Networked Agents in the Dark: Team Value Learning under Partial Observability

Guilherme S. Varela
Instituto Superior Técnico, INESC-ID
Lisbon, Portugal
guilherme.varela@tecnico.ulisboa.pt

Alberto Sardinha
PUC-Rio
Rio de Janeiro, Brazil
sardinha@inf.puc-rio.br

Francisco S. Melo
Instituto Superior Técnico, INESC-ID
Lisbon, Portugal
fmelo@inesc-id.pt

ABSTRACT

We propose a novel cooperative multi-agent reinforcement learning (MARL) approach for networked agents. In contrast to previous methods that rely on complete state information or joint observations, our agents must learn how to reach shared objectives under partial observability. During training, they collect individual rewards and approximate a team value function through local communication, resulting in cooperative behavior. To describe our problem, we introduce the networked dynamic partially observable Markov game framework, where agents communicate over a switching topology communication network. Our distributed method, DNA-MARL, uses a consensus mechanism for local communication and gradient descent for local computation. DNA-MARL increases the range of the possible applications of networked agents, being well-suited for real world domains that impose privacy and where the messages may not reach their recipients. We evaluate DNA-MARL across benchmark MARL scenarios. Our results highlight the superior performance of DNA-MARL over previous methods.

KEYWORDS

Artificial Intelligence; Multi-agent Systems; Reinforcement Learning; Deep Learning; Partial Observability

ACM Reference Format:
Guilherme S. Varela, Alberto Sardinha, and Francisco S. Melo. 2025. Networked Agents in the Dark: Team Value Learning under Partial Observability. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 9 pages.

1 INTRODUCTION

Cooperative multi-agent reinforcement learning involves rational agents learning how to behave under uncertainty in a shared environment to maximize a single utility function. While distributed training has once been the dominating paradigm in learning for MARL [2] systems, the community's focus has shifted to centralized training and decentralized execution (CTDE) in recent years. The fundamental reason is that centralized training agents benefit from a either a single loss function to train a common policy, e.g., parameter sharing [8], or from agent-wise policies that can be factorized between agents, e.g., Q-MIX [17].

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

Centralized training is ideal in settings where data is centralized. During training, agents benefit from sharing information, such as *joint observations* for partial observability mitigation, *joint actions* for modeling teammates and *team rewards* for cooperation. However, CTDE has its limitations. It assumes the existence of a central node (or entity) that actually trains the agents, knows all system information, performs all the necessary computations, and then distributes the resulting individual policies to agents for execution.

Distributed training with decentralized execution has re-emerged, e.g, [26] and [3], as an alternative to CTDE systems in real world domains where there is no central entity capable of performing computations in behalf of the agents. For instance, in scenarios like distributed economic dispatch [22], where agents collaborate to determine optimal power generation, it is crucial to preserve the privacy of agents' observations–their power generation and cost curves. This privacy protection is essential for the fair bidding in the sale or purchase of energy. Hence agents collaborate to achieve a common goal, but they are less forthcoming about sharing their own observations. Another example of application is distributed packet routing in a dynamically changing networks [1]. Agents are nodes and by using *local* observations and collecting *individual rewards*, must balance the selection of routes that minimize the number of "hops" of a given packet, against the risk of overflowing links along popular routes.

In this work, we advance upon the decentralized training and decentralized execution (DTDE) [7] paradigm, wherein *networked agents* use peer-to-peer communication during training and operate in isolation during execution. Prior work has produced networked agents under relaxed assumptions: Zhang et al. [26] assumed a fully observable state while the rewards are kept private, and Chen et al. [3] proposed networked agents that choose when and to whom request observations. In contrast, we introduce a novel approach that is not bound by the same restrictions and our agents learn under partial observability. The key to our method is the use of a consensus mechanism to force agents to agree on a team value, resulting in cooperative value function learning under the partial observability setting.

In summary, our key contributions can be outlined as follows. First, we formalize the *networked dynamic partially observable Markov game* (ND-POMG), a specialized framework derived from *partially observable Markov game* where agents communicate over a switching topology network. Second, we present a novel approach, DNA-MARL, for solving ND-POMG problems with a team policy gradient. This approach is implemented in an actor-critic algorithm and extended to the deep Q-network algorithm, showing its generality. Finally, we evaluate our approach and show that it outperforms other decentralized training, decentralized execution systems.

2 BACKGROUND

Partially observable Markov game (POMG): We define a *partially observable Markov game* [14] for N agents as the tuple:

$$(\mathcal{N}, \mathcal{S}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, \{O^i\}_{i\in\mathcal{N}}, \mathcal{P}, \{r^i\}_{i\in\mathcal{N}}, \gamma),$$

where $\mathcal{N} = \{1, \ldots, N\}$ denotes a set of N agents. $\mathcal{S}$ represents the *state space* describing the system, which is not observed. Instead, at each time step t, each agent $i \in \mathcal{N}$ observes $o_t^i \in O^i$, that depends on the true system state $s \in \mathcal{S}$. The set of *actions* available to agent i is denoted by $\mathcal{A}^i$. The joint action set $\mathcal{A}$ is the Cartesian product of the individual action spaces, *i.e.*, $\mathcal{A} = \mathcal{A}^1 \times \cdots \times \mathcal{A}^N$. The transition probability $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ denotes the probability distribution over the next state, they depend on the joint action $a \in \mathcal{A}$ and the current state s. The instantaneous individual reward for agent i is given by $r^i : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$; $\gamma \in [0, 1)$ is a discount factor.

Actor-critic: is a class of model-free reinforcement learning (RL) algorithms aimed at optimizing the policy π_θ, parameterized by $\theta \in \Theta$. Particularly, the *actor* component updates the policy π_θ, and the *critic* component evaluates the actor's policy performance by using the Q-function:

$$Q(s, a; \theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{k=t}^{\infty} \gamma^{k-t} r_{k+1} | s_t = s, a_t = a\right].$$

The Q-function yields the expected discounted return by taking action a on state s at time t, and then following π_θ thereafter. In the single agent setting, the policy gradient theorem [21] prescribes the direction for the gradient updates to maximize the *total discounted return*:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log\left(\pi(a_t|s_t;\theta)\right) Q(s_t, a_t; \theta)\right], \tag{1}$$

Actor-critic with advantage (A2C) [5], in the single agent episodic setting, the history of the interactions with the environment are collected into trajectories. A mini-batch is the concatenation of many trajectories, drawn from the same policy using parallel processing. A2C maintains one neural network for the actor, and one another for the critic, their weights are adjusted via gradient descent. The critic updates its parameters ω by minimizing the *least mean squares* loss function:

$$\mathcal{L}(\omega; \tau) = \frac{1}{T} \sum_{(s,a)\in\tau} ||A(s, a; \omega)||_2^2, \tag{2}$$

where T is the length of an episode, $A(s, a; \omega) = Q(s, a; \omega) - V(s; \omega)$ is the *advantage function*, and the value function

$$V(s; \omega) = \mathbb{E}_{\pi_\theta}\left[\sum_{k=t}^{T} \gamma^{k-t} r_{k+1} | s_t = s, \omega\right],$$

captures the discounted return for being on state s at time t, and then following π_θ thereafter until the episode's end at T. The advantage function reduces the variance of the actor-critic gradient updates. The actor updates its parameters θ by minimizing the loss function:

$$\mathcal{L}(\theta; \tau) = -\sum_{(s,a)\in\tau} \log\left(\pi(a|s;\theta)\right) A(s, a; \omega). \tag{3}$$

Consensus: The goal of randomized consensus algorithms is to asymptotically reach an agreement on the global average of individual parameters held by nodes in a switching topology communication network through local communication. Formally, the switching topology communication network is defined by an undirected graph $\mathcal{G}_k(\mathcal{N}, \mathcal{E}_k)$, where $\mathcal{N} = \{1, \ldots, N\}$ is the node set, and $\mathcal{E}_k \subseteq \mathcal{N} \times \mathcal{N}$ denotes the time-varying edge set with respect to communication step k[1]. Nodes n and m can communicate at communication step k, if and only if, $(n, m) \in \mathcal{E}_k$. Each node n, initially holding a parameter $\phi^n(0)$, has the opportunity at each communication step k, to synchronously interact with its neighbors, updating its parameter value by replacing its own parameter with the average of its parameter and the parameters from neighbors. The distributed averaging consensus algorithm [24] prescribes the updates:

$$\phi^n(k+1) = \sum_{m\in\mathcal{N}_k^n} W_k^{n,m} \cdot \phi^m(k), \tag{4}$$

where $\mathcal{N}_k^n = \{m | (n, m) \in \mathcal{E}_k\}$ represents the neighborhood of agent n at time k. For a switching topology dynamic with random link dropouts, it is possible to show that in the limit, the values of the parameters for each node n converge to the network's average, *i.e.*:

$$\lim_{k\to\infty} \phi^n(k) = \frac{1}{N} \sum_{n}^{N} \phi^n(0). \tag{5}$$

Moreover, for an arbitrary graph $\mathcal{G}_k(\mathcal{N}, \mathcal{E}_k)$, it is possible to derive the weights $W_k^{n,m}$ that guarantee consensus locally. For instance, the *Metropolis weights* matrix [24] in (Appendix A [23][2]) is a matrix that guarantee consensus, requiring only that each node be aware of its closest neighbor degree.

Networked agents is a class of distributed reinforcement learning agents that combines consensus iterations in (4) for localized approximations and actor-critic updates in (3) and (2). Relevant previous works include:

Critic consensus: Zhang et al. [26] introduce networked agents where the critic network $V(\cdot, \cdot; \omega)$, parameterized with ω, approximates the value-function $V^\pi(\cdot)$. The distributed critic emulates a central critic. Agents observe the transition (s_t, a_t, s_{t+1}), perform critic the update in (2), then agents average the parameter using consensus:

$$\omega^i(k+1) = \sum_{j\in\mathcal{N}_k^i} W_k^{i,j} \cdot \omega^j(k) \quad \forall i \in \mathcal{N}. \tag{6}$$

Policy consensus: Chen et al. [3] introduce the class of homogeneous Markov games wherein there is no suboptimality incurred by performing consensus on the actor parameters. Their motivation is to emulate parameter sharing under the decentralized setting, while minimizing the number of communication rounds. Agents perform the actor update in (3), then average the parameters using consensus:

$$\theta^i(k+1) = \sum_{j\in\mathcal{N}_k^i} W_k^{i,j} \cdot \theta^j(k) \quad \forall i \in \mathcal{N}. \tag{7}$$

[1] In this work t represents timesteps in episodic interactions with the environment, while k represents communication timesteps (rounds) that occur between episodes. Communication only happens during training, in between episodes. During execution agents are *fully decentralized*.

[2] This article's preprint version.

3 NETWORKED DYNAMIC POMG

In this section, we present the first key contribution, which is a formalization of *networked dynamic partially observable Markov game*, ND-POMG. We define the ND-POMG as the septuple:

$$\mathcal{M} = (\mathcal{G}_k, \mathcal{S}, \{\mathcal{O}^i\}_{i\in\mathcal{N}}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, \mathcal{P}, \{r^i\}_{i\in\mathcal{N}}, \gamma),$$

where $\mathcal{G}_k(\mathcal{N}, \mathcal{E}_k)$ represents a switching topology communication network, and the latter six elements represent the POMG elements.

In this work, we fix the agents set $N = |\mathcal{N}|$, to ensure that no agent is added or removed from the network. We also introduce the hyperparameter $C = |\mathcal{E}_k|$ for all k, that shapes the topology of the communication network by fixing the cardinality of every possible edge set $\mathcal{E}_k$. Moreover, we let $\mathcal{E}_k$ change according to an uniform distribution at each communication round. The uniform distribution over the edge sets is the least specific distribution that guarantees that over a sufficiently long round of communications agents will reach consensus (Appendix B.1 [23]).

4 DOUBLE NETWORKED AVERAGING MARL

This section presents our second key contribution which is the DNA-MARL an approach to solve ND-POMG problems. Since our method requires an extra consensus iteration step, we call it *double networked averaging* MARL (DNA-MARL). Any single agent reinforcement learning algorithm can be cast as a DNA-MARL with our method, we elaborate the case for the A2C, an *on-policy* method (Sec. 4.1) and extend to the deep Q-network (DQN) (Sec. 4.2), an *off-policy* method.

4.1 Double Networked Averaging A2C

In order to make agents cooperate with decentralized training, we factorize the shared objective between agents. Hence, it is possible to maximize performance via local communication and local gradient descent updates.

The *total discounted team return*, $J(\pi_\theta)$, serves as a measure of the joint policy π_θ performance:

$$J(\theta) = \mathbb{E}_{\substack{s_0\sim\mu(\cdot)\\ a\sim\pi_\theta(\cdot|s)}}\left[\sum_{t=0}^{T}\gamma^t r_{t+1}\right], \tag{8}$$

where $r_{t+1} = \frac{1}{N}\sum_{i\in\mathcal{N}} r^i_{t+1}$ is the *instantaneous team reward*. The expectation is taken by drawing the initial state s_0 from the initial state distribution μ and taking actions from π_θ, thereafter. For simplicity, we follow the convention of writing $J(\pi_\theta)$ as $J(\theta)$.

4.1.1 Team Policy Gradient. To obtain the team policy gradient, we replace the team reward in (4.1) by the average of *individual rewards*.

$$\begin{aligned} J(\theta) &= \mathbb{E}_{\substack{s_0\sim\mu(\cdot)\\ a\sim\pi_\theta(\cdot|s)}}\left[\sum_{t=0}^{T}\gamma^t\Big(\frac{1}{N}\sum_{i\in\mathcal{N}} r^i_{t+1}\Big)\right] \\ &= \sum_{i\in\mathcal{N}}\mathbb{E}_{\substack{s_0\sim\mu(\cdot)\\ a\sim\pi_\theta(\cdot|s)}}\left[\sum_{t=0}^{T}\frac{\gamma^t}{N} r^i_{t+1}\right] = \sum_{i\in\mathcal{N}} J^i(\theta) \end{aligned}$$

The result above suggests how the cooperative system's objective can be distributed across the participating agents, thus the total discounted team return is computed as the weighted sum of the discounted individual rewards. However, the behaviors of the agents are still coupled, depending on the joint policy parameterized by $\theta \in \Theta$ and on the common system state s_t. Formally, the objective of the cooperative distributed system is to maximize the *total discounted team return*:

$$\max_\theta \sum_{i\in\mathcal{N}} J^i(\theta) \text{ with } J^i(\theta) = \mathbb{E}_{\tau\sim\mathbb{P}(\cdot|\theta)}\left[\sum_{t=0}^{T}\gamma^t r^i_{t+1}\right]. \tag{9}$$

We drop the scaling constant N as it does not change the stationary points of the maximization. $\mathbb{P}(\cdot|\theta)$ is the short hand notation for the probability distribution of the trajectories,

$$\tau = \big(s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_T\big),$$

generated from system dynamics $\mathcal{P}$ under the joint policy $\pi_\theta(a|s)$. The maximization can be achieved through iterative gradient search methods. More specifically, the policy gradient theorem (1) prescribes the direction of the parameter updates:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\mathbb{P}(\cdot|\theta)}\left[\nabla_\theta \log\big(\pi_\theta(a|s)\big) A_\theta(s,a)\right],$$

that maximize the total discounted team return. We replaced the Q-function by the *advantage function* (2) to mitigate the variance on the weight updates. We note that the single agent policy gradient update in (1) serves as the policy gradient for a centralized agent in control of all agents. Departing from the centralized setting, we use the fact that the joint policy π_θ factorizes between agents, to set:

$$\begin{aligned} \nabla_\theta J(\theta) &= \mathbb{E}_{\tau\sim\mathbb{P}(\cdot|\theta)}\left[\nabla_\theta \log\Big(\Pi_{i\in\mathcal{N}}\pi^i_{\theta^i}(a^i|s)\Big) A_\theta(s,a)\right] \\ &= \mathbb{E}_{\tau\sim\mathbb{P}(\cdot|\theta)}\left[\Big(\sum_{i\in\mathcal{N}}\nabla_{\theta^i}\log\big(\pi^i_{\theta^i}(a^i|s)\big)\Big) A_\theta(s,a)\right] \\ &= \sum_{i\in\mathcal{N}}\mathbb{E}_{\tau\sim\mathbb{P}(\cdot|\theta)}\left[\nabla_{\theta^i}\log\big(\pi^i_{\theta^i}(a^i|s)\big) A_\theta(s,a)\right]. \end{aligned} \tag{10}$$

There are two limitations in (10) preventing its use for conducting local updates. In the context of partial observability, the states s are unavailable in trajectory τ. Second, the gradient update depends on global $\theta \in \Theta$, and no agent has access to θ .

4.1.2 Distributed Reinforcement Learning. We address the limitations in (10) by considering the *information structure* of the problem, or what do agents know [25]. We propose localized approximations that allow agents to perform local updates: considering a synchronous system where agents interact with the environment to collect their individual trajectories τ^i. Distributed learning *requires* a localized approximation $\nabla_\theta J^i(\theta)$ for gradient of the discounted team return $\nabla_\theta J(\theta)$ in (10). Moreover, each agent maximizes its policy π^i_θ which is parameterized by θ^i, *i.e.*, $\pi^i_\theta = \pi^i_{\theta^i}$ and $\nabla_\theta J^i(\theta) = \nabla_{\theta^i} J^i(\theta)$. By combining these three facts, the localized approximation for (10) can be rewritten as:

$$\nabla_{\theta^i} J(\theta^i) = \mathbb{E}_{\tau^i\sim\mathbb{P}(\cdot|\theta)}\left[\nabla_{\theta^i}\log\Big(\pi^i_{\theta^i}(a^i|o^i)\Big) A^i_\theta(o^i,a^i)\right], \tag{11}$$

where $\tau^i = \big(o^i_0, a^i_0, r^i_1, o^i_1, a^i_1, r^i_2, \ldots, o^i_T\big)$ is available locally for agent i. The replacement s^i_t with o^i_t under the partially observability setting is standard practice in MARL literature [3, 17, 20]. The system's dynamics still depend on the joint behavior, parameterized by θ, but the gradient in (11) is locally defined. Straightforward application of the actor-critic updates in (3) and (2), with individual

rewards over local parameters lead to *independent learners*, which evaluate their individual policies. Individual learners assume that the approximation $A^i_{\theta^i}(o^i, a^i) \approx A_\theta(o, a)$ holds.

4.1.3 Distributed Cooperation. We propose a better approximation for the gradient of the discounted team return by performing the updates in the direction of the *team advantage* $A_\theta(s, a)$ in (10), rather than the local advantage $A^i_\theta(o^i, a^i)$ in (11). However, since the team advantage is unavailable, agents should instead perform local updates in the direction of the *team advantage under the partially observable* setting $A_\theta(o, a)$. Since $o = \left[o^1, \ldots, o^N\right]$ is the concatenation of observations, $A_\theta(o, a)$ can be defined by:

$$A_\theta(o, a) = Q(o, a; \omega) - V(o; \omega) \approx r + \gamma V(o'; \omega_-) - V(o; \omega), \quad (12)$$

where, r and o' are respectively the rewards and the joint observations on the next time step. The parameter ω_- is a periodic copy of the critic's parameters $\omega \in \Omega$, which serves to stabilize learning. Decentralized learning agents neither observe o nor collect r, but may resort to local communication schemes to obtain factorized representations for $V(o, a; \omega)$. The local critic update is a straightforward adaptation of the single agent critic in (2):

$$\mathcal{L}\left(\omega^i; \tau^i\right) = \frac{1}{T} \sum_{t=0}^{T-1} \left(y^i_t - V(o^i_t; \omega^i)\right)^2, \quad (13)$$

with

$$y^i_t = r^i_{t+1} + \gamma V(o^i_{t+1}; \omega^i_-).$$

We propose to use communication to combine y^i_t by performing team-V consensus:

$$y^i_t(k+1) = \sum_{j \in \mathcal{N}^i_k} W^{i,j}_k \cdot y^j_t(k) \quad \forall i \in \mathcal{N}, k = 1, \ldots, K. \quad (14)$$

After each training episode, agents concurrently approximate the team-V using y^i_t based on their individual rewards and observations. Then, we let K consensus updates per mini-batch aimed at approximating the team-V. At each communication round, a connected agent averages its team-V estimation with team-Vs from neighbors. Ideally, the following approximation will hold:

$$\bar{y}^i_t = \sum_{i=1} \frac{1}{N} \left[r^i_{t+1} + \gamma V(o^i_{t+1}; \omega^i_-)\right] \approx \sum_{i=1} \frac{1}{N} V(o^i_t; \omega^i_-). \quad (15)$$

We empirically test for suitable values of K. The consensus steps in (14) result in a flexible degree of cooperation: When $K = 0$, agents behave as *independent learning* agents. For a high enough values of K, the approximation error should be small enough, such that agents recover the same team-V.

This section concludes a core contribution to our method: distributed cooperation whereby agents produce localized approximations for a team-V (or team-Q) using consensus. Cooperation requires that each agent approximates the same critic, and this critic must evaluate the joint policy, so that the actor updates its parameters in the direction of the team-V, thus the best local actions for the team will be reinforced. Moreover, the updates in (15) do not require agents to be homogeneous. Previous networked agents works [3, 26] provide asymptotic convergence guarantees for linear function approximation on the state-action space. Under the linearity approximation on the critic and fully observable setting the update in (6) is sufficient to guarantee cooperation. Under partial observability and/or non-linear function observation agents are unable to obtain localized approximations for the team-V/team-Q by the averaging of critic parameters.

4.1.4 Algorithm. To design our algorithm, we combine the localized approximations for the team-V in (15) with critic consensus in (6) [26]. And actor consensus in (7) [3] for improving sample efficiency. Hence, the updates comprising the double networked averaging actor critic with advantage are given by:

$$y^i_t(k+1) = \sum_{j \in \mathcal{N}^i_k} W^{i,j}_k \cdot y^j_t(k) \quad k = 1, \ldots, K \quad \text{(i)}$$

$$\bar{y}^i_t = y^i_t(K+1) \quad \text{(ii)}$$

Every agent interacts locally with the environment to collect the individual trajectories τ^i. Then, in (i) agents use localized approximation for team-V, $\bar{y}^i_t$, by performing K consensus steps; (ii) the final approximation for the team-V is defined; The next steps consist of local weight updates:

$$\mathcal{L}\left(\omega^i; \tau^i, \bar{y}^i\right) = \frac{1}{T} \sum_{t=0}^{T-1} \left(\bar{y}^i_t - V(o^i_t; \omega^i)\right)^2 \quad \text{(iii)}$$

$$\mathcal{L}\left(\theta^i; \tau^i, \bar{y}^i\right) = -\frac{1}{T} \sum_{t=0}^{T-1} \log \pi(a^i_t | o^i_t; \theta^i)(\bar{y}^i_t - V(o^i_t; \omega^i)) \quad \text{(iv)}$$

In (iii) the local critic updates its parameters using $\bar{y}^i_t$ instead of their own estimations y^i_t; (iv) Similarly, actor updates its parameters in the direction of team-V; Finally, periodically agents perform actor and critic parameter consensus, represented by steps (v) and (vi):

$$\omega^i(k+1) = \sum_{j \in \mathcal{N}^i_k} W^{i,j}_k \cdot \omega^j(k) \quad k = 1, \ldots, K \quad \text{(v)}$$

$$\theta^i(k+1) = \sum_{j \in \mathcal{N}^i_k} W^{i,j}_k \cdot \theta^j(k) \quad k = 1, \ldots, K \quad \text{(vi)}$$

We note that actor-critic parameters can be concatenated to avoid extra communication rounds, and that (v) utilizes (6) and that (vi) utilizes (7). The consensus updates in (v) require agents to have homogeneous observation spaces, while updates in (vi) require agents to have homogeneous action spaces.

Figure 1 illustrates the information flow of the algorithm, clockwise from the left: (a) Four agents (circles) interact with the environment and evaluate a team-V (blue gradient) from their individual experiences; (b) Consensus on team-V occurs over the time varying communication network. At each step, certain agents aggregate their opinions on the team-V (14); (c) Agents independently update their parameters using gradient descent, resulting in varying actor-critic evaluations and policies (orange gradient); (d) Parameter consensus occurs periodically over the time varying communication network. At each step, certain agents aggregate their opinions on the parameters in ([3, 26]). Pseudo codes for double networked

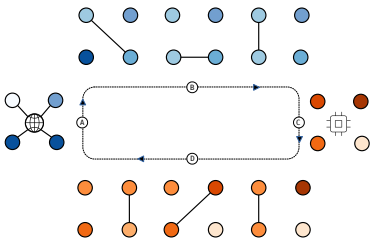


Figure 1: Diagram illustrating the information flow from the algorithm, clockwise from the left.

agents are provided in (Appendix B.2 [23]) and we open source the codebase DNA-MARL[3].

4.2 Double Networked Averaging Q-learner

We extend the DNA method to the independent Q learner. To compute the team-Q consensus, which involves the averaging of individual Q-function evaluations, agents must first determine locally the *learning target*,

$$y_t^i = r_{t+1}^i + \gamma \max_{a' \in \mathcal{A}^i} Q(o_{t+1}^i, a'; \theta_-^i). \tag{16}$$

Similarly to single agent DQN [13], the learning target consists of the sum of the local reward r_{t+1}^i and the Q-value assigned to the individual action a' that yields the highest Q-value over the next observation o_{t+1}^i. Moving from the V-function in (13) to Q-function in (16), the learning target becomes a function of a max operator, which is performed locally. The consensus updates for the team-Q are then calculated as follows:

$$y_t^i(k+1) = \sum_{j \in \mathcal{N}_k^i} W_k^{i,j} \cdot y_t^j(k) \quad \forall t \in \tau^i, \tag{17}$$

which is performed K times. We thus assign the result from the consensus steps $y_t^i(K+1)$ to the variable $\bar{y}_t^i$. The third step is the parameter update:

$$\mathcal{L}(\theta^i; \tau^i, \bar{y}^i) = \frac{1}{|\tau^i|} \sum_{\tau^i \in \tau^i} \left(\bar{y}_t^i - Q(o_t^i, a_t^i; \theta)\right)^2. \tag{18}$$

Finally, the fourth step consists in parameter consensus:

$$\theta^i(k+1) = \sum_{j \in \mathcal{N}_k^i} W_k^{i,j} \cdot \theta^j(k),$$

for K rounds. As a result, agents obtain a local approximation of a common average θ. However, for a finite number of consensus steps K, it is impossible to guarantee that agents will obtain identical copies $\theta^1 = \cdots = \theta^N = \theta$. Hence, agents are left with the parameters from this finite step approximation.

5 EXPERIMENTS

We evaluate the performance of DNA-MARL following the methodology outlined by Papoudakis et al. [16] for benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. This section presents the scenarios, baselines, and evaluation metrics.

[3]https://github.com/GAIPS/DNA-MARL

5.1 Scenarios

Multi-agent environments are typically designed for the cooperative setting [15], but they can also be configured for the mixed setting. In the mixed setting, individual rewards are emitted, and the team reward is obtained by averaging all individual rewards. With minor adaptations which we outline briefly, the multi-agent particle environment (MPE) [12] scenarios were adjusted for partial observability and individual rewards. The scenarios include:

Adversary[4]: The first MPE adaptation has two teammates protecting a target landmark from a third adversary agent. Teammates are rewarded the adversary's distance from the target and penalized with their negative distance to the landmark. The teammates observations include the position and color from the closest agent (either adversary or teammate), their relative distance to landmark, and the position of the two landmarks.

Spread[4]: The second MPE adaptation has three agents that must navigate to three landmarks while incurring a penalty for collisions. We adapt the observation and reward for the partially observable and decentralized setting. Each agent's observation contains its own absolute location, the relative locations of the nearest agent, and the relative location of the nearest landmark. The reward is the negative distance of the agent to the closest landmark.

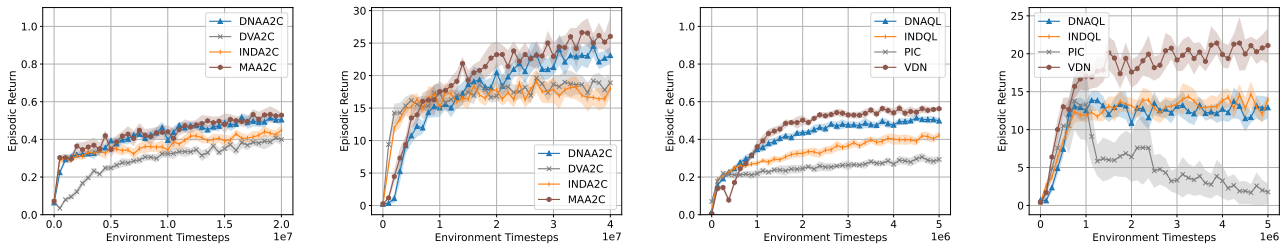
Tag[4]: The third MPE adaptation has three big predators (agents) that rewarded for catching a smaller and faster fleeing agent that follows a pre-trained policy. Additionally, two landmarks are placed as obstacles. Agents navigate a two-dimensional grid with continuous coordinates. The reward is sparse, and we adapt the environment for partial observability and decentralization. Each agent's observation includes its own position and velocity, the closest predator's position, and the prey's position and velocity. The reward is individual where the agent that catches the prey is the one receiving a reward of ten points.

Level-Based Foraging (LBF) [16][5]: In this scenario, agents can move on a two-dimensional discrete position grid and collect fruits. Since both agents and fruits have associated levels, successful fruit loading occurs only if the total level of the agents attempting to load it exceeds the fruit's level. Observations consist of relative positions of agents and fruits within a two-block radius centered around the agent. The rewards are sparse, and only the agents that successfully load a fruit receive positive reward. We configure three instances in the partially observable setting, in increasing levels of difficulty: (i) Easy: 10 x 10 grid, 3 players, and 3 fruits (ii) Medium: 15 x 15 grid, 4 players, and 5 fruits (iii) Hard: 15 x 15 grid, 3 players, and 5 fruits.

5.2 Baselines

Following Papoudakis et al. [16], we divide our experiments into the *on-policy* and *off-policy* settings. Furthermore, for each scenario, we compare three different approaches: (i) individual learners (IL), (ii) decentralized training and fully decentralized execution (DTDE) and (iii) the centralized training and decentralized execution (CTDE) algorithms. ILs are always self interested and CTDE are always fully cooperative as they have access to the team reward. For the on-policy setting, the baselines include:

[4]https://github.com/GAIPS/multiagent-particle-envs.
[5]https://github.com/semitable/lb-foraging.



(a) (on-policy) LBF Hard (b) (on-policy) MPE Tag (c) (off-policy) LBF Hard (d) (off-policy) MPE Tag

Figure 2: From left to right, episodic returns for on-policy setting and episodic returns for the off-policy setting, for two selected tasks. We plot the 95% bootstrap CI for each algorithm. In chestnut, CTDE algorithms that establish the upper bound of performance. In grey, DTDE algorithms that are DNA-MARL's closest competitors. In orange, IL algorithms that establish a lower bound on performance. We can see that for three algorithms-environments combinations, (a), (b), (c), DNA-MARL (in blue) has the closest performance to the upper bound.

Multi-Agent Actor-Critic with Advantage (MAA2C) [16]: This is a CTDE algorithm with a central critic that has more information during training: (i) it has access to the *joint reward*, (ii) the central critic has access to the concatenation of the all agents' observation, and (iii) uses parameter sharing. Hence, it can compute the *team advantage under partial observability* in (12) precisely.

Distributed-V with Advantage (DVA2C): This DTDE algorithm implements networked agents in Algorithm 2, of Zhang et al. [26], where it performs a consensus on the critic's network parameters. Moreover, it is a model-based algorithm, whereby it has a neural network that estimates the discounted team return.

Independent Actor-Critic with Advantage (INDA2C): This IL algorithm implements updates in (3) and (2) and generates self-interested agents.

In addition, for the off-policy setting, we specifically use the following baselines:

Value Decomposition Networks (VDN) [20]: A CTDE algorithm that learns a central Q-function that can be factorized among agents.

Permutation Invariant Critic (PIC) [11]: Since the DTDE implementation of Chen et al. [3] is not open-sourced, we represent its implementation using the CTDE algorithm in which it was based. PIC has a central critic that employs a graph convolution neural network [10] that learns from joint observations, joint actions and joint rewards. Resulting in Q-function representations that remain unchanged regardless of the ordering of the concatenation of observations and actions from the agents. Since PIC sets an upper bound in performance for the networked agents proposed by Chen et al. [3] the comparison is fair.

Independent Q-Learner (INDQL): This IL algorithm implements deep Q-network updates [13] and generates self-interested agents.

5.3 Evaluation

In this section we establish the performance metric, the deviation metric and the hypothesis test to discriminate results.

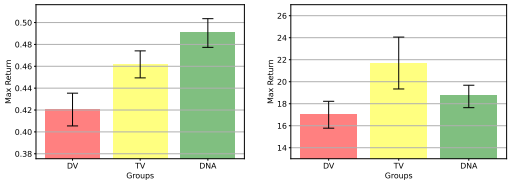
Performance metrics: We evaluate the performance of the algorithms using the maximum average episodic returns [16] criteria. For each algorithm, we perform forty one evaluation checkpoints during training each comprising of a hundred rollouts. Then we identify the evaluation time step at which the algorithm achieves the highest average evaluation returns across ten random seeds. Moreover, for each algorithm-scenario combination we report the 95% bootstrap confidence interval (CI) constructed by resampling the empirical maximum average episodic returns ten thousand times.

Bootstrap hypothesis test [4]: In addition to reporting 95% bootstrap confidence interval, we gauge how similar two results are by evaluating a bootstrap hypothesis test[6]. The test's null hypothesis is that the means of both samples are the same. The test is performed by drawing an observation from each sample and computing their difference. This procedure is repeated a thousand times. Finally, from the resulting sample of differences we perform the 95% bootstrap confidence interval. If the CI doesn't contain zero, than we must reject the null hypothesis that both means are equal. We refer to Appendix C [23] for a detailed description of the experimental methodology, hyperparameters, and supplementary results.

6 RESULTS

Figure 2 presents a comparative analysis of DNA-MARL's performance for the on-policy and off-policy settings for two selected tasks from the LBF and MPE environments. The CTDE algorithms serve as an upper benchmark for other methods, while the IL algorithms establish a lower performance boundary. Notably, in three specific algorithm-task pairings – (a), (b), and (c) – DNA-MARL demonstrates superior results compared to its nearest competitors when utilizing decentralized training combined with decentralized execution strategies. These results indicate that our double networked averaging A2C (DNAA2C), an algorithm that learns using local observations, can indeed emulate a central critic that uses system-wide observations. In spite of having information loss due to randomized communication. For the off-policy setting, in the

[6] https://github.com/flowersteam/rl_stats



(a) LBF Hard **(b) MPE Tag**

Figure 3: Ablation for DNAA2C: From left to right, DV (distributed-V) group has critic consensus. TV (team-V) group has team-V consensus and critic consensus. DNA group has team-V consensus and both actor and critic consensus. We can see a performance improvement moving from DV to TV which highlights the impact of our contribution.

MPE environment for the Tag task, our double networked averaging Q-learner (DNAQL) has the performance comparable to the IL algorithm, while the alternative decentralized training algorithm has the worst performance. To improve the analysis of these results, we present the maximum average episodic return obtained per algorithm-scenario combination in Table 1.

In Table 1, results are separated into on-policy and off-policy settings, and the CTDE algorithms is outlined for each method. For each algorithm-scenario pairings, the values highlighted in bold represent the best results for the scenario. The asterisk shows the results of the bootstrap hypothesis test cannot reject the null hypothesis (*i.e.*, the performance is equal to the best performing algorithm for the scenario). The double asterisk indicates results that are worse than the highlighted result but still outperform the other algorithms, according to the bootstrap hypothesis test.

For the on-policy setting, we note, from six scenarios, there are four scenarios that the results are comparable to MAA2C's performance. In addition, in the remaining two scenarios, both results are very close. For the off-policy setting and LBF environment we observe similar outcomes: DNAQL ranks as second best. Moving to MPE environment for the off-policy setting, the IL method outperforms the CTDE for two scenarios, followed by DNAQL. This is not a surprising result, since Papoudakis et al. [16] state that for most MPE scenarios, VDN's assumption of additive value function decomposition is mostly violated for this environment. However, for the third scenario, Tag, VDN outperforms other decentralized algorithms by a wide margin. Here, additive value decomposition seems to have played a major role in the performance. Additive value decomposition limits the range of representable functions, but simplifies the learning over a large combined observation and actions spaces. Neither DNAQL or INDQL are guaranteed to generate additive value decomposition.

6.1 Ablations

To assess the impact of consensus steps within the DNA-MARL framework, we conducted ablations regarding the discounted return consensus outlined in (14), specifically focusing on both the critic parameters and actor parameters.

Figure 3 presents the ablation results for DNAA2C in the tasks LBF Hard and MPE Tag. The bars represent the averages within a neighborhood of size two around the maximum average episodic returns, while the intervals denote the 95% bootstrap confidence interval. Moving from left to right, we have three groups: (i) Distributed-V (DV) which conducts consensus on the critic parameters, (ii) Team-V (TV) which performs consensus on both the V-values and critic parameters, and (iii) DNA which carries out consensus on both the V-values and actor-critic parameters. Due to space restrictions the remaining ablation plots are reported in (Appendix D [23]). Here, we highlight how this ablation study connects DNA-MARL to previous works: Zhang et al. [26] propose critic parameter consensus (DV group). Our original contribution proposes consensus on the team-V (14) in addition to consensus update on the critic's parameters (TV group): While Fig. 3 (a) shows that team-V provides an improvement in overall performance for LBF Hard task, Fig. 3 (b) shows that team-V consensus provides a significant improvement in performance for the MPE Tag task. Finally, DNA-MARL combines team-V consensus with consensus on the agents' policies proposed by Chen et al. [3] emulating *parameter sharing* in the decentralized setting.

7 RELATED WORK

We relate our work with five other lines of research, two of which we present herein. Due to space restrictions, we further discuss related works in Appendix D [23].

Central Training and Decentralized Execution: CTDE is the prevailing approach in multi-agent reinforcement learning, where a *central critic* learns a system action-value function to mitigate the risk of non-stationarity. The policies are factorized and executed by individual actors, utilizing only local information to address the large state space problem. Examples from works that learn a central critic include MADDPG [12], COMA [6] and PIC [11]. Furthermore, actors benefit from *parameter sharing* as proposed by [8], wherein agents use a single neural network to approximate a policy trained with experiences collected from the behavior policies of all agents. Parameter sharing reduces wall clock time and increases sample-efficiency, enabling faster agent learning [9]. Another possibility is building utility functions that factorize into agent-wise function. Sunehag et al. [20] propose value decomposition networks, where the team-Q function is recovered by adding the agent-wise Q-functions. Finally, QMIX [17] extend VDN by proposing a mixing network, that has a dynamic set of parameters that vary according to the system state. The mixing network produces more expressive team-Q function decomposition, requiring that the joint action that minimizes the team Q-value be the same as the combination of the individual actions maximizing the agent-wise Q-values.

Networked agents with multi-agent reinforcement learning. Zhang et al. [26] is DTDE MARL system that apply the consensus mechanism over the critic's parameters to obtain a joint policy evaluation. However, their system requires full observability of both state and action spaces. In contrast Zhang and Zavlanos [27] propose DTDE MARL system that performs consensus on the actor's parameters while the critics are individual. As a limitation the policies must represent the joint action space. Chen et al. [3] apply networked agents to homogeneous Markov games, a subclass of Markov game, where agents observe different permutations of the state space but share the same action space, making individual

Table 1: Results for level-based foraging and multiagent particle environments: Maximum average episodic returns over ten independent runs and 95% bootstrap confidence interval. Highlighted results indicate the best performing algorithm. The asterisk indicates results that are not significantly different from the best result. Double asterisks indicate the second best result.

Methods	Algorithm	LBF			MPE		
		Easy	Medium	Hard	Adv.	Spread	Tag
on-policy	MAA2C (CTDE)	**0.96** (−0.01, 0.01)	**0.76** (−0.04, 0.04)	**0.53** (−0.04, 0.04)	17.39* (−0.56, 0.62)	−92.19* (−0.35, 0.36)	**26.63** (−1.68, 1.70)
	DNAA2C (ours)	0.93** (−0.01, 0.01)	0.75* (−0.02, 0.02)	0.52* (−0.02, 0.02)	**17.68** (−0.67, 0.68)	**−91.96** (−0.28, 0.24)	26.09* (−2.11, 2.01)
	DVA2C	0.83 (−0.01, 0.01)	0.67 (−0.04, 0.04)	0.41 (−0.02, 0.02)	16.46 (−0.89, 0.70)	−93.49 (−0.78, 0.78)	19.63 (−1.17, 1.07)
	INDA2C (IL)	0.89 (−0.02, 0.01)	0.69 (−0.03, 0.03)	0.45 (−0.03, 0.03)	16.30 (−0.68, 0.67)	−94.39 (−0.52, 0.52)	19.10 (−1.74, 1.90)
off-policy	VDN (CTDE)	**0.94** (−0.01, 0.01)	**0.79** (−0.02, 0.02)	**0.56** (−0.02, 0.02)	9.64 (−0.64, 0.77)	−94.77 (−0.26, 0.28)	**23.27** (−2.69, 2.69)
	DNAQL (ours)	0.88** (−0.01, 0.01)	0.75** (−0.02, 0.02)	0.51** (−0.02, 0.02)	12.51* (−1.05, 0.92)	−93.06* (−0.45, 0.48)	15.77 (−1.70, 1.88)
	PIC	0.48 (−0.03, 0.04)	0.48 (−0.02, 0.02)	0.31 (−0.02, 0.02)	11.10 (−0.96, 0.92)	−93.94 (−0.41, 0.39)	13.76 (−3.72, 3.94)
	INDQL (IL)	0.86 (−0.02, 0.01)	0.70 (−0.01, 0.01)	0.42 (−0.02, 0.02)	**13.61** (−1.13, 1.02)	**−92.69** (−0.33, 0.32)	15.54 (−1.10, 1.27)

Table 2: MARL settings. The codes for reward column: Individual (I), or team (T). The codes for state space observability (Observ.): Fully observable (FO), joint fully observable (JFO), and partially observable (PO). The codes for training (Train.) column: Centralized (C), or decentralized (D). The codes for the base Markov decision problem framework [14]: Markov game (MG), decentralized Partially observable Markov decision process (dec-POMDP), homogenous Markov game (HMG), and partially observable Markov game (POMG). The is homogeneous column requires a special structure on agents.

Works	Reward	Observ.	Train.	Base Framework	Communicates Observ.	Is Homogeneous
Lowe et al. [12]	T/I	PO	C	Dec-POMDP/POMG	No	Heterogeneous
Sunehag et al. [20]	T	PO	C	Dec-POMDP	No	Heterogeneous
Zhang et al. [26]	I	FO	D	MG	No	Heterogeneous
Chen et al. [3]	I	JFO	D	HMG	Yes	Homogeneous
DNA-MARL	I	PO	D	POMG	No	Homogeneous

agents interchangeable. To improve observability agents choose when and to whom communicate their observation. Differently from other approaches our agents obtain team-V estimation using consensus. Experimental results indicate that DNA-MARL outperforms both [26] and [3] under partially observable settings. Table 2 summarizes the differences between our DNA-MARL method and previous networked agents systems.

8 CONCLUSION AND FUTURE WORK

We propose the DNA-MARL that learn to cooperate in a ND-POMG under the decentralized training and fully decentralized execution paradigm. The key is performing consensus steps on the V-values. Our experiments show that DNA-MARL agents, with limited access to system information, can often reach the performance of their centralized training counter parts and outperform previous works. Moreover, the framework is quite generic, offering opportunities for extensions of popular single agent algorithms, *e.g.*, TRPO [18], PPO [19]. And also combine them with multi-agent belief systems.

ACKNOWLEDGMENTS

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) through the projects with references UIDB/50021/2020 (DOI: 10.54499/UIDB/50021/2020), PTDC/CCI-COM/5060/2021, and the Center for Responsible AI (ref. n. C628696807-00454142). Guilherme S. Varela is supported by FCT scholarship 2021.05435.BD

REFERENCES

[1] Justin A. Boyan and Michael L. Littman. 1993. Packet routing in dynamically changing networks: a reinforcement learning approach. In *Proceedings of the 6th International Conference on Neural Information Processing Systems* (Denver, Colorado) *(NIPS'93)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 671–678.

[2] Lucian Busoniu, Robert Babuska, and Bart De Schutter. 2010. *Multi-agent Reinforcement Learning: An Overview*. Vol. 310. Sp, 183–221.

[3] Dingyang Chen, Yile Li, and Qi Zhang. 2022. Communication-Efficient Actor-Critic Methods for Homogeneous Markov Games. In *International Conference on Learning Representations*.

[4] Cédric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. 2019. A Hitchhiker's Guide to Statistical Comparisons of Reinforcement Learning Algorithms. (04 2019).

[5] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. 2017. OpenAI Baselines.

[6] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual Multi-Agent Policy Gradients. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence (AAAI'18/IAAI'18/EAAI'18)*. AAAI Press, 9.

[7] Sven Gronauer and Klaus Diepold. 2022. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review* 55, 2 (2022), 895–943.

[8] Jayesh K. Gupta, Maxim Egorov, and Mykel Kochenderfer. 2017. Cooperative Multi-agent Control Using Deep Reinforcement Learning. In *Autonomous Agents and Multiagent Systems*. Springer International Publishing, 66–83.

[9] Pablo Hernandez-Leal, Bilal Kartal, and Matthew E. Taylor. 2019. A Survey and Critique of Multiagent Deep Reinforcement Learning. *Autonomous Agents and Multi-Agent Systems* 33 (2019), 750–797.

[10] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*. https://openreview.net/forum?id=SJU4ayYgl

[11] Iou-Jen Liu, Raymond A. Yeh, and Alexander G. Schwing. 2020. PIC: Permutation Invariant Critic for Multi-Agent Deep Reinforcement Learning. In *Proceedings of the Conference on Robot Learning (Proceedings of Machine Learning Research, Vol. 100)*. PMLR, 590–602.

[12] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. *Neural Information Processing Systems (NIPS)* (2017).

[13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2015. Playing Atari with Deep Reinforcement Learning. *Nature* 518, 7540 (2015), 529–533.

[14] Frans A. Oliehoek and Christopher Amato. 2016. *A Concise Introduction to Decentralized POMDPs*. Springer International Publishing.

[15] Afshin Oroojlooy and Davood Hajinezhad. 2022. A Review of Cooperative Multi-Agent Deep Reinforcement Learning. *Applied Intelligence* 53, 11 (2022), 13677–13722.

[16] Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano Albrecht. 2021. Benchmarking Multi-Agent Deep Reinforcement Learning Algorithms in Cooperative Tasks. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, J. Vanschoren and S. Yeung (Eds.), Vol. 1.

[17] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning*, Vol. 80. PMLR, 4295–4304.

[18] John Schulman, Sergey Levine, Philipp Moritz, Michael Jordan, and Pieter Abbeel. 2015. Trust Region Policy Optimization. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37 (ICML'15)*. JMLR.org, 1889–1897.

[19] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *CoRR* abs/1707.06347 (2017).

[20] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. 2018. Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '18)*. International Foundation for Autonomous Agents and Multiagent Systems, 2085–2087.

[21] R. Sutton, David A. McAllester, Satinder Singh, and Y. Mansour. 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *NIPS*.

[22] Hao Tu, Yuhua Du, Hui Yu, Xiaonan Lu, and Srdjan Lukic. 2024. Privacy-Preserving Robust Consensus for Distributed Microgrid Control Applications. *IEEE Transactions on Industrial Electronics* 71, 4 (2024), 3684–3697. https://doi.org/10.1109/TIE.2023.3274846

[23] Guilherme S. Varela, Alberto Sardinha, and Francisco S. Melo. 2025. Networked Agents in the Dark: Team Value Learning under Partial Observability. arXiv:2501.08778 [cs.LG] https://arxiv.org/abs/2501.08778

[24] Lin Xiao, Stephen Boyd, and Seung-Jean Kim. 2007. Distributed average consensus with least-mean-square deviation. *J. Parallel and Distrib. Comput.* 67, 1 (2007), 33–46.

[25] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. 2021. *Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms*. Springer International Publishing, Cham, 321–384.

[26] Kaiqing Zhang, Zhuoran Yang, Han Liu, Tong Zhang, and Tamer Basar. 2018. Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*. PMLR, 5872–5881.

[27] Yan Zhang and Michael M. Zavlanos. 2019. Distributed off-Policy Actor-Critic Reinforcement Learning with Policy Consensus. In *2019 IEEE 58th Conference on Decision and Control (CDC)*. 4674–4679.