Communication-Free Two-Stage Multi-Agent DDPG under Partial States and Observations

1st Joohyun Cho
Dept. of ECE
Univ. of Utah
joohyun.cho@utah.edu

2nd Mingxi Liu
Dept. of ECE
Univ. of Utah
mingxi.liu@utah.edu

3rd Yi Zhou
Dept. of ECE
Univ. of Utah
yi.zhou@utah.edu

4th Rong-Rong Chen
Dept. of ECE
Univ. of Utah
rchen@ece.utah.edu

2021 55th Asilomar Conference on Signals, Systems, and Computers | 978-1-6654-5828-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/IEEECONF53345.2021.9723197

***Abstract*—In this work, we propose a two-stage multi-agent deep deterministic policy gradient (TS-MADDPG) algorithm for communication-free, multi-agent reinforcement learning (MARL) under partial states and observations. In the first stage, we train prototype actor-critic networks using only partial states at actors. In the second stage, we incorporate partial observations resulting from prototype actions as side information at actors to enhance actor-critic training. This side information is useful to infer the unobserved states and hence, can help reduce the performance gap between a network with fully observable states and a partially observable one. Using a case study of building energy control in the power distribution network, we successfully demonstrate that the proposed TS-MADDPG can greatly improve the performance of single-stage MADDPG algorithms that use partial states only. This is the first work that utilizes partial local voltage measurements as observations to improve the MARL performance for a distributed power network.**

***Index Terms*—Multi Agent Reinforcement Learning, Policy Gradient, Partially Observable, Actor-Critic**

I. Introduction

Multi-Agent reinforcement learning (MARL) [1]–[4] has been applied to a variety of applications including traffic control [5], energy distribution [6], robots [7], and the analysis of economic and social problems [8]. In MARL, agents learn collaboratively to find optimal policies. Policy gradient based MARL algorithms [2], [4], in particular, have been proposed to directly optimize policy parameters along the gradient direction. For continuous action space and deterministic policies, multi-agent deep deterministic policy gradient (MADDPG) [4] applies model-free, off-policy actor-critic algorithm to compute policy gradient. It adopts the framework of centralized training with decentralized execution to overcome the challenge of non-stationarity in MARL. Note that in MADDPG, while the critic is augmented with full state information and other agents' policies to enhance training, each agent often has access to partially available states under partially observable environments. This causes performance degradation of MADDPG compared to the case with full state.

In this work, to tackle the challenge of partial observability, we propose a two stage MADDPG algorithm, termed as TS-MADDPG, to improve the performance of the existing one-stage MADDPG by utilizing side information. This work is motivated from the study of a power network, where each agent has access to not only local state (baseline demand), but also local voltage measurements, which is a function of both globe states and actions (power consumption). We propose the use of side information (voltages in the case of power network) to improve the performance of MADDPG. In the first stage of the proposed TS-MADDPG, we train prototype actor-critic networks assuming actors only have access to local state information. Once the prototype networks are trained, we train the second stage actor-critic networks assuming actors now have access to not only local state information, but also local voltage measurements. The latter result from actions generated using the first stage prototype networks. We show performance enhancement of TS-MADDGE over that the one-stage version using the power network as a case study.

While the use of MADDPG has been investigated in the literature for power networks [9], to the best of our knowledge, this is the first work that utilizes voltage measurements and the two stage design to improve its performance. We note that the proposed TS-MADDPG has a lower training complexity than those based on the recursive structure [10], [11]. This is because the two stage training is done in a sequential manner, which provides better training stability than that of the recursive based structure. In addition, in TS-MADDPG, we do not require any communication among agents. This saves the communication overhead required for the MAML algorithms in [1].

The remainder of the paper is organized as follows: Section II provides the background of MARL. Section III presents the architecture and algorithm of the proposed TS-MADDPG. Section IV introduces a case study of power distribution network. In Section V, we present simulation studies of TS-MADDPG and compare with other existing algorithms. Section VI includes conclusions and future work.

II. Background

We consider multi-agent reinforcement learning (MARL) with a decentralized markov decision process (MDP) and partially observable states, denoted as $(\mathcal{M}, \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$. Here, $\mathcal{M}$ is a set of m agents, $\mathcal{S} = \times_i \mathcal{S}^{(i)}$ is the set of joint state space, $\mathcal{A} = \times_i \mathcal{A}^{(i)}$ is the joint action space, $\mathcal{R}$ is the

The work by Rong-Rong Chen is supported in part by NSF grants CCF-1817154 and ECCS-1824558. The work of Yi Zhou is supported in part by NSF grants CCF-2106216 and DMS-2134223.

reward function. Each agent i executes action $a^{(i)} \in \mathcal{A}^{(i)}$. The joint action $a = (a^{(1)}, \cdots, a^{(m)})$ causes state transition from $s \in \mathcal{S}$ to $s' \in \mathcal{S}$ with probability $P(s'|s;a) = \mathcal{P}(s,a;s')$. Each agent i only has access to its local state $s^{(i)}$ and has its own policy $\mu^{(i)} : \mathcal{S}^{(i)} \to \mathcal{A}^{(i)}$. The joint policy is denoted as $\mu = (\mu^{(1)}, \cdots, \mu^{(m)})$. The agents receive a shared joint reward of $r_{t+1} = \mathcal{R}(s_t, a_t)$ at each time $t+1$. The goal is to maximize the expected return, $J = \mathbb{E}(\sum_{t=0}^{\infty} \gamma^t r_{t+1})$, where γ is the discount factor.

While the MADDPG algorithm can be applied to an MDP with partially observable states, the performance loss can be significant compared to that with full states. Hence, in this work, we propose to compensate this loss by the use of local observations. Specifically, at each time t, we assume that each agent also has access to local observations $o_t^{(i)} \in \Omega^{(i)}$, which are determined by the joint action a_{t-1} and joint state s_{t-1} from time $t-1$. We propose to incorporate this side information in our RL algorithm design such that the agents can take advantage of these to find better policies. We note that our MARL formulation is different from that of the standard decentralized partially observable MDP (Dec-POMDP). The latter assumes that each agent takes an action based on local observations only. However, in our setting, local observations are used as side information besides the local states to help generate better policies. Furthermore, the states considered in our MDP formulation cannot be thought of observations in the Dec-POMDP because they are not necessary a consequence of the actions.

III. Two-Stage Multi-Agent DDPG (TS-MADDPG)

In this section, we present the proposed TS-MADDPG. We will begin with a brief description of the one-stage MADDPG, followed by the proposed two-stage design.

One-stage MADDPG: We can apply the MADDPG algorithm to optimize the multi-agent policies. This gives the one-stage MADDPG. Each policy is parameterized by weights θ. To handle nonstationarity in the multi-agent framework, each agent's critic uses all agents' states and actions for training, while each actor uses only its partial states. The policy gradient with respect to agent i's policy parameterization is

$$\nabla_{\theta_\mu^{(i)}} J(\mu^{(i)}) = \mathbb{E}_{s,a\sim\mathcal{D}}\Big[\nabla_{\theta_\mu^{(i)}} \mu^{(i)}(a^{(i)}|s^{(i)}) \nabla_{a^{(i)}} Q_\mu^{(i)}(s,a)|_{a^{(i)}=\mu^{(i)}(s^{(i)})}\Big],$$

where $\mathcal{D}$ is replay buffer.

We modify the actor critic network proposed in DDPG [12] for the One-stage MADDPG. The one-stage MADDPG actor network consists of three fully connected layers, each of which has layer normalization and ReLU, the final output layer of the actor is a $tanh$ layer. Critic network has two separate inputs(states, actions). states input is through 2 fully connected layer with Layer normalization and ReLU and actions are not included until the second fully connected layer of the critic network. The last fully connected layer is attached to the critic network output.

Two-stage MADDPG: While one can directly apply the one-stage MADDPG to train actor-critic networks, we find that, due to the partial state input $s^{(i)}$, the performance of the trained networks is much inferior to that of the case where each agent has access to the full state s. Thus, to overcome this issue, we propose a novel two-stage MADDPG algorithm, termed TS-MADDPG. We employ a novel actor-critic design that allows the networks trained in the first stage to interact with the environment and generate partial observations. The resulting partial observations, together with the partial actions that led to these observations, are used as the input of the new actor network in the second stage. This approach allows the network to implicitly learn the unobserved states from the past experience.

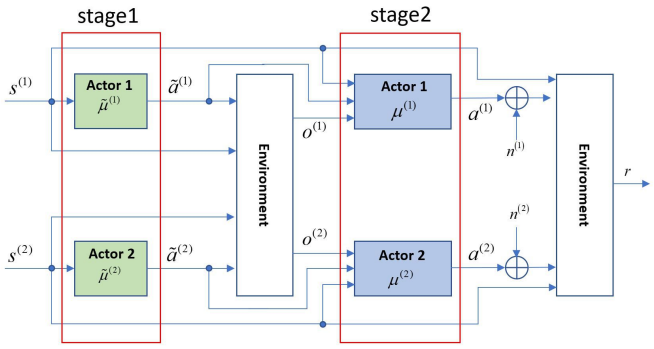
Figure 1a shows a diagram of the sample generation in the training of TS-MADDPG with two agents. The actor networks in stage 1, denoted by $\tilde{\mu}^{(1)}$ and $\tilde{\mu}^{(2)}$, are obtained by a one-stage MADDPG algorithm with partial state inputs $s^{(1)}$ and $s^{(2)}$, respectively. The outputs of the stage 1 actor networks produce initial actions $a = (\tilde{a}^{(1)}, \tilde{a}^{(2)})$. By applying joint action a to the environment, we obtain observations $o = (o^{(1)}, o^{(2)})$. The action-observation pair $(\tilde{a}^{(i)}, o^{(i)})$ and $s^{(i)}$ are inputs to the second stage actor network for each agent i. In Figure 1b, we show the actor and critic updates in TS-MADDPG training. Note that the mini-batches are taken from the replay buffer that are filled with samples generated as shown in Figure 1a. A detailed description of the TS-MADDPG is shown in Algorithm 1 at the end of this document.

IV. A Case Study of Decentralized Building Energy Control in Power Distribution Network

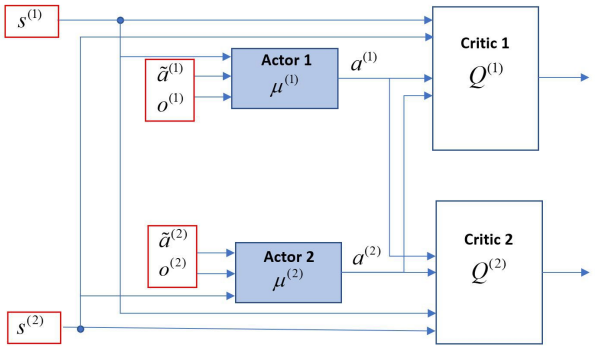
To evaluate the proposed TS-MADDPG in practical applications, we conduct a case study by considering a building energy control problem in a power distribution network for reliable and low-cost grid operation. To simplify the discussion and better illustrate the proposed TS-MADDPG, assume each node of the distribution network is connected to only one building complex whose real and reactive power consumption and generation can be controlled through charging and discharging onsite energy storage systems, setpoint adjustment of HVAC units, and generation adjustment of power generation units. Each building is an *agent* and the whole power distribution network is the *environment*. The radial power distribution network (the environment) used here is a simplified single-phase IEEE-13 Node Test Feeder, as shown in Figure 2.

Let the 13 nodes indexed by $i = 0, \ldots, 12$, where Node $i = 0$ is the feeder head with a constant voltage V_0. To ensure reliable grid operation, voltage magnitudes at all nodes should be maintained within a certain range. Let $\boldsymbol{V} \in \mathbb{R}^{12}$ denote the vector containing the squared voltage magnitude p.u. of all the remaining 12 nodes. Then based on the LinDistFlow model, at any time instant, we have

$$\boldsymbol{V} = V_0 \mathbf{1}_{12} - 2\boldsymbol{R}\boldsymbol{P} - 2\boldsymbol{X}\boldsymbol{Q}, \tag{1}$$



(a) Replay buffer sample generation



(b) Actor-critic updates

Fig. 1: Illustration of TS-MADDPG training with two agents

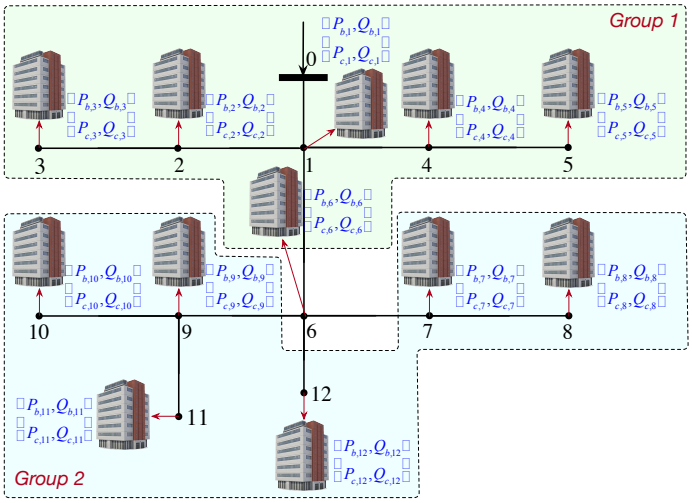


Fig. 2: Case study: Building energy control in a power distribution network.

where $\boldsymbol{P} = \boldsymbol{P}_b + \boldsymbol{P}_c \in \mathbb{R}^{12}$ and $\boldsymbol{Q} = \boldsymbol{Q}_b + \boldsymbol{Q}_c \in \mathbb{R}^{12}$ are the net real and reactive power consumption at all 12 nodes with positive indicating consumption and negative indicating generation, and matrices $\boldsymbol{R} \in \mathbb{R}^{12\times 12}$ and $\boldsymbol{X} \in \mathbb{R}^{12\times 12}$ are constant matrices derived from the distribution network topology and configuration. In the proposed TS-MADDPG framework, $\boldsymbol{P}_b$ and $\boldsymbol{Q}_b$ are the baseline net real and reactive power consumption vectors, which are regarded as *state*, $\boldsymbol{P}_c$ and $\boldsymbol{Q}_c$ are the controllable net real and reactive power consumption, which are considered as *action*. We note that the actions here are continuous valued. The voltage magnitude V_i, at the ith node, is considered as *local observation*. Similarly, by using the LinDistFlow model, the total power loss of the entire distribution network at any time instant can be calculated as

$$L(\boldsymbol{P}, \boldsymbol{Q}) = \boldsymbol{P}^{\mathsf{T}} \tilde{\boldsymbol{R}} \boldsymbol{P} + \boldsymbol{Q}^{\mathsf{T}} \tilde{\boldsymbol{X}} \boldsymbol{Q}, \tag{2}$$

where $\tilde{\boldsymbol{R}} \in \mathbb{R}^{12}$ and $\tilde{\boldsymbol{X}} \in \mathbb{R}^{12}$ are constant matrices derived from the distribution network topology and configuration. The local generation function $c_i(p, q)$ of each building i can take the form as

$$c_i(p,q) = \alpha_{i,P} \cdot p^2 + \beta_{i,P} \cdot p + \gamma_{i,P} + \alpha_{i,Q} \cdot q^2 + \beta_{i,Q} \cdot q + \gamma_{i,Q}, \tag{3}$$

where $\alpha_{i,P} > 0$ and $\alpha_{i,Q} > 0$.

Moreover, at any time, the negative total power loss of the distribution network $-L(\boldsymbol{P}, \boldsymbol{Q})$ is regarded as the *global reward*, and the negative generation/consumption cost $-c_i(P_{c,i}, Q_{c,i})$ of each building i is regarded as the *local reward*. The goal is to minimize the total power loss plus all the local generation/consumption costs, which can be formulated as the following optimal power flow problem.

$$\begin{aligned} \min_{\boldsymbol{P}_c, \boldsymbol{Q}_c} \; & L(\boldsymbol{P}, \boldsymbol{Q}) + \sum_{i=1}^{12} c_i(\boldsymbol{P}_c(i), \boldsymbol{Q}_c(i)) \\ \text{s.t.} \; & \underline{\boldsymbol{P}}_c \le \boldsymbol{P}_c \le \overline{\boldsymbol{P}}_c \\ & \underline{\boldsymbol{Q}}_c \le \boldsymbol{Q}_c \le \overline{\boldsymbol{Q}}_c \\ & \underline{\boldsymbol{V}} \le \boldsymbol{V} \le \overline{\boldsymbol{V}}, \end{aligned} \tag{4}$$

where $\underline{\boldsymbol{P}}_c$, $\overline{\boldsymbol{P}}_c$, $\underline{\boldsymbol{Q}}_c$, and $\overline{\boldsymbol{Q}}_c$ are vectors containing local physical limits of all buildings' energy units, and $\underline{\boldsymbol{V}}$ and $\overline{\boldsymbol{V}}$ are vectors denoting the nodal voltage bounds.

V. Numerical Results

In this section, we present numerical results of the proposed TS-MADDPG using the case study of building energy control in a power distribution network. The following hyper parameters are adopted in the simulations.

- Actor and critic learning rate: 2.5×10^{-4}
- Soft target update τ : 0.001
- Dimension of the 1st, 2nd hidden layers in actor and critic network: 64
- Action-observation pair $(\tilde{a}^{(i)}, o^{(i)})$ and $s^{(i)}$ are concatenated for the inputs to the second stage actor network for each agent i
- Exploration noise process: Gaussian with $\mu = 0$ and $\sigma = 0.05$
- Replay buffer size: 10^5, Batch size: 1024

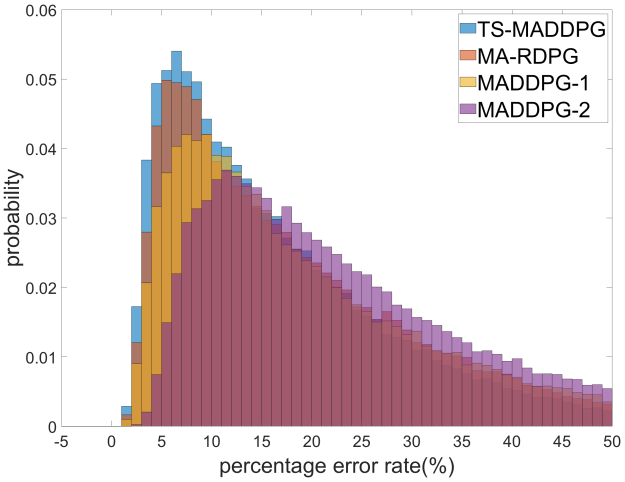
The training and testing data are generated in the following way: The feeder head voltage v_0 is 2.97 kV. 50,000 sets of data were generated for training and another 10,000 sets were generated for testing. In each set of data, $\boldsymbol{P}_b$ was generated randomly by following a uniform distribution in ± 210 kW to mimic real power generation and consumption,

$\boldsymbol{Q}_b$ was generated accordingly to maintain 0.9 power factor, the optimal $\boldsymbol{P}_c$ and $\boldsymbol{Q}_c$ were obtained by directly solving Eqn. (4) using quadratic programming with ± 105 kW (kVar) real (reactive) power bounds and [0.95, 1.05] p.u. voltage magnitude bounds, the optimal local and global rewards were generated accordingly and were used as benchmark.

In Figure 3, using the power distribution network described in Section IV, we show performance comparisons of the proposed TS-MADDPG with two versions of MADDPG (one-stage only) and MA-RDPG, in which DDPG of MADDPG [4] is replaced with RDPG [10]. We divide the 12 nodes (see Fig. 2) into two groups of six nodes each and treat each group as an agent. Each agent has access to only the local states and observations of the six nodes in the group. The MA-RDPG and two versions of one-stage MADDPG have partial states as inputs, but one (MADDPG-1) has a shared critic for both agents, and the other (MADDPG-2) has a separate critic for each agent. The Figure 3a shows a histogram of the evaluation results of the trained networks using the three methods. The $x-$axis is the reward percentage error rate (PER), defined as $\mathcal{E}_r = (r^* - r)/r^*$, where r^* is the optimal reward obtained by a conventional centralized optimization algorithm, and r is the reward obtained by applying the generated actions using each of the three RL algorithms. Here, the expectation is calculated over a total of $6{\cdot}10^4$ independently generated states. Here, the states denote the nodal baseline power consumption/generation, which are in nature independent of each other and over time. This corresponds to $T = 1$ and a discount factor $\gamma = 0$ in Algorithm 1. We assume that components of each state vector follow a Gaussian distribution with zero mean and a variance of 10^6. We note that the conventional algorithm is centralized and assumes full knowledge of the states and the distribution network. It has to be re-run for each new state to compute r^*. The four RL algorithms, on the other hand, do not assume any prior knowledge of the power network topology and configuration.

From Figure 3, we see that the histogram of TS-MADDPG has a peak that is noticeably higher and located closer to the left side of the graph. This shows that for a higher percentage of states, TS-MADDPG can generate near optimal actions with rewards that are closer to those of the optimal values (corresponding to a smaller reward PER). We also observe that, with the one-stage MADDPG-1 and MADDPG-2, the histograms show a heavier tail. This means the probability that the one-stage MADDPG algorithms fail to generate a near optimal action is higher than that of TS-MADDPG. These observations are further confirmed in Figure 3b. We note that TS-MADDPG consistently achieves a lower reward PER with higher probabilities. MA-RDPG shows relatively lower PER than the two MADDPGs, but it requires more training episodes to achieve the optimized performance as depicted in Figure 3c. TS-MADDPG's first stage training time is similar to MADDPG but the second stage can be trained with a much smaller number of episodes.

We comment that while TS-MADDPG shows improved performance compared to the one-stage algorithms, the average reward PER remains about 17.5%, which is higher than the 8% PER of a centralized MADDPG algorithm with full states. This occurs likely because the large variance of the states causes some unresolved uncertainty in the actor network training due to partial states, even with the help of local observations. We plan to extend TS-MADDPG to multiple stages ($\geq$ 3) or take into account additional past action-observations pairs in order to further reduce this gap.



(a) Histograms of reward percentage error rate (PER)

		TS-MA DDPG	MA-RDPG	MA DDPG-1	MA DDPG-2
$\mathbb{E}[\mathcal{E}_r]$		17.5%	19.21%	28.25%	28.25%
$\Pr(\mathcal{E}_r < x)$	$x = 5\%$	10.79%	8.49%	0.97%	0.97%
	$x = 10\%$	35.81%	32.23%	13.98%	13.98%
	$x = 15\%$	54.39%	49.64%	31.75%	31.75%
	$x = 20\%$	68.26%	63.28%	46.89%	46.89%
	$x = 25\%$	78.24%	73.63%	59.2%	59.2%
	$x = 30\%$	85.21%	81.44%	68.77%	68.77%

(b) $\mathcal{E}_r$ denotes reward PER achieved by the RL algorithm

	# of episodes for training
TS-MADDPG	1st stage 620,000 2nd stage 10,000
MA-RDPG	1,140,000
MADDPG-1	620,00
MADDPG-2	750,00

(c) The number of episodes for training over various algorithms

Fig. 3: Performance comparisons over different MARL architectures.

VI. Conclusions and Future Work

In this work, we proposed a novel two-stage MADDPG algorithm that can effectively improve the agents' collaboration without the need for communications among agents. The use of local observations as side information to enhance

the learning performance of a one-stage MADDPG with partial states is a main novelty of this work. We demonstrate substantial improvements of TS-MADDPG using a case study of the power distribution network. In prior studies of RL for power networks, only partial states were used to train the optimal control policies. This work is the first to use voltage observations in additional to partial states to train actor-critic networks and show improved performance. As a continuation of this work, we plan to investigate multi-stage MADDPG (MS-MADDPG) to examine the impact of having a greater number of stages (≥ 3) on the RL performance. It is expected that the performance of MS-MADDPG will continue to improve as the number of stage increases. Furthermore, we note that the training time of the later stage networks decreases greatly as the trained networks from previous stage continue to improve. We also plan to explore the possibility of designing a fixed actor network for each agent that it can be applied iteratively under the same environment, using refined action and observation pairs from previous iterations as inputs.

REFERENCES

[1] S. Sukhbaatar, A. Szlam, and R. Fergus, "Learning multiagent communication with backpropagation," ser. NIPS'16. Curran Associates Inc., 2016, p. 2252–2260.

[2] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018.

[3] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 80. PMLR, 10–15 Jul 2018, pp. 4295–4304.

[4] R. Lowe, Y. WU, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[5] P. K.J., H. K. A.N, and S. Bhatnagar, "Multi-agent reinforcement learning for traffic signal control," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2014, pp. 2529–2534.

[6] I. Dusparic, C. Harris, A. Marinescu, V. Cahill, and S. Clarke, "Multi-agent residential demand response based on load forecasting," in *2013 1st IEEE Conference on Technologies for Sustainability (SusTech)*, 2013, pp. 90–96.

[7] M. Colby, L. Yliniemi, and K. Tumer, "Autonomous multiagent space exploration with high-level human feedback," *Journal of Aerospace Information Systems*, vol. 13, no. 8, pp. 301–315, 2016.

[8] J. Z. Leibo, V. Zambaldi, M. Lanctot, J. Marecki, and T. Graepel, "Multi-agent reinforcement learning in sequential social dilemmas," in *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '17. International Foundation for Autonomous Agents and Multiagent Systems, 2017, p. 464–473.

[9] D. Cao, W. Hu, J. Zhao, Q. Huang, Z. Chen, and F. Blaabjerg, "A multi-agent deep reinforcement learning based voltage regulation using coordinated pv inverters," *IEEE Transactions on Power Systems*, vol. 35, no. 5, pp. 4120–4123, 2020.

[10] N. Heess, J. J. Hunt, T. P. Lillicrap, and D. Silver, "Memory-based control with recurrent neural networks," *NIPS Deep Reinforcement Learning Workshop*, 2015.

[11] R. E. Wang, M. Everett, and J. P. How, "R-MADDPG for partially observable environments and limited communication," *ICML Reinforcement Learning for Real Life Workshop*, 2019.

[12] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *4th International Conference on Learning Representations, ICLR*, 2016.

Algorithm 1: Two-Stage Multi-Agent DDPG under Partial States and Observations

Stage 1: Train the first stage actor-critic network $\tilde{\mu}^{(i)}(s^{(i)})$ and $\tilde{Q}^{(i)}(s,a)$ by MADDPG algorithm, $i = 1, \cdots, m$;

Stage 2: Load pre-trained first stage actor-critic networks $\tilde{\mu}^{(i)}$, $\tilde{Q}^{(i)}$, $i = 1, \cdots, m$

Randomly initialize all agents' second stage actor network $\mu^{(i)}(s^{(i)}, \tilde{a}^{(i)}, o^{(i)})$ with weight $\theta_\mu^{(i)}$, $i = 1, \cdots, m$

Initialize replay buffer $\mathcal{D}$

for *episode=1 to* M_2 **do**

  Initialize a random process $\mathcal{N}$ for action exploration

  Start with a new state s

  for t*=1 to max-episode-length* **do**

    for each agent i, select action $\tilde{a}^{(i)} = \tilde{\mu}^{(i)}(s^{(i)})$ according to the current policy

    Execute actions $\tilde{a} = \left(\tilde{a}^{(1)}, \cdots, \tilde{a}^{(m)}\right)$ and obtain local observation $o = \left(o^{(1)}, \cdots, o^{(m)}\right)$

    for each agent i, select action $a^{(i)} = \mu^{(i)}(s^{(i)}, \tilde{a}^{(i)}, o^{(i)}) + \mathcal{N}_t$ w.r.t the current policy and exploration noise

    Execute actions $a = \left(a^{(1)}, \cdots, a^{(m)}\right)$ and observe reward r and the new state s' and the new observation o'

    Store $(s, \tilde{a}, o, a, r, s', o')$ in replay buffer $\mathcal{D}$

    $s \leftarrow s'$

    for *agent* $i = 1$ *to* m **do**

      Sample a random mini-batch of S samples $(s_j, \tilde{a}_j, o_j, a_j, r_j, s'_j, o'_j)$ from $\mathcal{D}$, j is the sample index

      Set $y_j = r_j^{(i)} + \gamma Q_{\mu'}^{\prime(i)}(s', a'^{(1)}, \cdots, a'^{(m)})|_{a'^{(i)}}$, where $a'^{(i)} = \mu^{(i)}(s_j'^{(i)}, a_j^{(i)}, o_j'^{(i)})$

      Update critic by minimizing the loss $\mathcal{L} = \frac{1}{S}\sum_j \left(y_j - Q_\mu^{(i)}(s_j, a_j)\right)^2$

      Update actor using the sampled policy gradient:

$$\nabla_{\theta_\mu^{(i)}} J \approx \frac{1}{S}\sum_j \Big[\nabla_{\theta_\mu^{(i)}} \mu_j^{(i)} \cdot \nabla_{a^{(i)}} Q_\mu^{(i)}\Big(s_j, a_j^{(1)}, \cdots, a_j^{(i)}, \cdots, a_j^{(m)}\Big)\Big|_{a_j^{(i)}=\mu_j^{(i)}}\Big]$$

      where $\mu_j^{(i)} = \mu^{(i)}(s_j^{(i)}, \tilde{a}_j^{(i)}, o_j^{(i)})$

    end

    Update target network parameters for each agent i:

$$\theta_\mu^{\prime(i)} \leftarrow \tau\theta_\mu^{(i)} + (1-\tau)\theta_\mu^{\prime(i)}$$

$$\theta_Q^{\prime(i)} \leftarrow \tau\theta_Q^{(i)} + (1-\tau)\theta_Q^{\prime(i)}$$

  end

end