Decentralized actor decentralized critic for multi-agent cooperative environments

Anonymous Author(s)
Affiliation
Address
email

Abstract

We present fully decentralized multi-agent actor-critic reinforcement learning framework for cooperative environments where both the actor and the critic are decentralized. To ensure consistency between the agents, each critic learns the common-information based value (or advantage) functions and each actor learns a coordination policy which maps the common information to functions from local information to actions. Both the critic and the actor compress the common-information using a Wasserstein or KL-divergence regularized RNN. We prove that, for partial history sharing information structure, the proposed framework is guaranteed to converge to an approximately locally optimal cooperative multi-agent planning solution. To the best of our knowledge, this is the first provably convergent multi-agent actor-critic framework with decentralized critic with optimality guarantees. We illustrate this approach on multiple multi-agent environments.

1 Introduction

In recent years, reinforcement learning (RL) has achieved considerable success and is able to achieve super-human performance in many games such as Atari [33] and Go [43]. Most of this success is limited to single-agent RL. In contrast, multi-agent reinforcement learning (MARL) remains challenging in spite of the considerable recent progress [13, 14, 19, 27, 36, 51].

In this paper, we highlight one aspect of MARL which, in our opinion, hasn't received much attention in the literature. **How can multiple agents, which are operating in an unknown environment, learn to act optimally when *a system simulator is not available*?** In such a setup, agents observe data according to the information structure of the system and need to learn policies that are *optimal* for that information structure. There are various results which show that most single agent reinforcement learning algorithms have this property [7, 44, 46] but most algorithms for cooperative MARL do not possess this property for one of two reasons: (i) It is assumed that there is a centralized critic which evaluates the performance of the decentralized actors [13, 29]; a centralized critic cannot exist in online decentralized multi-agent environments. (ii) It is assumed that agents can communicate with their neighbors and use this inter-agent communication to learn the centralized critic [21, 51] *for the system where no inter-agent communication is present*; when agents have the ability to communicate to their neighbors, they must decide what to communicate to their neighbors and how to react to received messages in order to act *optimally*.

Main contribution: Our main contribution is to develop a MARL algorithm with a decentralized actor and a decentralized critic (DADC) which operates in a truly decentralized manner. The algorithm is based on the common-information approach for optimal planning for cooperative multi-agent systems [35]. As part of this algorithm, we develop a novel method to construct an approximate information state which is a compression of the common information with guaranteed approximation

Submitted to 33rd Conference on Neural Information Processing Systems (NeurIPS 2019). Do not distribute.

bounds. The proposed DADC algorithm uses this approximate information state and *any* actor-critic framework for single agent RL to construct decentralized actors and decentralized critics. We show that DADC algorithm possesses the same convergence guarantees as single agent actor-critic algorithms. To the best of our knowledge, this is the first MARL algorithm with decentralized actor and decentralized critic that provably converges to locally optimal decentralized policies.

2 Model of a multi-agent cooperative team

A multi-agent team is a tuple $\langle \mathcal{K}, \mathcal{S}, (\mathcal{O}^k)_{k\in\mathcal{K}}, (\mathcal{A}^k_t)_{k\in\mathcal{K}}, P_0, P, T, r\rangle$ where

- $\mathcal{K} = \{1, \dots, K\}$ is the set of agents.
- $\mathcal{S}$ is the state space. $\mathcal{O}^k$, $\mathcal{A}^k$, $k \in \mathcal{K}$, are the observation and action spaces of agent k. Let $\mathcal{O} = \prod_{k\in\mathcal{K}} \mathcal{O}^k$ and $\mathcal{A} = \prod_{k\in\mathcal{K}} \mathcal{A}^k$. We use $S_t \in \mathcal{S}$, $O_t := (O^k_t)_{k\in\mathcal{K}} \in \mathcal{O}$, and $A_t := (A^k_t)_{k\in\mathcal{K}} \in \mathcal{A}$, to denote the system state, observations, and actions at time t.
- $P_0 \in \Delta(\mathcal{S})$ is the initial distribution of the initial state S_0.
- $P\colon \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ denotes the transition probability of the system, i.e.,
$$\begin{aligned}\mathbb{P}(S_{t+1} = s_{t+1} \mid S_{0:t} = s_{0:t}, A_{0:t} = a_{0:t}) &= \mathbb{P}(S_{t+1} = s_{t+1} \mid S_t = s_t, A_t = a_t)\\ &= P(s_{t+1}|s_t, a_t).\end{aligned}$$
- $T\colon \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{O})$ denotes the observation probability of the system, i..e,
$$\begin{aligned}\mathbb{P}(O_t = o_t \mid S_{0:t} = s_{0:t}, A_{0:t-1} = a_{0:t-1}) &= \mathbb{P}(O_t = o_t \mid S_t = s_t, A_t - 1 = a_t - 1)\\ &= T(o_t|s_t, a_{t-1}).\end{aligned}$$
- $r\colon \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ denotes the per-step reward function. The team receives a reward $R_t = r(S_t, A_t, S_{t+1})$ at time t.

Information structure: A critical feature of a multi-agent system is the *information structure* which captures the knowledge of who knows what about the system and when. We use I^k_t to denote the information known to agent k at time t. In general, I^k_t is a subset of the total information $(O_{0:t}, A_{0:t-1}, R_{0:t-1})$ known to all agents in the system. Examples of information structures commonly used in the literature are presented below. We use $\mathcal{I}^k_t$ to denote the space of the information available to agent k at time t. Note that, in general, the information available to agent k increases with time. So, $\mathcal{I}^k_t$ are sets that are increasing with time.

Policy: The policy of agent k is a collection $\pi^k = (\pi^k_0, \pi^k_1, \dots)$, where $\pi^k_t\colon \mathcal{I}^k_t \to \Delta(\mathcal{A}^i)$. Note that since the information sets are increasing with time, in general, the policy π^k is time-varying. We use $\pi = (\pi^k)_{k\in\mathcal{K}}$ to denote the policy for all agents. The performance of a policy π is given by

$$J(\pi) = \mathbb{E}^\pi\left[\sum_{t=0}^{\infty} \gamma^t R_t\right], \tag{1}$$

where $\gamma \in (0, 1)$ is the discount factor. The objective is to find a (possible time-varying) policy π that maximizes the performance $J(\pi)$ defined in (1).

Examples of information structures: Some examples of the information structures are as follows.

- **Delayed sharing:** $I^k_t = \{O_{0:t-d}, A_{0:t-d}, O^k_{t-d+1:t}, A^k_{t-d+1:t-1}\}$. This models systems where agents broadcast their information and communication has delay of d. Planning for models where $d = 1$ has been considered in [40, 50] and for general d has been considered in [34].
- **Periodic sharing:** $I^k_t = \{O_{0:t-\tau}, A_{0:t-\tau}, O^k_{t-\tau+1:t}, A^k_{t-\tau+1:t-1}\}$, where $\tau = p\lfloor \frac{t}{p} \rfloor$. This models systems where agents periodically broadcast their information every p steps. Planning for this model has been considered in [37].
- **Control sharing:** $I^k_t = \{O^k_{1:t}, A_{1:t-1}\}$. This models systems where control actions are observed by everyone (which is the case for certain communication and economic applications). Planning for variations of this model has been considered in [8, 30, 40].
- **Mean-field sharing:** $I^k_t = \{S^k_{1:t}, A^k_{1:t-1}, M_{1:t}\}$, where the state S_t is $(S^1_t, \dots, S^K_t)$, the observation of agent k is S^k_t, and $M_t = \big(\sum_{k\in\mathcal{K}} \delta_{S^k_t}\big)/K$ denotes the empirical distribution of the states. This models systems where mean-field is observed by all agents (which is the case for smart grid and other large-scale systems). Planning for variations of this model has been considered in [1, 2].

3 Common information based approach for planning

In general, finding the optimal plan for multi-agent teams is NEXP-complete [6]. However, it is shown in [35] that when the information structure is of a particular form (known as partial history sharing), it is possible to reduce the multi-agent planning problem to a single agent planning problem from the point of view of a virtual agent called the coordinator. We summarize this approach below.

Common and local information: Define

$$C_t = \bigcap_{s \geq t} \bigcap_{k \in \mathcal{K}} I_s^k \qquad \text{and} \qquad L_t^k = I_t^k \setminus C_t, \quad k \in \mathcal{K}.$$

C_t denotes the *common information*, i.e., the information that is common to all agents all the time in the future and L_t^k denotes the *local information* at agent k. By construction, $I_t^k = \{C_t, L_t^k\}$. Let $\mathcal{C}_t$ and $\mathcal{L}_t^k$ denote the space of realizations of C_t and L_t^k and let $L_t = (L_t^k)_{k \in \mathcal{K}}$ and $\mathcal{L} = \prod_{k \in \mathcal{K}} \mathcal{L}^k$. By construction, $C_t \subseteq C_{t+1}$. Let $Z_t = C_{t+1} \setminus C_t$ denote the new common information at time t. Then, C_t may be written as $Z_{0:t}$.

Sufficient condition for dynamic programming decomposition: The information structure is called *partial history sharing* [31, 35] if the local information satisfies the following two properties:

(L1) For any agent k, any subset $\mathcal{B}$ of L_{t+1}^k, and any realization c_t of C_t, ℓ_t^k of L_t^k, a_t^k of A_t^k, and o_{t+1}^k of O_{t+1}^k, we have

$$\begin{aligned}\mathbb{P}(L_{t+1}^k \in \mathcal{B} \mid C_t = c_t, L_t^k = \ell_t^k, A_t^k = a_t^k, O_{t+1}^k = o_{t+1}^k) \\ = \mathbb{P}(L_{t+1}^k \in \mathcal{B} \mid L_t^k = \ell_t^k, A_t^k = a_t^k, O_{t+1}^k = o_{t+1}^k),\end{aligned}$$

i.e., conditioned on the current local information, current action, and future observation of agent k, the future realization of the local information does not depend on the current realization of the common information.

(L2) The size of the local information is uniformly bounded. Thus, without loss of generality, we may assume that $\mathcal{L}_t^k$ does not depend on t and simply denote it by $\mathcal{L}^k$.

Among the information structures illustrated in Sec. 2, delay and periodic sharing information structures satisfy conditions (L1) and (L2). Control sharing and mean-field sharing information structures don't immediately satisfy condition (L1). However, for specific types of dynamics and observations, these information structures can be simplified by shedding irrelevant information (see [2, 30, 32]) and the simplified information structures satisfy (L1) and (L2).

Prescriptions: Given a policy $\pi = (\pi^k)_{k \in \mathcal{K}}$ and a realized trajectory $(c_0, c_1, \dots)$ of the common information, the prescription h_t^k is the partial application of c_t to π_t^k, i.e., $h_t^k = \pi_t^k(c_t, \cdot)$, $k \in \mathcal{K}$. Note that h_t^k is a function from $\mathcal{L}_t^k$ to $\Delta(\mathcal{A}_t^k)$. Let h_t denote $(h_t^k)_{k \in \mathcal{K}}$.

A virtual coordinated system: Any model that satisfies conditions (L1) and (L2) may be viewed as virtual single agent planning problem known as the coordinated system. The environment of the virtual coordinated system consists of two components: the first component is the same as the environment of the original multi-agent system which evolves according to dynamics P; the second component consists of K *passive agents*, whose operation we will describe later. There is a virtual coordinator who observes the common information C_t and chooses *prescriptions* $H_t = (H_t^k)_{k \in \mathcal{K}}$, where $H_t^k \colon \mathcal{L}^k \to \Delta(\mathcal{A}^k)$ using a *coordination rule* ψ_t, i.e., $H_t = \psi_t(C_t)$. Each passive agent $k \in \mathcal{K}$ uses the component H_t^k of the prescription H_t to sample an action $A_t^k \sim H_t^k(L_t^k)$. Then, the virtual coordinated system is equivalent to the original multi-agent system in the following sense.

Theorem 1 ([35]) *Let $\pi = (\pi^k)_{k \in \mathcal{K}}$ be an optimal policy of the original multi-agent system. Then, the coordination policy $\psi = (\psi_0, \psi_1, \dots)$ given by $\psi_t(c_t) = \big(\pi_t^k(c_t, \cdot)\big)_{k \in \mathcal{K}}$ is optimal for the virtual coordinated system. Conversely, let $\psi = (\psi_0, \psi_1, \dots)$ be an optimal coordination policy for the virtual coordinated system. Then, the multi-agent policy $\pi = (\pi^k)_{k \in \mathcal{K}}$ given by $\pi_t^k(c_t, \ell_t^k) = \psi_t^k(c_t)(\ell_t^k)$, where $\psi_t^k(c_t)$ denotes the k-th component of $\psi_t(c_t)$, is optimal for the original multi-agent system.*

Dynamic program: The virtual coordinated system has a single decision maker and the information available to the decision maker is increasing with time. Thus, the optimal coordination policy may be obtained using standard dynamic program for single-agent partially observed systems as follows.

Theorem 2 ([35]) *Let $b_t \in \Delta(\mathcal{S} \times \mathcal{L})$ denote the belief on (S_t, L_t) given the past realizations of the common information and prescriptions, i.e., $b_t(s, \ell) = \mathbb{P}(S_t = s, L_t = \ell \mid C_t = c_t, H_{0:t} = h_{0:t})$. Then, b_{t+1} may be written as $\Phi(b_t, h_t, z_t)$, where the update function Φ does not depend on the coordination policy ψ. Furthermore, let $V \colon \Delta(\mathcal{S}) \to \mathbb{R}$ denote the fixed point of the following DP:*

$$V(b) = \max_{h=(h^k)_{k\in\mathcal{K}}} \mathbb{E}[r(S_t, A_t, S_{t+1}) + \gamma V(\Phi(b, h, Z_t)) \mid B_t = b, H_t = h]. \tag{2}$$

Let $\psi^*(b)$ denote any arg max of the right hand side of (2)*. Then, the time-homogeneous coordination policy $(\psi^*, \psi^*, \dots)$ is optimal for the virtual coordinated system. The policy $\pi = (\pi^k)_{k\in\mathcal{K}}$ for the multi-agent system obtained using the construction in Thm. 1 is optimal for the multi-agent system.*

The above approach is called the *common information approach* [35] and has been used to obtain planning solutions for delayed sharing [34], control sharing [30], and mean-field sharing [1, 2].

Two interpretations of the common information approach: There are two ways to interpret the common information approach to obtain optimal multi-agent strategies. The first interpretation is to assume that all agents get together before the system starts running, solve the dynamic program (2), and agent k stores the component $\psi^{*,k}$ of the optimal coordination policy. The second interpretation does not require the agents to be physically get together. The dynamic program (2) is based on common information, which is known at all agents. So, another interpretation is that before the system starts running each agent independently solves the dynamic program (2) and implements its component $\psi^{*,k}$ of the optimal coordination policy. The only caveat with the second interpretation is that if there are multiple actions that achieve the arg max of the right hand side of (2), then all agents must consistently choose the arg max. Such a consistent choice can be guaranteed if the agents agree upon a rule to break ties (e.g., sort all arg max in a lexicographical order and choose the smallest or the largest alternative). We will use the second interpretation to develop a decentralized RL algorithm.

4 Approximate information state (AIS) for the common information

The coordinated system is a partially observed system. The belief state formulation used in the dynamic program (2) works for planning but cannot be used for RL because the construction of the belief state depends on the model, which is unknown. There are some results that circumvent these difficulties in a principled manner [5, 18, 28] but many of the results suggest using a RNN or LSTM for compressing the history of observations [3, 4, 16, 17, 48, 49]. However, such constructions do not provide any optimality guarantees. In this section, we present an online data-driven approach to construct an information state which may be viewed as a Wasserstein or KL-divergence regularized RNN and provides an ε-approximation performance guarantee.

Let $\mathcal{F}_t = \sigma(C_t)$ denote the filtration generated by the common information.

Definition 1 A (ε, δ)-approximate information state (AIS) $\{X_t\}_{t\geq 0}$, $\mathcal{F}_t$ adapted processes (i.e.,, there exist functions $\{\rho_t\}_{t\geq 0}$ s.t. $X_t = \rho_t(C_t)$) belonging to a Polish space $(\mathcal{X}, d)$ which satisfies:

(P1) **Sufficient for approximate performance evaluation:** i.e.,

$$\big|\mathbb{E}[R_t \mid C_t = c_t, H_t = h_t] - \mathbb{E}[R_t \mid X_t = \rho_t(c_t), H_t = h_t]\big| \leq \varepsilon.$$

(P2) **Sufficient to predict itself approximately:** For any Borel subset B of $\mathcal{X}$, define $\mu_t(B) = \mathbb{P}(X_{t+1} \in A \mid C_t = c_t, H_t = h_t)$ and $\nu_t(B) = \mathbb{P}(X_{t+1} \in A \mid X_t = \rho_t(c_t), H_t = h_t)$. Then, $\mathcal{W}(\mu_t, \nu_t) \leq \delta$, where $\mathcal{W}(\cdot, \cdot)$ denotes the Wasserstein distance[1] between two distributions.

(P3) **Time-homogeneity:** The expectation $\mathbb{E}[R_t \mid X_t = x, H_t = h]$ and the transition kernel $\mathbb{P}(X_{t+1} \in \cdot \mid X_t = x, H_t = h)$ are time-homogeneous.

Theorem 3 *Let $\{X_t\}_{t\geq 0}$ be an (ε, δ)-AIS and let $\hat{V}$ be the fixed point of the dynamic program:*

$$\hat{V}(z) = \max_{h=(h^k)_{k\in\mathcal{K}}} \mathbb{E}[r(S_t, A_t, S_{t+1}) + \gamma \hat{V}(X_+) \mid X = x, H = h]. \tag{3}$$

[1] Let $(\mathcal{X}, d)$ be a Polish metric space. For any two probability measures μ, ν on $\mathcal{X}$, the Wasserstein distance $\mathcal{W}(\mu, \nu)$ is $\inf_{\pi\in\Pi(\mu,\nu)} \int_{\mathcal{X}} d(x, y) \, \mathrm{d}\pi(x, y)$ where Π represents the product space of the two distributions.

Suppose $\hat{V}$ is Lipschitz continuous with Lipschitz constant L_V. Then for any time t and any realization of c_t of the common information and b_t of the belief, we have

$$\big|V(b_t) - \hat{V}(\rho_t(c_t))\big| \le \big(\varepsilon + \gamma L_V \delta\big)/(1-\gamma).$$

See supplementary material for proof.

Wasserstein distance vs other metrics: It is possible to replace the Wasserstein distance in (P2) by the total variation distance[2] and assume that $\|\mu_t - \nu_t\|_{\mathrm{TV}} \le \delta$. There are two ways to bound the approximation error. (i) If $\mathcal{X}$ is a bounded metric space with diameter D, then [47, Case 6.16] shows that $\mathcal{W}(\mu,\nu) \le D\|\mu-\nu\|_{\mathrm{TV}}$. Thus, the result of Theorem 3 holds with δ replaced by $D\delta$. (ii) If $\hat{V}$ is uniformly bounded with sup-norm L_∞, then the result holds with L_V replaced by L_∞. See supplementary material for details.

It is also possible to replace Wasserstein distance in (P2) by KL-divergence instead and assume that $\mathrm{KL}(\mu_t,\nu_t) \le \delta$. Pinsker's inequality [12] implies that $\|\mu_t - \nu_t\|_{\mathrm{TV}} \le \sqrt{\mathrm{KL}(\mu_t,\nu_t)/2} \le \sqrt{\delta/2}$ and then the previous bounds using total variation hold.

5 Common information based approach for multi-agent learning

The high-level idea for decentralized RL: Our main contribution is to build on the second interpretation of the common information approach described above to develop a decentralized RL algorithm. In particular, suppose agents pick any standard *single-agent* actor-critic RL algorithm (such as TRPO [41], PPO [42], or NAFDQN [15]), each agent then uses this actor-critic algorithm to keep keeps track of value function V and the policy ψ^* (which is time-homogeneous), sample the prescription $h_t = (h_t^k)_{k\in\mathcal{K}}$ according to the policy ψ^*, and then update V and ψ^* using the common observation Z_t and the common reward R_t. As in planning setup, one caveat with using this approach is to ensure that all agents pick the sample for the prescription h_t. One way to guarantee identical samples is to use the same random number generator and the same seed at all agents.[3]

Parameterized prescriptions: The prescription H_t^k takes values in the space of functions from $\mathcal{L}^k \to \Delta(\mathcal{A}^k)$. We model this space of functions using a family of parameterized function $\mathcal{H}_{\Theta^k} = \{h_\theta\}_{\theta\in\Theta^k}$ such as Gibbs/Boltzmann functions or neural networks. So, we consider that the virtual coordinator generates the prescription parameters $(\theta^k)_{k\in\mathcal{K}}$ for all agents and then agent k uses the parameterized prescription h_{θ^k}. Thus, the coordinator's "action" $\theta = (\theta_k)_{k\in\mathcal{K}}$ is a continuous variable that lies in the closed convex set $\Theta = \prod_{k\in\mathcal{K}} \Theta^k$.

Reinforcement learning framework: We propose the following multi-agent decentralized-actor decentralized-critic (DADC) RL architecture shown in Fig. 1(a). Each agent runs a separate instance of DADC and has three components as shown in Fig. 1(b).

1. **State approximator**, which consists of two parts: (i) an AIS encoder which is an RNN/LSTM with the common observation Z_t and the past prescription parameters θ_{t-1} as inputs and AIS X_t as the output; and (ii) a feedforward NN with AIS X_t and current prescription parameters θ_t as inputs and estimated reward $\hat{R}_t$ and distribution ν_{t+1} of next AIS as outputs. We use the smooth L1 loss between the predicted $\hat{R}_t$ and the observed reward R_t to minimize the ε parameter of (P1) and use the negative log likelihood loss[4] of ν_t to minimize the parameter δ of (P2). Thus, the overall loss function for the state approximator is $\mathcal{L}_{\mathrm{AIS}} = \lambda\mathcal{L}_R + (1-\lambda)\mathcal{L}_\nu$, where $\lambda \in (0,1)$ is a hyper-parameter and, given a batch size of B,

$$\mathcal{L}_R = \frac{1}{B}\sum_{t=0}^{B-1} \texttt{smoothL1}(\hat{R}_t - R_t) \quad \text{and} \quad \mathcal{L}_\nu = -\sum_{t=0}^{B-2} \log(\nu_{t+1}(X_{t+1})).$$

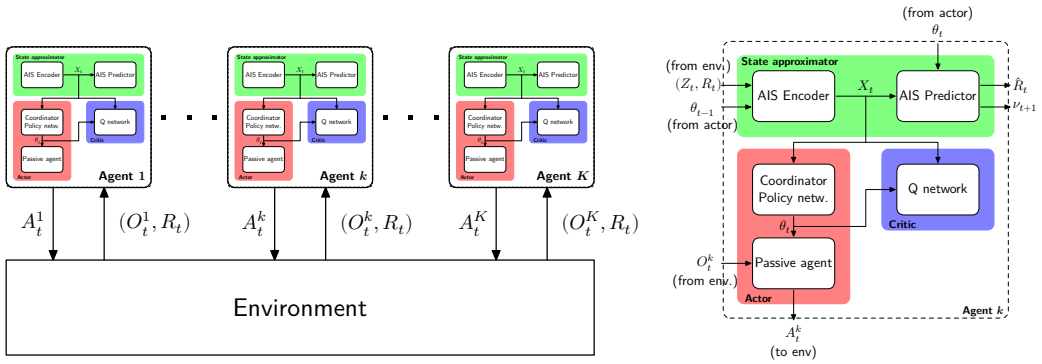
   Let ξ denote the combined parameters of the AIS encoder and predictor, which are updated using stochastic gradient ascent

$$\xi_{k+1} = \xi_k + a_k \nabla_\xi \mathcal{L}_{\mathrm{AIS}}(\xi_k), \tag{4}$$

[2] The total variation distance $\|\mu-\nu\|_{\mathrm{TV}}$ is $\sup_{f:\|f\|_\infty\le 1}\big|\int f(x)\mu(dx) - \int f(x)\nu(dx)\big|$.

[3] Each passive agent samples only its own action $A_t^k \sim \psi^{*,k}(B_t)$ which can be done independently.

[4] The negative log likelihood loss approximates KL-divergence.



(a) The multi-agent RL environment

(b) DADC at an agent

Figure 1: The multi-agent decentralized-actor decentralized-critic (DADC) RL architecture.

where the learning rate $\{a_k\}_{k\geq 0}$ satisfies the standard conditions $\sum a_k = \infty$ and $\sum a_k^2 < \infty$.

2. **Critic for virtual coordinator,** which is a feedforward neural network with the AIS X_t and prescription parameters θ_t as input and $Q(X_t, \theta_t)$ as output. Let φ denote the parameters of this network, which are updated using batch temporal-difference (TD):

$$\varphi_{k+1} = \varphi_k + b_k \nabla_\varphi \mathcal{L}_{\text{TD}}(\xi_k, \varphi_k, \vartheta_k), \tag{5}$$

where $\mathcal{L}_{\text{TD}}(\xi_k, \varphi_k, \vartheta_k) \coloneqq \dfrac{1}{B} \displaystyle\sum_{t=0}^{B-1} \texttt{smoothL1}(Q_{\xi_k,\varphi_k,\vartheta_k}(X_t, \theta_t),\ R_t + \gamma Q_{\xi_k,\varphi_k,\vartheta_k}(X_{t+1}, \theta_{t+1}))$

and ϑ_k are the parameters of the coordination policy at the actor (explained below) and the learning rate $\{b_k\}_{k\geq 0}$ satisfies the standard conditions $\sum b_k = \infty$ and $\sum b_k^2 < \infty$. In addition, we require, $\lim_{k\to\infty} b_k/a_k = 0$ to ensure that the state approximator converges faster. Instead of the TD(1) update in (5), we can also use TD(λ) update with eligibility traces to reduce variance [44].

3. **Actor,** which has two parts: (i) a coordinator policy network with network with AIS X_t as input and a distribution on prescription parameters θ_t as output; (ii) a passive agent which samples the action A_t^k for the k-th agent using the parameterized prescription $h_{\theta_t^k}$. Let ϑ denote the parameters of the coordinator policy network. Then, we use ψ_ϑ to denote the parameterized coordination rule. The parameter ϑ is updated using the policy gradient theorem [23, 45]:

$$\vartheta_{k+1} = \vartheta_k + c_k \nabla_\vartheta J(\xi_k, \varphi_k, \vartheta_k), \tag{6}$$

where the policy gradient is given by

$$\nabla_\vartheta J(\xi_k, \varphi_k, \vartheta_k) = \frac{1}{B} \sum_{t=0}^{B-1} Q_{\xi_k,\varphi_k,\vartheta_k}(X_t, \theta_t) \nabla_\vartheta \log \psi_{\vartheta_k}(X_t, \theta_t)$$

and the learning rate $\{c_k\}_{k\geq 0}$ satisfies $\sum c_k = \infty$ and $\sum c_k^2 < \infty$. In addition we require $\lim_{k\to\infty} c_k/b_k = 0$ to ensure that the critic learns faster.

The choice of the learning rates implies that there is a separation of timescales between the updates at the state approximator, the critic, and the actor. Thus, we can show convergence of the algorithm using ideas from [10]. We impose the following mild technical assumptions:

Assumption 1

1. All network parameters $(\xi_k, \varphi_k, \vartheta_k)$ lie in convex and bounded subsets of Euclidean spaces.
2. The gradient of the loss function $\nabla_\xi \mathcal{L}_{\text{AIS}}(\xi_k)$ of the state approximator is Lipschitz in ξ_k, the gradient of the TD loss $\nabla_\varphi \mathcal{L}_{\text{TD}}(\xi_k, \varphi_k, \vartheta_k)$ and the policy gradient $\nabla_{\vartheta_k} J(\xi_k, \varphi_k, \vartheta_k)$ are Lipschitz in $(\xi_k, \varphi_k, \vartheta_k)$.
3. All the gradients—$\nabla_\xi \mathcal{L}_{\text{AIS}}(\xi_k)$ at the state approximator; $\nabla_\varphi \mathcal{L}_{\text{TD}}(\xi_k, \varphi_k, \vartheta_k)$ at the critic; and $\nabla_{\vartheta_k} J(\xi_k, \varphi_k, \vartheta_k)$ at the actor—are unbiased with bounded variance.

The proposed RL framework has the following convergence guarantees.

Theorem 4 *Suppose in addition to Assumption 1 the following regularity conditions hold: the ODE corresponding to* (6) *is locally asymptotically stable and the ODEs corresponding to* (4) *and* (5) *are globally asymptotically stable with the ODE corresponding to* (5) *having a fixed point which is Lipschitz continuous in ϑ; . Then, along any sample path, almost surely we have:* (a) *iteration* (4) *converges to a state estimator that minimizes the loss function $\mathcal{L}_{AIS}$;* (b) *iteration* (5) *converges to a critic that minimizes the error with respect to the true Q-function; and* (c) *iteration* (6) *converges to a local maximum of the performance $J(\xi^*, \varphi^*, \vartheta)$.*

PROOF The conditions stated in the theorem and the choice of learning rates satisfy all the conditions stated in [26, page 35], [10, Theorem 23]. The result then follows from the application of the theorem given in [26, page 35], [10, Theorem 23]. See supplementary material for detailed proof.

The converge guarantees of Theorem 4 provides the following approximation bounds.

Theorem 5 *At convergence, let ε and δ be the error constants in* (P1) *and* (P2)*, and let $\kappa = \|V_{\xi,\varphi,\vartheta} - \hat{V}\|_\infty$ where $V_{\xi,\varphi,\vartheta}$ is the converged value function at the critic and $\hat{V}$ is the solution of* (3)*. Then, by the triangle inequality, we have that for any time t, and any realization c_t of the common information, and realization b_t of the belief,*

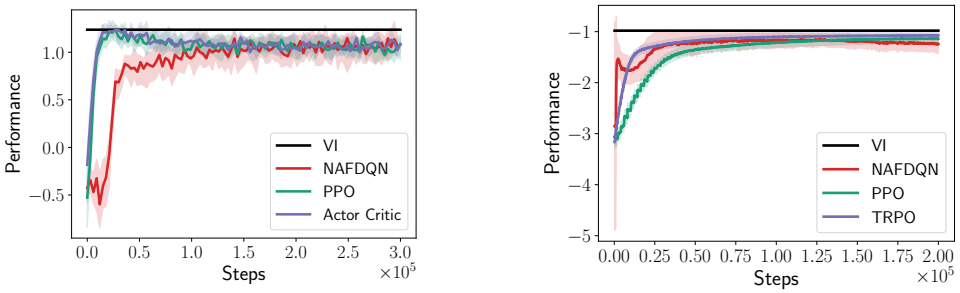
$$\left|V(b_t) - V_{\xi,\varphi,\vartheta}(\rho_t(c_t))\right| \leq \kappa + \frac{\varepsilon + L_V \delta}{1-\gamma}.$$

6 Numerical experiments

We test the convergence of DADC on two multi-agent environments for which the optimal planning solution is known. We don't explicitly compare with other MARL algorithm because they don't have a decentralized critic. We use the following environments (details in the supplementary material):

1. **Multi-access broadcast [31],** which is a communication model with two transmitters where packets arrive according to a Bernoulli process; the transmitters transmit over a Markov ON-OFF channel. Each transmitter has two options: transmit or do not transmit. Transmission is successful only if the channel is ON and only one transmitter transmits. Each transmitter observes the number of packets in its buffer. If either of the transmitter transmits, then both transmitters observe the state of the channel and the control action of both transmitters. The objective is to maximize the discounted successful transmissions.
2. **Demand response in smart-grids [1],** which is a smart-grid environment with 100 ON-OFF devices. Each load observes the number of nodes which are in the ON-state and has the option to forcibly turn on, forcibly turn off, or follow its natural (Markov) dynamics. Forcibly turning the device incurs an inconvenience cost to the user. The objective is to minimize the distance of the empirical distribution from a desired target distribution and the inconvenience cost incurred.

For both environments, we run DADC with different choices of the actor critic framework. The details of the network architecture and the hyperparameters are provided in the supplementary material. The results, shown in Fig. 2, illustrate that DADC is robust to the choice of the actor-critic framework and converges to the optimal decentralized policies obtained by value iteration.



(a) Multi-access broadcast

(b) Demand response in smart grids

Figure 2: Performance of DADC for different choices of actor-critic framework. The solid lines show the median and the shaded areas show the region between the first and third quartile across 25 runs.

7 Discussion

Perceived non-stationarity of the environment: A naive approach for MARL is for all agents to simply use a standard single agent RL algorithm. Such a naive approach does not work because the environment for a single agent appears to be non-stationary due to the presence of other agents. Such a perception of non-stationarity does not appear in the common information approach because virtual coordinated environment is a single agent environment.

Sufficient statistics for the history of observation: A challenge in MARL is to identify sufficient statistics for the history of observations at each agent. In single agent partially observed environments, the belief of the current state is such a sufficient statistic. However, that is not the case for multi-agent environments. As an example, consider a two agent environment. If agent 1 chooses its action based on its belief on the state of the system, then agent 2 must form a belief on the belief of agent 1. Then, agent 1 must form a belief on the belief of agent 2 on the belief of agent 1. This belief on belief process continues ad infinitum. Some of the existing literature simply uses an RNN/LSTM to learn sufficient statistics on the fly but such a construction isn't guaranteed to learn a statistic that has any approximation guarantees. In contrast, we take a more principled approach. We restrict attention to partial history sharing models where the belief on the current state and the local information given the common information is a sufficient statistic for the history of common information. We then construct a data-driven sufficient statistic for this belief using the notion of AIS, which provides approximation guarantees.

Alternative characterization of AIS: Property (P2) in the definition of AIS can be replaced by the following stronger conditions: (P2a) **Evolves in a state-like manner,** i.e., there exist Lipschitz functions $\{\hat{\rho}_t\}_{t\geq 0}$ with Lipschitz constant L_ρ such that $X_{t+1} = \hat{\rho}_t(X_t, O_{t+1}, A_t)$. (P2b) **Is sufficient for predicting future observations approximately.** i.e., for any Borel set B of $\mathcal{O}$, define $\hat{\mu}_t(A) = \mathbb{P}(O_{t+1} \in B \mid C_t = c_t, H_t = h_t)$ and $\hat{\nu}_t(A) = \mathbb{P}(O_{t+1} \in B \mid X_t = \rho_t(c_t), H_t = h_t)$. Then, $\mathcal{W}(\hat{\mu}_t, \hat{\nu}_t) \leq \delta$. It can be shown that the result of Theorem 3 holds with L_V replaced by $L_\rho L_V$.

A state approximator using properties (P1), (P2a), (P2b), may be thought of as an RNN/LSTM that predicts the reward R_t and a distribution on next observation O_{t+1}, where the loss function is a weighted combination of L1 loss in predicting the rewards and the KL-divergence between the predicted and the actual distribution of observations.

Remarks on the restrictiveness of the model: The results presented in this paper only work for models which satisfy (L1) and (L2). This assumption does not hold for some of the MARL benchmark domains such as StarCraft but it does hold for many engineering applications where parts of the observations or actions are shared naturally. For example, in MABC environment, it is not unreasonable to assume that agents can observe whether the other agent has transmitted; in the smart grid environment, it is not unreasonable to assume that agents can sense the total consumption based on voltage and frequency droop; and so on.

Improving the scalability of the algorithm: In this paper, we presented the basic framework of a decentralized actor and decentralized critic algorithm for MARL. The scalability of the approach can be improved further by using factored representations [14] and counterfactual reasoning [13].

Related work: Perhaps the closest approach to our work is [14], which uses the common information approach and constructs a public belief MDP for cooperative partially observable multi-agent system. The public belief MDP in [14] is the same as the virtual coordinated system in [35] and the public belief in [14] is same as the common information based belief in [35] (or the belief b_t in Theorem 2). In [14], it is assumed that the model has a specific factored representation which is exploited to update the public belief. In contrast, we present a data-driven approach to construct an AIS from observed samples.

Another related class of algorithms are [21, 51] which present fully decentralized multi-agent actor critic algorithms which use inter-agent communication to build a critic. In these algorithms, agents are allowed to communicate during learning but not during the execution. In our opinion, if agents have the ability to communicate, then they should try to achieve the optimal performance when communication is used during execution as well.

References

[1] Jalal Arabneydi and Aditya Mahajan. Team optimal control of coupled subsystems with mean-field sharing. In *IEEE Conference on Decision and Control*, pages 1669–1674. IEEE, 2014.

[2] Jalal Arabneydi and Aditya Mahajan. Linear quadratic mean field teams: Optimal and approximately optimal decentralized solutions. arXiv:1609.00056v2, 2016.

[3] Andrea Baisero and Christopher Amato. Learning internal state models in partially observable environments;. *Reinforcement Learning under Partial Observability, NeurIPS Workshop*, 2018.

[4] Bram Bakker. Reinforcement learning with long short-term memory. In *NIPS*, 2002.

[5] Jonathan Baxter and Peter L Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

[6] Daniel S. Bernstein, Shlomo Zilberstein, and Neil Immerman. The complexity of decentralized control of markov decision processes. In *UAI*, pages 32–27, Stanford, CA, June 2000.

[7] D.P. Bertsekas and J.N. Tsitsiklis. *Neuro-dynamic Programming*. Anthropological Field Studies. Athena Scientific, 1996.

[8] Jean-Michel Bismut. An example of interaction between information and control: The transparency of a game. *IEEE Trans. Autom. Control*, 18(5):518–522, October 1972.

[9] Vivek Borkar. *Stochastic Approximation: A Dynamical Systems Viewpoint*. Cambridge University Press, 2008.

[10] Vivek S Borkar. Stochastic approximation with two time scales. *Systems & Control Letters*, 29(5):291–294, 1997.

[11] ChainerRL Repository. https://github.com/chainer/chainerrl, 2018.

[12] Imre Csiszàr and Jànos Köner. *Information Theory: Coding Theorems for Discrete Memoryless Systems*. Academic Press, 1986.

[13] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *AAAI*, pages 2974–2982, 2018.

[14] Jakob N. Foerster, Francis Song, Edward Hughes, Neil Burch, Iain Dunning, Shimon Whiteson, Matthew Botvinick, and Michael Bowling. Bayesian action decoder for deep multi-agent reinforcement learning. arXiv:1811.01458, 2018.

[15] Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep Q-learning with model-based acceleration. In *ICML*, 2016.

[16] Matthew Hausknecht and Peter Stone. Deep recurrent Q-learning for partially observable MDPs. In *2015 AAAI Fall Symposium Series*, 2015.

[17] Nicolas Heess, Jonathan J Hunt, Timothy P Lillicrap, and David Silver. Memory-based control with recurrent neural networks. arXiv:1512.04455, 2015.

[18] Ahmed Hefny, Zita Marinho, Wen Sun, Siddhartha Srinivasa, and Geoffrey Gordon. Recurrent predictive state policy networks. arXiv:1803.01489, 2018.

[19] P. Hernandez-Leal, B. Kartal, and M. E. Taylor. Is multiagent deep reinforcement learning the answer or the question? A brief survey. arXiv:1810.05587, 2018.

[20] M. Huang and Y. Ma. Mean field stochastic games: Monotone costs and threshold policies. In *IEEE Conference on Decision and Control*, pages 7105–7110, Dec 2016.

[21] M. Hüttenrauch, A. Šošić, and G. Neumann. Deep RL for Swarm Systems. arXiv:1807.06613, 2018.

[22] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv:1412.6980, 2014.

[23] Vijay R Konda and John N Tsitsiklis. On actor-critic algorithms. *SIAM Journal on Control and Optimization*, 42(4):1143–1166, 2003.

[24] Ilya Kostrikov. https://github.com/ikostrikov/pytorch-ddpg-naf, 2019.

[25] Harold Kushner and G George Yin. *Stochastic approximation and recursive algorithms and applications*, volume 35. Springer Science & Business Media, 2003.

[26] D. S. Leslie. *Reinforcement learning in games*. PhD thesis, The University of Bristol, 2004.

[27] Shihui Li, Yi Wu, Xinyue Cui, Honghua Dong, Fei Fang, and Stuart Russell. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In *AAAI*, 2019.

[28] Michael L Littman, Richard S Sutton, and Satinder P Singh. Predictive representations of state. In *NIPS*, 2002.

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

[30] Aditya Mahajan. Optimal decentralized control of coupled subsystems with control sharing. *IEEE Trans. Autom. Control*, 58(9):2377–2382, September 2013.

[31] Aditya Mahajan and Mehnaz Mannan. Decentralized stochastic control. *Annals of Operations Research,*, 241:109–126, June 2016.

[32] Aditya Mahajan and Sekhar Tatikonda. An algorithmic approach to identify irrelevant information in sequential teams. *Automatica*, pages 178–191, November 2015.

[33] 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, 2015.

[34] Ashutosh Nayyar, Aditya Mahajan, and Demosthenis Teneketzis. Optimal control strategies in delayed sharing information structures. *IEEE Trans. Autom. Control*, 56(7):1606–1620, July 2011.

[35] Ashutosh Nayyar, Aditya Mahajan, and Demosthenis Teneketzis. Decentralized stochastic control with partial history sharing: A common information approach. *IEEE Trans. Autom. Control*, 58(7):1644–1658, 2013.

[36] Thanh Thi Nguyen, Ngoc Duy Nguyen, and Saeid Nahavandi. Deep reinforcement learning for multi-agent systems: A review of challenges, solutions and applications. arXiv:1812.11794, 2018.

[37] J. M. Ooi, S. M. Verbout, J. T. Ludwig, and G. W. Wornell. A separation theorem for periodic sharing information patterns in decentralized control. *IEEE Trans. Autom. Control*, 42(11):1546–1550, November 1997.

[38] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[39] Joelle Pineau. The Machine Learning Reproducibility Checklist. https://www.cs.mcgill.ca/ jpineau/ReproducibilityChecklist.pdf, 2019.

[40] N. Sandell and M. Athans. Solution of some nonclassical lqg stochastic decision problems. *IEEE Trans. Autom. Control*, 19:108–116, 1974.

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

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

[43] 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, 2017.

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

[45] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, pages 1057–1063, Nov. 2000.

[46] Csaba Szepesvári. *Algorithms for reinforcement learning*. Morgan & Claypool, 2009.

[47] Cédric Villani. *Optimal transport: Old and New*, volume 338. Springer Science & Business Media, 2008.

[48] Daan Wierstra, Alexander Foerster, Jan Peters, and Juergen Schmidhuber. Solving deep memory POMDPs with recurrent policy gradients. In *International Conference on Artificial Neural Networks*, 2007.

[49] Daan Wierstra, Alexander Förster, Jan Peters, and Jürgen Schmidhuber. Recurrent policy gradients. *Logic Journal of the IGPL*, 18(5):620–634, 2010.

[50] T. Yoshikawa. Dynamic programming approach to decentralized stochastic control problems. *IEEE Trans. Autom. Control*, 20(6):796 – 797, December 1975.

[51] Kaiqing Zhang, Zhuoran Yang, Han Liu, Tong Zhang, and Tamer Basar. Fully decentralized multi-agent reinforcement learning with networked agents. In *ICML*, 2018.

Supplementary material

A Proof of Theorem 3

We first prove the following finite horizon version theorem:

Theorem 6 *Let $\{X_t\}_{t=1}^T$ be an (ε, δ)-approximate information state. Recursively define value functions $\{\widehat{V}_t\}_{t=1}^{T+1}$, where $\widehat{V}_t \colon X_t \mapsto \mathbb{R}$ as follows: $\widehat{V}_{T+1}(x_{T+1}) = 0$ and for $t \in \{T, \dots, 1\}$:*

$$\begin{aligned} \widehat{Q}_t(x_t, h_t) &= \mathbb{E}[R_t + \widehat{V}_{t+1}(X_{t+1}) \mid \widehat{X}_t = x_t, H_t = h_t] \\ \widehat{V}_t(x_t) &= \max_{h_t \in \mathcal{H}} \widehat{Q}_t(x_t, h_t). \end{aligned} \tag{7}$$

Suppose $\widehat{V}_t$ is Lipschitz continuous with Lipschitz constant L_V. Then, we have the following:

$$\begin{aligned} |Q_t(b_t, h_t) - \widehat{Q}_t(\rho_t(c_t), h_t)| &\le (T-t)(\varepsilon + L_V \delta) + \varepsilon \\ |V_t(b_t) = \widehat{V}_t(\rho_t(c_t))| &\le (T-t)(\varepsilon + L_V \delta) + \varepsilon, \end{aligned} \tag{8}$$

where b_t is the true belief state and c_t is the common information at time t.

PROOF We prove the result by backward induction. By construction, (8) is true at time $T+1$. This forms the basis of induction. Assume that (8) is true at time $t+1$ and consider the system at time t. Let $\kappa = (T-t-1)(\varepsilon + L_V\delta) + \varepsilon$. Then,

$$\begin{aligned} Q_t(b_t, h_t) &= \mathbb{E}[R_t + V_{t+1}(b_{t+1}) \mid C_t = c_t, H_t = h_t] \\ &\overset{(a)}{\le} \mathbb{E}[R_t + \widehat{V}_{t+1}(\rho_{t+1}(C_{t+1})) \mid C_t = c_t, H_t = h_t] + \kappa \\ &\overset{(b)}{\le} \Big(\mathbb{E}[R_t \mid X_t = \rho_t(c_t), H_t = h_t] + \varepsilon\Big) \\ &\quad + \Big(\mathbb{E}[\hat{V}_{t+1}(X_{t+1}) \mid X_t = \rho_t(c_t), H_t = h_t] + L_V\delta\Big) + \kappa \\ &= \widehat{Q}_t(\rho_t(c_t), h_t) + (T-t)(\varepsilon + L_V\delta) + \varepsilon. \end{aligned}$$

where (a) follows from the induction hypothesis and (b) follows from (AP1) and Remark 1. The reverse inequality can be proven using a similar argument. By maximizing over actions, we get the relationship between the value functions.

Proof of Theorem 3:

PROOF Taking limits as $T \to \infty$ in the statement of Theorem 6, we get the infinite horizon result stated in Theorem 3.

Remark 1 Kantorovich-Rubinstein duality [47] states that for any probability measures μ and ν on $\mathcal{X}$,

$$\mathcal{W}(\mu, \nu) = \sup_{\|f\|_{\text{Lip}} \le 1} \Big| \int_{\mathcal{X}} f d\mu - \int_{\mathcal{X}} f d\nu \Big|$$

where $\|f\|_{\text{Lip}}$ denotes the Lipschitz constant of a function f (with respect to the metric d). This along with (P2) imply that for a Lipschitz continuous function $\hat{V} \colon \hat{\mathcal{Z}} \to \mathbb{R}$ with Lipschitz constant L_V (with respect to the metric d),

$$\big|\mathbb{E}[\hat{V}(\hat{Z}_{t+1}) | H_t = h_t, U_t = u_t] - \mathbb{E}[\hat{V}(\hat{Z}_{t+1}) | \hat{Z}_t = \widehat{\imath}_t(h_t), U_t = u_t]\big| \le L_V \delta. \tag{9}$$

B Wasserstein distance vs other metrics: Theorem 3 Total Variation distance

If $\hat{V}$ is uniformly bounded with sup-norm L_∞, then the result of Theorem 3 holds with L_V replaced by L_∞. This can be proven as follows. By assumption, $\|\mu_t - \nu_t\|_{TV} < \delta$. This implies that for any function $\hat{V}_t$:

$$\Big| \int_{\hat{\mathcal{X}}} \hat{V}_t d\mu_t - \int_{\hat{\mathcal{X}}} \hat{V}_t d\nu_t \Big| \le \|\hat{V}_t\|_\infty \delta \le L_\infty \delta.$$

This condition can be used instead of the Kantorovich-Rubinstein duality in the proof of Theorem 3 to get the desired result.

C Proof of Theorem 4

PROOF We first note that even though the state estimator uses actions from the actor, the state estimation process itself is independent of the actor. It only depends on the dynamics of the environment. Hence, in the proposed three time-scale algorithm given by iterations (4), (5) and (6), the state estimator parameter update equation becomes decoupled from the other two iterations. The only condition that the state estimator iteration needs to satisfy with respect to the other two iterations is that it must use a faster learning rate. This condition is satisfied by the specified choice of learning rates. Then a suitable continuous time interpolation of (4) is an asymptotic pseudotrajectory of the semiflow induced by the corresponding ODE. Hence, it converges to a limit point of this ODE. From our supposition, this ODE is globally asyptotically stable and hence the iteration (4) converges to this fixed point [9, 25]. Similarly suitable continuous time interpolations of (5) and (6) are asymptotic pseudotrajectories of the semiflows induced by the corresponding ODEs. Hence, these iterations converge to the limit points of the corresponding ODEs. These ODEs are globally and locally asyptotically stable as per our supposition. Hence, the corresponding iterations converge to the limit points of these ODEs [26, page 35], [10, Theorem 23]. The discussion regarding convergence to a local maximum for (6) can be found in [25].

D Environments

D.1 Multi-access broadcast

This is a stylized example of a communication system where $n = 2$. In this example, there are two devices ($n = 2$) which try to broadcast over a multiple access channel. It is assumed that the packets arrive at each of the device $j \in \{1, 2\}$ according to a a Bernoulli process $\{W_t^i\}_{t=0}^{\infty}$ with probability of success denotes as p^j. At time t, device j may store or $B_t^j \in \{0, 1\}$ packets in a buffer. When the buffer is full and a packet arrives, it is dropped.

Transmission of a packet over the channel depends on the state of the channel which can either be *busy* at time t denoted by $S_t = 1$ or *idle* which is denoted by $S_t = 0$. The state of the channel S_t evolves in a Markovian manner with initial known distribution given by $P = \begin{bmatrix} \alpha_0 & 1-\alpha_0 \\ 1-\alpha_1 & \alpha_1 \end{bmatrix}$, the channel state process is independent of the packet arrival process at the device.

The action of device j at time t is denoted by $A_t^j \in \{0, 1\}$ which corresponds to the action *transmit* and *no transmit* respectively. The packet is removed from the buffer if only one device transmits and the channel is *idle*. The buffer state evolution is given by

$$B_{t+1}^j = \min\{B_t^j - A_t^j(1 - A_t^j)(1 - S_t) + w_t^j, 1\} \quad \forall j \in \{1, 2\}, \quad j = 3 - j$$

Each transmission costs κ and a successful transmission yields a reward r. The total reward for both the devices are given by

$$R_t = -(A_t^1 + A_t^2)\kappa + (A_t^1 \oplus A_t^2)(1 - S_t)r$$

Each agent $j \in \{1, 2\}$ perfectly observes the number of packets B_t^j in the buffer. Additionally, both the agents observe one step delayed actions (A_{t-1}^1, A_{t-1}^2) of each other along with the state S_t of the channel if either of them transmits.

For our experiments, we considered $\alpha_0 = \alpha_1 = 0.75$, the packet arrival probability for device 1 and device 2 is given by 0.3 and 0.6 respectively. We considered a per-step reward $r = 1$ for successful transmission and the transmission cost $\kappa = 0.4$

D.2 Demand response in smart grids

This stylized model for demand response proposed in [1] consists of a system with n agents, where, $\mathcal{S} = \{0, 1\}$, $\mathcal{A} = \{\emptyset, 0, 1\}$

$$P(\cdot \mid \cdot, \emptyset, z) = D \tag{10}$$

$$P(\cdot \mid \cdot, 0, z) = (1 - \varepsilon_1)\left[\begin{smallmatrix} 1 & 0 \\ 1 & 0 \end{smallmatrix}\right] + \varepsilon_1 D \tag{11}$$

$$P(\cdot \mid \cdot, 1, z) = (1 - \varepsilon_2)\left[\begin{smallmatrix} 0 & 1 \\ 0 & 1 \end{smallmatrix}\right] + \varepsilon_2 D. \tag{12}$$

Here, D denotes the "natural" dynamics of the systems and ε_1 and ε_2 are small positive constants. The per-step reward is given by:

$$R_t = -\left(\frac{1}{n} \sum_{i \in N} \left(c_0 \mathbb{1}_{\{A_t^i = 0\}} + c_1 \mathbb{1}_{\{A_t^i = 1\}} \right) + KL(M_t \| \zeta) \right), \tag{13}$$

where c_0 and c_1 are costs for taking actions 0 and 1 respectively, ζ is a given target distribution and $KL(M_t \| \zeta)$ denotes the Kullback-Leibler divergence between M_t and ζ.

In our experiments, we consider we consider a system with $n = 100$ agents, initial state distribution $P_0 = [1/3, 2/3]$, $D = \left[\begin{smallmatrix} 0.25 & 0.75 \\ 0.375 & 0.625 \end{smallmatrix}\right]$, $c_0 = 0.1$, $c_1 = 0.2$, $\zeta = [0.7, 0.3]$, $\varepsilon_1 = \varepsilon_2 = 0.2$ and discount factor $\gamma = 0.9$.

E Network architecture

E.1 Multi-access broadcast

We implemented three algorithms (Actor Critic, PPO and NAFDQN) for the multi-access broadcast environment. For PPO and Actor Critic we used the same network architecture, which is described below. For NAFDQN, we use the critic only network described below instead of the actor critic network.

- **State approximator:** We choose X_t to be 5-dimensional. The architectures of the two parts of the state approximator are as follows. (i) The AIS encoder is a two layer RNN where the first layer is a linear layer with 5 neurons and $\tanh$ activation and the second layer is a 5 neuron GRU layer. (ii) The feedforward NN part of the state approximator is a two layer NN with two heads—one for predicting $\hat{R}_t$ and the other for predicting ν_{t+1}. For the $\hat{R}_t$ part, the first layer is a 5 neuron linear layer with $\tanh$ activation and the second layer is a single neuron linear layer. For the ν_{t+1} part, the first layer is a 5 neuron linear layer with $\tanh$ activation and the second layer is a 5 neuron layer that outputs the means of a 5-dimensional multivariate Gaussian distribution with an identity covariance matrix.
- **Actor Critic network (Actor Critic, PPO):** Following conventional RL approach, we use a single network for both the actor and the critic with the action distribution and value given by two separate heads. The initial common part of the network is a single linear layer with 5 neurons and $\tanh$ activation. The actor head following this layer is a 3 neuron linear layer with $\log \texttt{softmax}$ activation. The critic head following the common first layer has two layers— the first is a 5 neuron linear layer with $\tanh$ activation followed by a single neuron linear layer. The prescription for the passive agent is generated from the output of the actor network as follows. The actor network outputs the probabilities of three actions—first passive agent gets opportunity to transmit, second passive agent gets opportunity to transmit and both passive agents get opportunity to transmit. The passive agents then transmit if they receive the opportunity to transmit and have a packet in their buffer, else they do not transmit.
- **Critic only network (NAFDQN)**: We used the implentation of NAFDQN from [24] as the base for our implementation. We use the same architecture as specified in [24]. To this implementation we have added the State approximator, which is described above.

For this experiment we use a discount rate of $\gamma = 0.9$. We use ADAM [22] as the optimizer for both the state approximator and the actor critic networks, with learning rates of 0.08 and 0.01 respectively, with the other ADAM parameters being the default values. We use a batch size of 300 for this infinite horizon environment. We use the pytorch deep learning library [38] for this implementation. We evaluate the performance periodically using 100 independent Monte Carlo evaluations. We ran our experiments in a machine with Intel Xeon processor with 4 cores with a clock speed of 2.80GHz and an NVIDIA Quadro K1200 GPU.

E.2 Demand response in smart grids

For this problem, we do not need a State approximator as in this case the common observation (which is the mean-field) itself acts an an information state [20]. Hence, we only need the actor and

critic networks. We use three algorithms—TRPO [41], PPO [42] and NAFDQN [15] using their implementations in ChainerRL [11]. The network architecture used for these experiments are the default ones from Chainer RL, reproduced below:

- **PPO:** The policy network consists of three layers. The first two layers are linear layers with 64 neurons and $\tanh$ activation. The final layer has 6 neurons and it outputs the means of a multi-variate Gaussian with state independent covariance as the distribution over actions. The value network consists of two linear layers with 64 neurons with the first linear layer having a $\tanh$ activation. We use ADAM as the optimizer with the learning rate as 0.0003.
- **TRPO**: For TRPO the policy and value networks are identical to the networks used for PPO described above. We use ADAM as the optimizer for with the learning rate as 0.001.
- **NAF-DQN** This is a critic only method where the value network consists of three linear layers. The first two layers are 100 neuron layers with with `ReLU` activation. The output (last) layer has a single neuron value head, and two 6 neuron head to predict the μ and L functions in NAFDQN. [15]. We use ADAM as the optimizer for with the learning rate as 0.001.

For this experiment, we use a discount factor of $\gamma = 0.9$. We evaluate the performance periodically using 10 independent Monte Carlo evaluations. We use a batch size of 200 for this infinite horizon environment. We ran our experiments in a machine with Intel Xeon processor with 4 cores with a clock speed of 2.80GHz and an NVIDIA Quadro K1200 GPU.

F Reproducibility Checklist

We follow the reproducibility checklist from [39] and include further details here. For all the models and algorithms we have included details that we think would be useful for reproducing the results of this work.

- For all **models** and **algorithms** presented, check if you include:
  1. *A clear description of the mathematical setting, algorithm, and/or model*: **Yes**. The algorithm is described in detail in Section 5, with all the loss functions used for training being clearly defined. The details of the architecture, hyperparameters used and other algorithm and environment details are given in Sections E and D in the Supplementary material.
  2. *An analysis of the complexity (time, space, sample size) of any algorithm*: **No**. We do not include a formal complexity analysis of our algorithm. However, we do highlight the additional computational steps (in terms of losses and parameter updates) in Sec. 5 over standard single agent RL algorithms that would be needed in our approach.
  3. *A link to a downloadable source code, with specification of all dependencies, including external libraries.*: **No**. We will make the source code available if the paper is selected with the camera-ready version.
- For any **theoretical claim**, check if you include:
  1. *A statement of the result*: **Yes**. We have stated our key results in Theorems 3, 4 and 5. We have included proofs of these in the Supplementary material.
  2. *A clear explanation of any assumptions*: **Yes**. We have stated all our assumptions either as part of the theorem statements or separately titled as an Assumption.
  3. *A complete proof of the claim*: **Yes**. We have included proofs of all proposed theorems in the Supplementary material.
- For all **figures** and **tables** that present empirical results, check if you include:
  1. *A complete description of the data collection process, including sample size*: **NA**. We did not collect any data for our work.
  2. *A link to a downloadable version of the dataset or simulation environment*: **No**. We will make the source code available if the paper is selected with the camera-ready version.
  3. *An explanation of any data that were excluded, description of any pre-processing step*: **NA**. We did not perform any pre-processing step.

4. *An explanation of how samples were allocated for training / validation / testing*: **Yes**. The number of samples (steps) used for learning (training) by various algorithms is given in our plots (Fig. 2). We periodically evaluate the performance of our policy using multiple Monte Carlo runs. The details of the number of runs and the batch sizes used for various environments are given in the Supplementary material in Sec. E.
5. *The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results*: **Yes**. We did not do any hyperparameter tuning as part of this work. All the hyperparameters that we used are specified in the Supplementary material in Sec. E.
6. *The exact number of evaluation runs*: **Yes**. For all our environments, we repeat the experiment 25 times. We state this in the caption in Fig. 2. We do not fix any seeds. We periodically evaluate the performance of our policy using multiple Monte Carlo runs. The details of the number of runs and the batch sizes used for various environments are given in the Supplementary material in Sec. E.
7. *A description of how experiments were run*: **No**. We will make the source code available if the paper is selected with the camera-ready version. We will provide a README with instructions on how to run the experiments along with the source code.
8. *A clear definition of the specific measure or statistics used to report results*: **Yes**. We plot the median and the region between the first and third quartile across 25 runs for all our numerical experiments. This is stated in the caption in Fig. 2.
9. *Clearly defined error bars*: **Yes**. We plot the median and the region between the first and third quartile across 25 runs for all our numerical experiments. This is stated in the caption in Fig. 2.
10. *A description of results with central tendency (e.g. mean) & variation (e.g. stddev)*: **Yes**. We plot the median and the region between the first and third quartile across 25 runs for all our numerical experiments. This is stated in the caption in Fig. 2.
11. *A description of the computing infrastructure used*:**Yes**. We have provided this detail in the Supplementary material in Sec. E. We ran our experiments in a machine with Intel Xeon processor with 4 cores with a clock speed of 2.80GHz and an NVIDIA Quadro K1200 GPU.