-free methods such as model-free Banzhaf and Loo both yield similar performance to Q-Shapley. Overall, MB-Banzhaf has the lowest variance among the three variants.

4.2.4 Average Actions. Figures 3c and 3d show the mean absolute value of actions, averaged across the agents (joints actuators). This quantity refers to the controller output (normalised between [0, 1]), and higher action values mean higher energy consumption in a practical context. In cheetah, MB-Loo has the highest average action, with MB-Shapley and MB-Banzhaf having medium values. In the ant case, MB-Shapley, MB-Banzhaf, and MB-Loo all have low control output. We observe that the ant robot trained using MB-Shapley, MB-Banzhaf learn to move forward using a subset of joints. In contrast, other joints are mainly learned for steering, implying that using these two semivalues can also encourage diverse roles of different agents in a locomotion task.

4.3 Component-wise Analysis

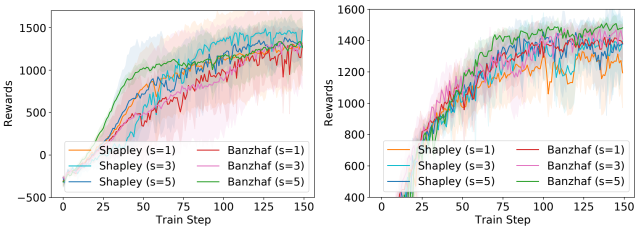
4.3.1 Variation in Coalition Sizes. In this section, we discuss the model-based semivalues from the perspective of coalition sizes. Figures 4a and 4b show the performance of agents learned using two semivalues MB-Shapley ($p_c = \frac{1}{N}$), MB-Banzhaf ($p_c = \frac{1}{2^N}\binom{N-1}{c}$), and some semivalues defined by $p_c = \mathbb{1}_{c=x}$, e.g., a semivalue with $p_c = \mathbb{1}_{c=3}$ is an agent's average marginal contributions towards size-3 coalitions. In the case of cheetah (Figure 4a), the semivalues that use small coalition sizes (e.g., $\mathbb{1}_{c=0}$) yield unsatisfactory performance, while those using mid-sized coalitions (e.g., $\mathbb{1}_{c=0}$) yield a high reward. In the case of the ant (Figure 4b), the semivalues focused on coalitions of heterogeneous sizes all yield high rewards. This suggests that the agents' actions in the cheetah case are more dependent; hence, they require agents to closely coordinate their actions. While for the ant, the agents have more distinct roles and their actions are more independent. The advantage of Shapley value and Banzhaf value is that they take into account all coalitions sizes and therefore are robust across different robots. Shapley value uniformly weighs all coalition sizes while Banzhaf value puts higher weights on mid-sized coalitions.

4.3.2 Variation in Sample Sizes. Lastly, we examine how the number of coalitions sampled per agent per step (when estimating the semivalues) affects the robot's performance. We compare MB-Shapley



(a) Cheetah **(b) Ant**

Figure 4: Results for Varying Coalition Sizes



(a) Cheetah **(b) Ant**

Figure 5: Results for Varying Sample Sizes

for the number of samples of $s = 1, 3$, and 5. We do the same for `MB-Banzhaf`. Figures 5a and 5b show that for both robots and both semivalue variants, reducing the number of sampled coalitions per node does not drastically decrease the performance in terms of reward and sample efficiency. Even using a small number of sampled coalitions, such as one per agent in each step, can still yield sufficiently good performance. We conjecture that this robustness property arises because, despite the small number of sampled coalitions per step, an RL algorithm typically runs on the scale of million timesteps. Hence, the number of sampled coalitions throughout the entire learning phase often will be sufficiently large. Importantly, because RL requires many credit assignment computations, this property helps us significantly reduce computation cost without compromising the learning performance.

5 RELATED WORK

Multiagent RL. The simplest form of multiagent RL is Independent Q-learning (IQL) [35], where a group of agents each learns on their own and treats other agents as part of the environment. While IQL delivers decent performance, the agents frequently face the challenge that the environment appears non-stationary, as other agents simultaneously learn and update their policies. To address this issue, centralised training, decentralised execution (CTDE) was first proposed separately by [9] and [20]. This paradigm allows the agents to access the global information during training and has been a standard approach to recent multiagent RL algorithms since then. [9] proposed an actor-critic algorithm that addresses multiagent credit assignment explicitly using the counterfactual value of each agent and is applied to domains with discrete action space such as Starcraft. [20] proposed a multiagent policy gradient algorithm where each agent receives a separate reward and uses their critic. This method can apply to competitive settings but does not address the multiagent credit assignment problem. Another line of work using CTDE and Q-learning are the implicit credit assignment methods such as [26] [25] [37]. However, performing local maximisation over the decomposed Q-values is non-trivial in the continuous tasks. More recently, [44] showed the effectiveness of PPO in multiagent problems for discrete action space domains. Their model is optimised using a centralised reward and demonstrated to outperform baseline methods such as [20][26].

Game-theoretic Credit Assignment for Multiagent RL. Several works have considered the Shapley value for credit assignment in multiagent RL with discrete action spaces [17][39][38][43]. Most notably, [17] showed state-of-the-art performance on Starcraft. In our experiments, we adapt their credit assignment framework from discrete domains to the continuous multiagent PPO. Unlike most prior credit assignment methods, which only consider the Shapley value, we introduce semivalues to credit assignment in multiagent RL and define a more generic game-theoretic framework that encompasses a family of common solution concepts and analyze their relations. Moreover, we use the new framework to address the problem of multiagent continuous control, and show that our model-based RL module can better estimate the coalition values for credit assignment and improve the sample efficiency accordingly.

Model-based RL. Model-based RL approaches typically alternate between fitting a predictive model of the environment dynamics/rewards and updating the control policies. The model can be used in various ways, such as execution-time planning [5, 22], generating imaginary experiences for training the control policy [13, 33]), etc. Our work is inspired by [7], which addresses the problem of error in long-horizon model dynamics prediction. [7] presents a hybrid algorithm that uses the model to simulate the short-term horizon and Q-learning to estimate the long-term value beyond the simulation horizon. Unlike the previous model-based RL works, our work is the first to introduce model-based RL for enabling game-theoretic credit assignment in multiagent continuous control.

6 CONCLUSIONS

In this paper, we studied multiagent robotic continuous control with model-based game-theoretic credit assignments. We first modelled robot joints as a fully-cooperative multiagent system, and optimised the system using a multiagent version of PPO. We then proposed a generic game-theoretic credit assignment framework using semivalues for evaluating agent-specific advantage functions. Furthermore, we proposed a model-based framework which significantly improved estimation of coalition values, which empowers game-theoretic credit assignments for multiagent continuous control. Finally, we empirically demonstrate that our model-based credit assignments leads to sample-efficient and robust multiagent learning on MuJoCo robot locomotion tasks. Future directions include studying other continuous control algorithms and robot variants (e.g., real robots), and analysis on the model-based RL methods with other configurations (e.g., longer prediction horizons).

ACKNOWLEDGMENTS

Wooldridge was supported by a UKRI AI World Leading Researcher Fellowship (grant number EP/W002949/1

REFERENCES

[1] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *arXiv preprint arXiv:1707.01495*, 2017.

[2] John F Banzhaf III. Weighted voting doesn't work: A mathematical analysis. *Rutgers L. Rev.*, 19:317, 1964.

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[4] Georgios Chalkiadakis, Edith Elkind, and Michael Wooldridge. Computational aspects of cooperative game theory. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 5(6):1–168, 2011.

[5] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *arXiv preprint arXiv:1805.12114*, 2018.

[6] Pradeep Dubey, Abraham Neyman, and Robert James Weber. Value theory without efficiency. *Mathematics of Operations Research*, 6(1):122–128, 1981.

[7] Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I Jordan, Joseph E Gonzalez, and Sergey Levine. Model-based value estimation for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101*, 2018.

[8] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pages 49–58. PMLR, 2016.

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

[10] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017.

[11] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[12] Dongge Han, Michael Wooldridge, Alex Rogers, Shruti Tople, Olga Ohrimenko, and Sebastian Tschiatschek. Replication-robust payoff-allocation for machine learning data markets. *arXiv preprint arXiv:2006.14583*, 2020.

[13] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *arXiv preprint arXiv:1906.08253*, 2019.

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[15] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[16] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[17] Jiahui Li, Kun Kuang, Baoxiang Wang, Furui Liu, Long Chen, Fei Wu, and Jun Xiao. Shapley counterfactual credits for multi-agent reinforcement learning. *arXiv preprint arXiv:2106.00285*, 2021.

[18] 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. *arXiv preprint arXiv:1509.02971*, 2015.

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

[20] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *arXiv preprint arXiv:1706.02275*, 2017.

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[22] Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566. IEEE, 2018.

[23] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*, 2018.

[24] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

[25] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. *arXiv preprint arXiv:2006.10800*, 2020.

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

[27] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

[28] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

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

[30] L. S. Shapley. *17. A Value for n-Person Games*, pages 307–318. Princeton University Press, 2016.

[31] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[32] Mark W Spong, Seth Hutchinson, Mathukumalli Vidyasagar, et al. *Robot modeling and control*, volume 3. wiley New York, 2006.

[33] Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

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

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

[36] 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*, pages 5026–5033. IEEE, 2012.

[37] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. Qplex: Duplex dueling multi-agent q-learning. *arXiv preprint arXiv:2008.01062*, 2020.

[38] Jianhong Wang, Jinxin Wang, Yuan Zhang, Yunjie Gu, and Tae-Kyun Kim. Shaq: Incorporating shapley value theory into q-learning for multi-agent reinforcement learning. *arXiv preprint arXiv:2105.15013*, 2021.

[39] Jianhong Wang, Yuan Zhang, Tae-Kyun Kim, and Yunjie Gu. Shapley q-value: a local reward approach to solve global reward games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7285–7292, 2020.

[40] Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. Nervenet: Learning structured policy with graph neural networks. In *International Conference on Learning Representations*, 2018.

[41] David H Wolpert and Kagan Tumer. Optimal payoff functions for members of collectives. In *Modeling complexity in economic and social systems*, pages 355–369. World Scientific, 2002.

[42] Michael Wooldridge. *An introduction to multiagent systems*. John wiley & sons, 2009.

[43] Jing Xu, Fangwei Zhong, and Yizhou Wang. Learning multi-agent coordination for enhancing target coverage in directional sensor networks. *arXiv preprint arXiv:2010.13110*, 2020.

[44] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of mappo in cooperative, multi-agent games. *arXiv preprint arXiv:2103.01955*, 2021.