Improved Cooperative Multi-agent Reinforcement Learning Algorithm Augmented by Mixing Demonstrations from Centralized Policy

Paper #418

ABSTRACT

Many decision problems for complex systems that involve multiple decision makers can be formulated as a decentralized partially observable markov decision process (dec-POMDP) problem. Due to the computational difficulty with obtaining optimal policies, recent approaches to dec-POMDP often use a multi-agent reinforcement learning (MARL) algorithm. We propose a method to improve the existing cooperative MARL algorithms by adopting an imitation learning technique. For a reference policy in the imitation learning part, we use a centralized policy from a multi-agent MDP or a multi-agent POMDP model reduced from the original dec-POMDP model. In the proposed method, during the training process, we mix demonstrations from the reference policy by using a demonstration buffer. Demonstration samples from the buffer are used in the augmented policy gradient function for policy updates. We assess the performance of the proposed method for three well-known dec-POMDP benchmark problems – Mars rover, co-operative box pushing, and dec-tiger. Experimental results indicate that augmenting the baseline MARL algorithm by mixing the demonstrations significantly improves the quality of policy solutions. With these results, we conclude that the imitation learning can enhance MARL algorithms and that policy solutions from MMDP and MPOMDP models are a reasonable reference policy to use in the proposed algorithm.

KEYWORDS

Multi-agent reinforcement learning; Cooperative decision making problem; dec-POMDP; Imitation learning

1 INTRODUCTION

Cooperative multi-agent decision making problems are important to designing and operating complex systems, for example, smart factory [23, 41, 42] and disaster response system [1, 35]. Complex systems are characterized by a large number of interrelated elements to achieve predefined objectives [16], and many decision makers including humans and autonomous agents are involved in their operation. Decision makers make their decisions based on individual observations about the system, and often these observations available to the decision makers do not contain complete information. In order to make the best decision to achieve the system's overall objectives, decision makers must overcome the limitation of insufficient information to make coherent, well-aligned decisions. This is the goal of a cooperative multi-agent decision making problem.

Decentralized-Partially Observable Markov Decision Process (dec-POMDP) is a suitable choice to model a cooperative multi-agent sequential decision making problem. The objective of a dec-POMDP model is to find the best policy for each agent when multiple agents use partial observations on the system state to decide individual actions to achieve a common goal. In a dec-POMDP environment, individual agents do not know the true state of the system and other agents' actions, and must determine their action only based on their individual observation sequences. This makes a dec-POMDP model much harder to solve than a single agent decision model such as multi-agent MDP (MMDP) and multi-agent POMDP (MPOMDP) [4]. Various solution methods to find an optimal policy have been developed for finite horizon dec-POMDP problems, including dynamic programming, heuristic search, mixed integer programming or techniques based on sufficient statistics [3, 9, 15, 28, 38]. However, these exact solution methods face a scalability problem when applying to large-scale problems.

In recent years, deep Reinforcement Learning (deep-RL) has become a popular solution technique for dec-POMDP [10–12, 14, 24, 33, 39]. In general, deep-RL algorithms do not always guarantee an optimal policy solution for dec-POMDP problems. Nevertheless, algorithms based on deep-RL can efficiently search the solution space and find good policy solutions. This makes a deep-RL approach particularly useful when a target problem is too complex to solve with optimal solution algorithms. Building upon its success in solving single agent problems, there are many recent studies to develop deep-RL algorithms for multi-agent problems.

In this paper, we propose a method to improve the performance of a deep-RL approach to solving dec-POMDP problems. In particular, we augment a cooperative multi-agent RL (MARL) algorithm by incorporating an imitation learning technique. In the RL process, we use a mix of two streams of sample paths: ones that are generated from a current policy and demonstration sample paths from a reference policy. Some precautions are implemented to reduce drawbacks of using demonstrations from a potentially sub-optimal reference policy. During training, we gradually reduce the degree of mixing demonstration sample paths to avoid the learned policy from being overly influenced by the demonstrations. Also when evaluating state value functions, we only use the experiences by the current policy and exclude those from the reference policy.

Using demonstrations to find a better solution of sequential decision making problem is not a new concept. Imitation learning has been studied extensively for single agent problems [21, 26, 34, 45].

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 2019, Montreal, Canada

https://doi.org/doi

Imitation learning requires a good reference policy or demonstrations. In the previous studies on single-agent problems, demonstrations for imitation learning are obtained typically from human experts behaving in simulated or physical environment. For MARL, we propose as an alternative to use solutions from MMDP or MPOMDP problems as a source of demonstrations.

Our proposed idea of using MMDP or MPOMDP solutions for imitation learning is motivated by a few reasons. First, we can always construct demonstrations from a relevant MMDP or MPOMDP model. Because a dec-POMDP model is the most general model among cooperative sequential decision making problems, it can always be reduced to an MMDP or MPOMDP model by assuming full state observability (MMDP) or joint observation (MPOMDP). This can be particularly useful when obtaining demonstrations from human experiments or other means is infeasible or costly. Second, MMDP or MPOMDP solutions act as a good reference policy. MMDP and MPOMDP models assume all available information is shared by the participating agents, making it a centralized decision problem. Since agents can use more information for decision making in the MMDP and MPOMDP environment, quality of the obtained policy is higher than a dec-POMDP policy obtained for the same system. Of course, we cannot directly use a MMDP or MPOMDP solution as a policy for the dec-POMDP problem because it requires information that is not available in the dec-POMDP environment. In this sense, MMDP and MPOMDP solutions can be considered as a *oracle* solution to the dec-POMDP problem. Being an oracle solution, we expect the MMDP and MPOMDP solutions to contain good traits for a dec-POMDP policy to learn.

2 RELATED WORK

Imitation learning is a method to find a policy using demonstrations. For example, behavior cloning is a supervised learning which aims to develop a policy that *clones* the demonstrations so that the resulting policy outputs the same actions chosen in the demonstrations. However, if the demonstrations are from a sub-optimal policy, which is often the case, the best policy obtained by behavior cloning is also sub-optimal. To overcome this problem, many recent studies propose to combine imitation learning and RL [21, 26, 34, 45]. A general approach in these studies is to update a neural network by simultaneously considering the objective functions of an RL algorithm and behavior cloning during the training phase. For example, [21] updates a value network for Deep Q-Network (DQN) [25] by using the weighted sum of two objective functions: temporal difference (TD) error for Q-learning and cross-entropy error for behavior cloning. [26, 34, 45] incorporate the loss function from behavior cloning when calculating policy gradient. These studies adopt mechanisms to avoid harmful learning from the sub-optimal demonstrations, such as adjusting the weights for adding loss values. In [26], instances of demonstrations are filtered out in gradient calculation if their selected actions are judged worse than the actions from the current policy. Note that the above references discuss imitation learning - RL approaches in a single-agent setting. While there exist a few studies that apply imitation learning to multi-agent problems [8, 22, 44], the imitation learning - RL approach in a multi-agent setting has not been well reported; to our knowledge, our work is the first study to adopt the imitation learning - RL approach to a dec-POMDP.

Many MARL algorithms have been developed by modifying well-known single agent RL algorithms to fit a multi-agent environment. One of the early approaches is an independent learning algorithm where an agent treats the other agents as its environment [33, 39]. This algorithm has a limitation that it does not stably find a joint policy solution because the environment is non-stationary as other agents learn a different policy during the training process. This limitation has been addressed in several studies on deep-MARL algorithms by sharing the parameters of network between agents [10, 14]. More recently, some researchers propose to modify the actor-critic (AC) algorithm by updating a value network with full state information [11, 24]. They use individual observations and actions to train a policy network which represent a decentralized policy. In these algorithms, agents learn their individual policy while indirectly considering other agents through the guidance of the critic which know full state information. In this paper, we also use the AC algorithm with a centralized critic as a baseline MARL algorithm.

The last piece of related literature is RL as a Rehearsal (RLaR). RLaR is a model-based MARL algorithm with two stages. In the first stage, it uses full state information to learn a value function; it solves an MMDP problem by Q-learning. Then, it transforms the value function learned during the first stage to establish the initial value function for the second stage. With the inital value functions, the algorithm finds a dec-POMDP policy by a Q-learning based MARL. This algorithm is similar to our proposed algorithm in that it uses MMDP to establish a starting point to solve a dec-POMDP problem. However, our study differs from RLaR in two aspects. We can use demonstrations from MPOMDP or any other reference policy as well as an MMDP policy. Also, our approach is more scalable than RLaR as it uses a deep neural network to approximate value functions.

3 BACKGROUND

3.1 Dec-POMDP

Dec-POMDP is defined as a tuple $< I, S, \{A_i\}, P, r, \{\Omega_i\}, O, h >$ [29]. I is a finite set of agents, S is a finite set of states and $\{A_i\}$ is a finite set of actions for each agent i. At every decision epoch t, each agent obtains individual observation $o_{i,t}$, and chooses an individual action $a_{i,t}$ to maximize a common goal. When a joint action $a_t = \{a_{0,t}, a_{1,t}, \ldots, a_{N,t}\}$ is selected, the system state transitions from s_t to s_{t+1}, following a transition probability function $P(s_{t+1}|s_t, a_t)$. $\{\Omega_i\}$ is a finite set of observations for each agent i. Observation function O defines the probability of observing joint observation $o_t = \{o_{0,t}, o_{1,t}, \ldots, o_{N,t}\}$ when the next state is s_{t+1} by joint action a_t: $O(o_t|a_t, s_{t+1})$. Because agents seek to maximize common reward, they share the same reward defined by reward function r which defines the reward earned when joint action a_t is taken at s_t: $r(s_t, a_t)$. We use h to indicate a decision horizon.

The objective of a dec-POMDP model is to find a jointly separable policy π that maximizes discounted expected reward:

$$\pi^* = argmax_{\pi} E_{\pi}[\Sigma_{t=0}^{h} \gamma^t r(s_t, a_t)] \quad (1)$$

γ is a discount factor. We simply express the expectation over random variables such as visited states and chosen actions under policy π by Eq. (1) and this convention will be used throughout the paper. Because an agent does not know the exact state of the system and other agents' action at each decision epoch, it chooses its action based on its own previous actions and observations. A policy for individual agent, π_i, is a function which stochastically maps the agent's action-observation history (AOH) to its individual action. We denote AOH by $\bar{\theta}_{i,t} = (a_{i,0}, o_{i,1}, \ldots, a_{i,t-1}, o_{i,t})$. $\pi_i(a_{i,t}|\bar{\theta}_{i,t})$ is the probability of choosing $a_{i,t}$ given $\bar{\theta}_{i,t}$. Then, the jointly separable policy is the product of the individual policies: $\pi(a_t|\bar{\theta}_t) = \Pi_i \pi_i(a_{i,t}|\bar{\theta}_{i,t})$.

Recurrent Neural Network (RNN) is a general structure to represent a dec-POMDP policy when using deep-RL [10, 11, 24, 33]. RNN is an effective architecture to consider temporal dependency between input and output, and as such it is more suitable for a POMDP environment where an observation sequence is used as an input. A hidden state of an RNN can be interpreted as an approximation of a full-length individual AOH during execution.

3.2 Policy Gradient Algorithm

RL algorithms can be classified into valued-based algorithms and policy-based algorithms depending on which function is approximated. Value-based algorithms approximate a state-value function, $V^{\pi}(s_t) = E_{\pi}[\Sigma_{l=0}^{h-t} \gamma^l r(s_{t+l}, a_{t+l})|s_t]$, or q-function, $Q^{\pi}(s_t, a_t) = E_{\pi}[\Sigma_{l=0}^{h-t} \gamma^l r(s_{t+l}, a_{t+l})|s_t, a_t]$, which are used to find an action that maximizes the value of each state. Policy-based algorithms learn a policy function which defines the probability of choosing an action given recognized information.

Policy gradient algorithms aim to maximize $J(w) = E_{\pi_w}[r_1 + \gamma^1 r_2 + \cdots + \gamma^{h-1} r_h]$, by updating policy parameter w in the direction of policy gradient. Policy gradient for MDP problem can be expressed as follows:

$$\nabla_w J(w) = E_{\pi_w}[Q^{\pi_w}(s_t, a_t) \nabla_w log \pi_w(a_t|s_t)] \tag{2}$$

Because the exact Q-value for π_w is not easy to obtain, we substitutes $Q^{\pi_w}(s_t, a_t)$ with other values that we can easily estimate. REINFORCE algorithm is the most basic policy gradient algorithm. It uses a sample return to substitute Q-value; that is, we replace $Q^{\pi_w}(s_t, a_t)$ by $g_t = r_t + \gamma^1 r_{t+1} + \cdots + \gamma^{h-t} r_h$ [43]. This sample return is an unbiased estimator of $Q^{\pi_w}(s_t, a_t)$, but the variance is too large to stably find a policy solution.

AC algorithms learn both value function and policy function [20]. Actors update parameter w for a policy function by interacting with the environment according to the actions chosen under π_w. Critic updates parameter v for a value function to guide the actors by evaluating the value of an action. To do this, critic estimates $Q^{\pi_w}(s_t, a_t)$ by computing $Q_v^{\pi_w}(s_t, a_t)$ or $r(s_t, a_t) + V_v^{\pi_w}(s_t)$, both of which are biased estimators of $Q^{\pi_w}(s_t, a_t)$ but have a smaller variance than sample returns. We can further reduce the variance of these estimators by a control variates method, subtracting a baseline value. The most common choice for the baseline value is a state-value function because it is highly correlated with the Q-function. As a result, we have an advantage function, $A^{\pi}(s_t, a_t) = Q^{\pi}(s_t, a_t) - V^{\pi}(s_t)$, which is a relative value of the selected action.

4 METHOD

Our goal is to improve the performance of a baseline MARL algorithm by effectively using demonstrations from a reference policy. In this paper, we use a multi-agent AC with a centralized critic as our baseline MARL algorithm. Our algorithm is conceptually identical to central-V [11] and PS-A3C [14] but with some differences in its implementation details. For simplicity's sake, we focus on an undiscounted, finite horizon dec-POMDP problem with discrete actions. Nevertheless, we believe the proposed idea of mixing demonstrations from a centralized policy is still applicable to infinite horizon and discounted problems with continuous action space.

We describe the baseline MARL algorithm in section 4.1 and explain how to mix demonstrations in the baseline algorithm in section 4.2.

4.1 Multi-agent Actor Critic with Centralized Critic

[11, 24] recently show that a centralized critic, that uses global information, improves the performance of RL in a decentralized environment. It gives the full information on the current state to the critic – hence termed as a centralized critic –, with which the critic learns a value network to estimate state-value $V_v(s)$. Since the critic plays a role only during the training phase, we can allow a critic to use the full state information even in the decentralized environment.

Our algorithm adopts this approach, multi-agent AC with a centralized critic. We model the critic by using a feed-forward network which takes the current state, s, as its input and predicts state-value $V_v(s)$. A feed-forward network is a sufficient architecture for a centralized critic since the state transition is conditionally independent of the previous state if the most recent state is given. We train our value network based on a TD(λ) method, which aims to minimize the following loss function:

$$L(v) = \Sigma_{t=0}^{h} (G_t^{\lambda} - V_v^{\pi_w}(s_t))^2 \tag{3}$$

where $G_t^{\lambda} = (1-\lambda)\Sigma_{n=1}^{h-t} \lambda^{n-1} G_t^{(n)}$ and $G_t^{(n)} = \Sigma_{l=1}^{n} \gamma^{(l-1)} r_{t+l} + \gamma^n V_v^{\pi_w}(s_{t+n})$ which is the n-step return. The estimated value function is then used to calculate a policy gradient.

For the actor, we design a policy network to take current observations and the agents' indices as its inputs and to output the probability of selecting individual actions. An RNN is used to construct the policy network. Input to the RNN consists only of current observations rather than the entire observation history and also it does not include previous actions. Although we feed the RNN only with the current observation at each time step, a hidden state in the RNN contains approximate memory of the individual observation history. Hidden states of the RNN are updated at every time step as a function of the previous hidden states and current inputs: $h_{i,t} = f(h_{i,t-1}, o_{i,t}, i)$. Using only the observations in training RNN is sufficient for our model; [31] shows that there exists at least one optimal policy which maps observation history into an action for finite horizon dec-POMDP problems.

Instead of training an individual RNN for each agent, we build a single RNN by parameter sharing and use it for all the agents. It is known that training a single RNN by parameter sharing is

Table 1: Multi-agent AC with Centralized Critic using Demonstration Buffer

```
Algorithm 1.
Input:$(\bar{s}_h, \bar{\theta}_h, \bar{r}_h) \in B_D$
Initialize $w$, $v$ and set $\rho_D = 1$
for $k \leftarrow 0, 1, \ldots$ do
    $n_D \sim Binomial(N, \rho_D)$
    Sample $(N - n_D)$ episodes from current policy, $\pi_w$
        $\tau^n \sim \pi_w$ for $n = 1, \ldots, (N - n_D)$
    Update $v$ using Eq. (3) with learning rate $\alpha_v$
        $v \leftarrow v - \alpha_v \nabla_v \Sigma_{n=1}^{N-n_D} \Sigma_{(s_t, r_t) \in \tau^n} (G_t^\lambda - V_v(s_t))^2$
    Draw $n_D$ episodes from $B_D$
        $\tau^n \sim B_D$ for $n = (N - n_D + 1), \ldots, N$
    Update $w$ using Eq. (5) with learning rate $\alpha_w$
        $w \leftarrow w + \alpha_w \Sigma_{n=1}^{N} \Sigma_{(s_t, \bar{\theta}_t, r_t) \in \tau^n} A_t^{GAE} \nabla_w log\pi_w$
    Decay mixing probability
        $\rho_D \leftarrow \rho_D \times \eta$
```

better than training multiple networks independently for individual agents [14]. To allow different agents to learn different policies from the single RNN, we use an agent index as an identifier. Finally, an individual policy function, $\pi_{i,w}(a_{i,t})$, is obtained after output values of the RNN pass through an additional fully connected layer.

A policy gradient for a dec-POMDP model is similar to the policy gradient for MDP in Eq. (2). In a dec-POMDP, policy π_w is a jointly separable policy, which is a product of individual policies, $\pi_{i,w}$: $\pi_w(a_t|\bar{\theta}_t) = \Pi_i \pi_{i,w}(a_{i,t}|\bar{\theta}_{i,t})$. We calculate the policy gradient based on the generalized advantage estimator (GAE) [36]:

$$\nabla_w J_{dec}(w) = E_{\pi_w}[A_t^{GAE} \nabla_w log\pi_w(a_t|\bar{\theta}_t)] \quad (4)$$

where $A_t^{GAE} = \Sigma_{l=0}^{h-t} \lambda_{GAE}^l \{r_t + V_v^{\pi_w}(s_{t+1}) - V_v^{\pi_w}(s_t)\}$. GAE attempts to compromise between the variance and bias of the estimator by using λ_{GAE}.

4.2 Mixing Demonstrations from Centralized Policy

The main idea of our proposed approach is to augment the baseline MARL algorithm, described in the previous section, by mixing demonstrations from a reference policy. As explained in §1, we use a centralized policy – a policy solution of MMDP or MPOMDP model – to establish a reference policy to use in the baseline algorithm. In this section, we describe how we mix the demonstration samples into the algorithm. Our method is summarized in Table 1 and is explained in detail below.

A demonstration used in our algorithm is a trajectory, τ, that consists of state sequence, action-observation history (AOH) and reward sequence: $\tau = (\bar{s}_h, \bar{\theta}_h, \bar{r}_h)$ where $\bar{s}_h = (s_1, s_2, \ldots, s_h)$ and $\bar{r}_h = (r_1, r_2, \ldots, r_h)$. Each trajectory is a record of the action-observation history, the sequence of states visited and the obtained rewards, when the centralized policy is followed. Note that we have in our demonstration a reward sequence. Typical applications of imitation learning assume the true reward function is unknown or reward sequence data is not given [21, 22, 26, 34, 44]. In our case, on the other hand, immediate rewards can be recorded when collecting demonstrations from following a centralized policy in the simulation environment. We also want to emphasize that, because the demonstrations have the same format, trajectory τ, as the sampled episodes from a current policy in the baseline MARL algorithm, it is possible to directly use the demonstrations when mixing them in the baseline algorithm.

We collect demonstrations from the centralized policy in advance and store them in a demonstration buffer, B_D: $\tau \in B_D$. A demonstration buffer is often used in imitation learning on single agent problems [21, 26]. At each training step, we sample episodes from a current policy and also draw demonstrations from the demonstration buffer. The number of demonstrations used in each training step follows a binomial distribution (N, ρ_D) where N is the minibatch size and ρ_D is the probability of mixing.

We set the mixing probability ρ_D as an exponential function of the training steps. At each training step k, the mixing probability is decreased by multiplying common ratio, $\eta \in [0, 1]$; that is, $\rho_D(k) = \eta^k$. By using an exponentially decaying function for ρ_D, we let the centralized policy drive the learning at the beginning and, as the training progresses, put higher weights on the experience gained from the current policy. The gradual reduction of the demonstrations' influence is desirable because the actions taken by the centralized policy is sub-optimal in the decentralized environment. As a sanity check, we have empirically tested alternative mixing probabilities, a step function and a constant function for the benchmark problems used in this paper. The experimental results indicate the exponentially decaying function works the best.

The baseline MARL algorithm – multi-agent actor critic with centralized critic – updates a policy and value network based on sampled episodes. In the algorithm, we mix in the demonstrations only for updating the policy network. Reason for not using the demonstrations when updating the value network is to prevent overestimation of the value functions. The role of the value network is to accurately predict the expected return from a current state when agents choose actions according to π_w. Recall that a demonstration is obtained from a centralized policy, and a sample return calculated from it is an upper bound rather than a realistic value achievable by a dec-POMDP policy. Thus, if we update the value network using the reward sequence of a demonstration as a target, it can result in an overestimate of the value function. It is known that overestimation of a state-value in Deep-RL algorithms can degrade the performance of the algorithm [40], and therefore, we do not use demonstrations when updating the value network.

In updating the policy network, we use the reward sequence from demonstrations to compute GAE in the policy gradient. Specifically, the policy gradient is the sum of the gradient value obtained from the sampled episode ($\nabla_w J_{dec}$) and the gradient value obtained from the demonstrations:

$$\nabla_w J_{mix}(w) = \nabla_w J_{dec}(w) + E_{\tau \sim B_D}[A_t^{GAE} \nabla_w log\pi_w] \quad (5)$$

Expectation of Eq. (5) is computed using minibatch.

5 EXPERIMENTS

5.1 Experimental Setup

We assess the performance of our algorithm on three benchmark problems commonly tested in dec-POMDP research: Mars rover [2],

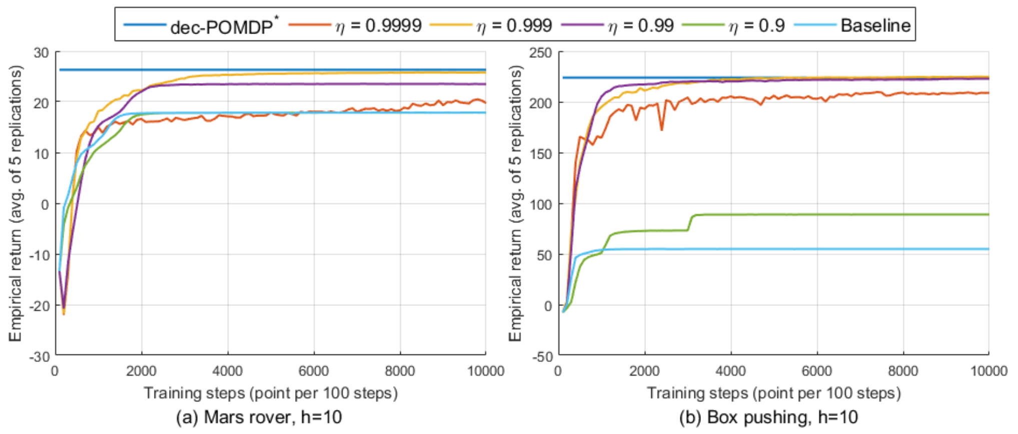

Figure 1: Improvement of baseline algorithm through MMDP demonstrations using different η

cooperative box pushing [37] and dec-tiger [27]. Optimal solutions are known for these benchmark problems [9].

In the Mars rover problem, two rovers perform their mission tasks while exploring Mars. Two rovers should cooperate on some tasks to successfully finish their mission. The goal of the cooperative box pushing problem is to control two agents when there are small boxes that can be pushed by individual agent and a big box that can only be moved with two agents pushing in the same direction. They get a higher reward for pushing the big box over the goal line. In these two problems, agents earn a smaller reward (or penalty) if they act without cooperation. In the dec-tiger problem, a tiger and a treasure are located behind of each door, and two agents work together to figure out the correct door to open (the one behind which is the treasure). In this problem, estimating true state through observations (hearing the sound from each door) is more important than the other two problems. For problem formulation, we refer to the dec-POMDP models provided in the Multiagent Decision Process (MADP) toolbox [30].

By reducing the dec-POMDP models, we construct an MMDP and MPOMDP model for the three problems. Then we find a reference policy solution for these problems which we later use to generate demonstrations for the MARL algorithm. It is known that solution methods to solve MDP models can be used to solve MMDP models [5], and we use a value iteration method to obtain an optimal MMDP solution. For MPOMDP models,we use the incremental pruning method [6] implemented in the 'pomdp-solve' software [7].

It turns out that the pomdp-solve cannot solve the Mars rover problem and box pushing problem. As such, we obtain the MMDP solutions for these problems and use them as a reference policy. With the solutions, we examine whether demonstrations from the MMDP solution enhance learning of a decentralized policy. For the dec-tiger problem, both MMDP and MPOMDP solutions are obtained. These solutions allow us to compare which of the two reference policies work better as a source of demonstration in the proposed MARL algorithm.

In all experiments, a value network is modeled by two ReLU layers with 64 units per layer, and we use a fully connected layer as the final layer. We construct a policy network by using an LSTM [17] layer with 64 memory cells and a fully connected layer is used as well as a final layer. Actions are selected based on Gumbel-Softmax estimator [18]. We set λ and λ_{GAE} to 0.95 for TD(λ) and GAE respectively. Both value and policy networks are trained with Adam optimizer [19] with the base learning rate at 0.001. The size of minibatch is 32, and we train for 10,000 steps per experiment. Under these settings, we obtain five policy solutions using our algorithm for each benchmark problem. The interim policy networks are saved every 100 training steps. Finally, we test each of the five policy solutions for 1,000 episodes while controlling their random seeds to ensure the five policy solutions are tested under the same condition.

5.2 Effectiveness of Demonstrations from MMDP Policy

As mentioned earlier, we use an MMDP solution as a reference policy to generate demonstrations for the Mars rover and box pushing problem. Figure 1 show the return values from the baseline algorithm (without mixing) and the proposed, demonstration-MARL algorithm with various values of η for the Mars rover and box pushing problem at horizon $h = 10$. In both Figure 1(a) and (b), we see that mixing the demonstrations from the MMDP policy solution during the training process improve the baseline MARL algorithm. For both problems, the baseline MARL algorithm converges to a return value far below the optimal value. Our demonstration-MARL algorithm, in all η settings, converges to a return value higher than the baseline algorithm. This suggests that mixing the demonstrations from MMDP reference policy does improve the performance of baseline algorithm.

While mixing the demonstrations delivers better policy solutions than the baseline MMDP, the experimental results indicate more mixing is not necessarily better. Recall that a high (low) value of η means more (less) mixing for a longer (shorter) training steps. In both cases, we see that the best result is obtained when η is 0.999; mixing too much (η=0.9999) or too little (η=0.9) does not help improving the quality of the solution. This is particularly evident in Figure 1(a). The plot for $\eta = 0.9$, which means we mix a small number of demonstrations and reduce the mixing quickly,

Table 2: Performance of learned policy by using MMDP demonstrations for various horizon length of box pushing problem. $\eta = 0.999$ **and parentheses indicates 95% confidence interval**

Horizon	dec-POMDP*	w/o demo.	with demo.
4	98.59	18.32 (0.28)	99.86 (3.14)
6	120.67	36.78 (0.38)	123.58 (3.56)
8	191.22	52.98 (0.44)	191.86 (4.35)
10	223.74	54.79 (0.49)	226.40 (4.74)

coincides with the baseline algorithm curve. Our interpretation is that the algorithm fails to take advantage of the useful information contained in the demonstrations. Quality of the solution improves as η increases to 0.99 and 0.999, but then it drops down when $\eta = 0.9999$. It seems that mixing too many demonstrations for too long causes negative effects, hence not a good strategy either. One possible explanation is that the reference policy, which is a centralized policy, is not a true optimal target and thus it is not desirable to overly pursue to imitate the demonstrations; it can hamper learning from its own experiences and lead to insufficient exploration of other possible decentralized policies.

In Table 2, we have the results at different lengths of decision horizon for the box pushing problem. Table 2 shows the return values from the best policy solution among the five solutions obtained by the baseline algorithm and the demonstration-MARL algorithm (Full results from all five policy solutions for both Mars rover and box pushing problems are presented in a table in the Appendix). Again, we see that the baseline algorithm ends up with a significantly sub-optimal policy solution; their return values are far lower than the known optimal values (dec-POMDP*). In fact, the return values are close to the reward values when the two small boxes are successfully pushed behind the goal line, and we suspect that in the baseline algorithm, both agents easily fall into a local optimal policy where they move small boxes to small rewards. Such behavior is indeed observed in the simulation results. We believe there is a sound explanation for this local optimal solution. Since there is no prior knowledge or experience about the system, it is difficult for the baseline MARL algorithm to discover that they can earn a larger reward when pushing the large box together to the goal line; an experience of successfully moving the large box over the goal line rarely occurs by chance. On the other hand, moving small boxes and getting a reward, albeit small, is more frequently experienced during exploration. If either agent continues to make actions to push the small box (because it does not know it can earn a large reward by pushing the large box together with the other agent), the other agent would receive a negative reward when it takes exploratory actions to push the large box. This is a typical challenge in MARL algorithms.

When we mix demonstrations from the centralized policy, the algorithm finds a better policy with a guidance from the episodes where the agents have gotten a high reward by moving the large box together. The MMDP policy solution generates demonstrations where the agents receive higher reward by moving the large box in collaboration. These demonstrations contain the information on how they should behave to push the large box even if these actions cannot be completely imitated in the decentralized environment. This gives an opportunity to overcome the hurdle of falling into a local optimal policy in the absence of demonstrations.

5.3 Demonstrations from MPOMDP Policy

Previous section demonstrates that using an MMDP solution as a reference policy can improve the performance of the baseline MARL. In this section, we discuss a case, dec-tiger problem in particular, where an MMDP solution is not a suitable reference policy. For a dec-tiger problem, using an MMDP solution to generate demonstrations leads to a poor decentralized policy solution. Table 3 shows that augmenting the baseline MARL by mixing the demonstrations from an MMDP solution, in fact, results in a worse policy solution than the baseline MARL solution. Considering the nature of the dec-tiger problem and the way an MMDP model is concocted, this result is rather expected. In the dec-tiger setting, the decision problem becomes trivial if we know the true state of the system; if we know behind which door a treasure sits, then all we need to know is to open that door and there is no need to pursue additional observation by listening to the sound. This is the case with the MMDP model where we assume the full knowledge about the system state. But then, the listen action to gather more information carries a critical importance in the dec-POMDP environment. Thus, demonstrations from the MMDP solution does not help the agents. Worse yet, mixing the demonstrations causes the algorithm to search the wrong solution space by feeding the episodes irrelevant to learning a good decentralized policy.

As a remedy to this problem, we adopt an MPOMDP solution as a reference policy for our algorithm. Table 3 shows that using the demonstrations from the MPOMDP solution does significantly improve the policy solution compared with the baseline algorithm. When augmented by the MPOMDP demonstrations, the return values from the policy solution get much closer to the optimal value. A possible reason for this improvement is that the MPOMDP model assume the same amount of information about the system state as the original dec-POMDP environment, except that the information is shared by the agents. In the MPOMDP solution policy, the two agents choose the listen action to estimate the true state and opens the door that is most likely to have a treasure. The degree to which the agents would choose the listen action will be less than the dec-POMDP policy because in the MPOMDP model, the two agents share the information after the listen action. Nevertheless, demonstrations from the MPOMDP solution contain valuable information for the decentralized agents and provide some opportunity for them to learn about the value of the listen actions

While learning about the value of the listen action is an important reason behind the performance of MPOMDP demonstrations, there is another reason. This is related to the poor performance of the solution from the baseline MARL algorithm, shown in the third column of Table 3. When we solve the problem without using demonstration, the policy solution converges to a local optimal. Because opening a door behind which sits a tiger incurs a high penalty, much greater than the reward of opening the treasure-door, the baseline MARL algorithm produces an extremely conservative policy; the return values from the baseline algorithm policy, shown

Table 3: Comparison between the results of using MPOMDP and MMDP solution for dec-tiger problem. $\eta = 0.999$ **and parentheses indicates 95% confidence interval**

Horizon	dec-POMDP*	w/o demo.	with MMDP demo.	with MPOMDP demo.
4	4.80	-8.00 (–)	-58.63 (4.24)	4.31 (0.86)
6	10.38	-12.00 (–)	4.94 (0.91)	5.63 (0.65)
8	12.22	-16.00 (–)	-115.35 (6.34)	9.24 (1.13)
10	15.18	-20.00 (–)	-145.18 (7.04)	10.62 (0.97)

in Table 3, indicate that the agents always choose to listen and do not take the risk of opening any door. Demonstrations from the MPOMDP policy solution, on the other hand, reveals the knowledge that opening a door, if the correct door, returns a reward for the action. Thus, by introducing demonstrations from the MPOMDP solution, the agents learns that they can earn positive reward once they attain some confidence by several listen actions.

Finally, we provide our conjecture as to why the dec-tiger problem requires MPOMDP solution as its reference policy when MMDP solutions work just fine for the Mars rover and box pushing problems. First, we note that there exists a much larger gap for the dec-tiger problem between the optimal dec-POMDP solution's value and the optimal MMDP value than the other two problems. Because the main difference between an MMDP model and the corresponding dec-POMDP model is the amount of information available to the decision maker, this gap can be thought of as an indicator for the advantage of having access to the full state observation. In case of $h = 10$, this gap is 8.8% for Mars rover and 11.1% for the box pushing problem. For the dec-tiger problem, the gap is far greater at 1217.2%. The extremely large gap in the dec-tiger problem suggests that the agents must perform actions to acquire additional observations to estimate the true state of the system. Second, in the dec-tiger problem, there is no second chance to make up one bad decision by a next action. Because the dec-tiger problem is reset to an initial state once any door is opened, there is a greater risk of incorrectly estimating true state; agents do not get a second chance to estimate the original state. A reference policy from an MMDP solution seems less beneficial to learning a decentralized policy if learning about the true state is relatively more important than learning about the cooperation of actions between agents. This is the case with the dec-tiger problem, and a possible reason why the MPOMDP solution is a better choice to generate its demonstrations.

6 CONCLUSION

In this study, we demonstrate that the imitation learning technique can be incorporated to reinforcement learning to solve a multi-agent problems. While combining the imitation learning and reinforcement learning has been demonstrated in a single agent setting, it has not been discussed in the multi-agent setting in the literature. In particular we examine dec-POMDP problems to show how imitation learning can be implemented in a multi-agent reinforcement learning. Our proposed method augments a cooperative multi-agent reinforcement learning algorithm (MARL), actor-critic with a centralized critic, by mixing demonstrations from a reference policy. We show that a reasonable reference policy for a dec-POMDP problem can be obtained from an MMDP or MPOMDP model reduced form the original dec-POMDP model, which can be particularly useful when obtaining demonstrations from human experiments or other means is infeasible or costly. Experiment results on three well-known dec-POMDP benchmark problems demonstrate that the proposed method improves the performance of the baseline algorithm without demonstrations.

Recent approaches to MARL shifts from extending proven single-agent RL algorithms to multi-agent environment to developing MARL by using the characteristics of the multi-agent environment [13, 32]. We believe that our idea of combining imitation learning with MARL belongs to these recent research efforts. Areas of future research in this direction include simultaneous learning of a centralized policy (as a reference policy) and a decentralized policy. Simultaneous learning will possibly improve off-policy learning through an importance sampling technique by using the policy function of a centralized policy.

A FULL RESULTS FROM ALL FIVE POLICY SOLUTIONS

The following results (Table 4, Table 5, Table 6) provide test results of five policy solutions for each setting.

ACKNOWLEDGMENTS

The authors would also like to thank the anonymous referees for their valuable comments and helpful suggestions.

REFERENCES

[1] Nezih Altay and Walter G Green III. 2006. OR/MS research in disaster operations management. *European journal of operational research* 175, 1 (2006), 475–493.

[2] Christopher Amato and Shlomo Zilberstein. 2009. Achieving goals in decentralized POMDPs. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, Budapest, Hungary, 593–600.

[3] Raghav Aras, Alain Dutech, and François Charpillet. 2007. Mixed Integer Linear Programming for Exact Finite-Horizon Planning in Decentralized POMDPs. In *Proceedings of the Seventeenth International Conference on International Conference on Automated Planning and Scheduling*. AAAI Press, Providence, Rhode Island, USA, 18–25.

[4] Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. 2002. The complexity of decentralized control of Markov decision processes. *Mathematics of operations research* 27, 4 (2002), 819–840.

[5] Craig Boutilier. 1999. Sequential optimality and coordination in multiagent systems. In *Proceedings of the 16th international joint conference on Artifical intelligence*, Vol. 99. Morgan Kaufmann Publishers Inc., Stockholm, Sweden, 478–485.

[6] Anthony Cassandra, Michael L. Littman, and Nevin L. Zhang. 1997. Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes. In *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., Providence, Rhode Island, USA, 54–61.

[7] Anthony R. Cassandra. 2015. pomdp-solve. http://www.pomdp.org/. (May 2015).

Table 4: Experiment results of five policy solutions for the Mars rover problem

Horizon	4	6	8	10
dec-POMDP*	10.18	18.62	22.47	26.31
Baseline: run 1	7.138 (0.06)	10.82 (0.07)	14.29 (0.08)	17.87 (0.10)
Baseline: run 2	7.138 (0.06)	10.68 (0.07)	14.28 (0.09)	17.80 (0.10)
Baseline: run 3	7.138 (0.06)	10.68 (0.07)	14.30 (0.08)	17.87 (0.10)
Baseline: run 4	7.126 (0.06)	10.69 (0.07)	14.29 (0.08)	17.87 (0.10)
Baseline: run 5	7.138 (0.06)	10.68 (0.07)	14.29 (0.08)	17.86 (0.10)
with MMDP demo.: run 1	10.24 (0.16)	18.42 (0.33)	22.15 (0.31)	25.67 (0.42)
with MMDP demo.: run 2	10.24 (0.16)	18.42 (0.33)	22.16 (0.31)	25.89 (0.42)
with MMDP demo.: run 3	10.24 (0.16)	18.42 (0.32)	22.15 (0.31)	25.71 (0.42)
with MMDP demo.: run 4	10.25 (0.15)	18.42 (0.32)	22.16 (0.31)	25.87 (0.42)
with MMDP demo.: run 5	10.25 (0.15)	18.42 (0.32)	22.15 (0.31)	25.67 (0.42)

Table 5: Experiment results of five policy solutions for the box pushing problem

Horizon	4	6	8	10
dec-POMDP*	98.59	120.67	191.22	223.74
Baseline: run 1	18.28 (0.27)	36.71 (0.38)	52.97 (0.44)	54.78 (0.48)
Baseline: run 2	18.28 (0.27)	36.78 (0.38)	52.98 (0.44)	54.79 (0.49)
Baseline: run 3	18.32 (0.28)	36.78 (0.38)	52.97 (0.44)	54.79 (0.49)
Baseline: run 4	18.28 (0.27)	36.78 (0.38)	52.98 (0.44)	54.79 (0.49)
Baseline: run 5	18.28 (0.27)	36.78 (0.38)	52.98 (0.44)	54.79 (0.49)
with MMDP demo.: run 1	99.76 (3.14)	121.26 (3.61)	191.60 (4.34)	226.40 (4.74)
with MMDP demo.: run 2	99.36 (3.18)	121.60 (3.60)	191.86 (4.35)	224.10 (4.86)
with MMDP demo.: run 3	99.86 (3.14)	123.58 (3.56)	190.76 (4.37)	223.98 (4.78)
with MMDP demo.: run 4	99.85 (3.15)	123.48 (3.56)	191.27 (4.34)	225.27 (4.78)
with MMDP demo.: run 5	99.86 (3.14)	122.56 (3.60)	191.59 (4.35)	223.72 (4.80)

Table 6: Experiment results of five policy solutions for the dec-tiger problem

Horizon	4	6	8	10
dec-POMDP*	4.80	10.38	12.22	15.18
Baseline: run 1	-8 (–)	-12 (–)	-16 (–)	-20 (–)
Baseline: run 2	-8 (–)	-12 (–)	-16 (–)	-20 (–)
Baseline: run 3	-8 (–)	-12 (–)	-16 (–)	-20 (–)
Baseline: run 4	-8 (–)	-12 (–)	-16 (–)	-20 (–)
Baseline: run 5	-8 (–)	-12 (–)	-16 (–)	-20 (–)
with MMDP demo.: run 1	-64.47 (4.32)	-92.53 (5.38)	-117.70 (6.13)	-156.23 (7.06)
with MMDP demo.: run 2	-64.25 (4.30)	4.94 (0.91)	-118.26 (6.32)	-145.18 (7.04)
with MMDP demo.: run 3	-58.63 (4.24)	-94.72 (5.34)	-115.35 (6.34)	-155.55 (7.05)
with MMDP demo.: run 4	-64.25 (4.30)	-94.82 (5.34)	-120.01 (6.28)	-145.18 (7.04)
with MMDP demo.: run 5	-64.42 (4.32)	-86.1 (5.35)	-125.40 (6.31)	-145.18 (7.04)
with MPOMDP demo.: run 1	4.28 (0.86)	5.43 (0.69)	9.10 (1.13)	9.54 (1.28)
with MPOMDP demo.: run 2	4.30 (0.86)	4.94 (0.91)	9.13 (1.15)	10.62 (0.98)
with MPOMDP demo.: run 3	4.31 (0.86)	5.50 (0.69)	9.24 (1.13)	9.94 (1.18)
with MPOMDP demo.: run 4	4.28 (0.86)	5.63 (0.65)	6.31 (1.13)	10.61 (1.00)
with MPOMDP demo.: run 5	-1.80 (0.80)	5.47 (0.69)	6.38 (0.86)	10.46 (1.01)

[8] Sonia Chernova and Manuela Veloso. 2007. Multiagent collaborative task learning through imitation. In *Proceedings of the fourth International Symposium on Imitation in Animals and Artifacts*. Newcastle upon Tyne, UK, 74–79.

[9] Jilles Steeve Dibangoye, Christopher Amato, Olivier Buffet, and François Charpillet. 2016. Optimally solving Dec-POMDPs as continuous-state MDPs. *Journal of Artificial Intelligence Research* 55 (2016), 443–497.

[10] Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. 2016. Learning to Communicate to Solve Riddles with Deep Distributed Recurrent Q-Networks. *CoRR* abs/1602.02672 (2016).

[11] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual Multi-Agent Policy Gradients. In *AAAI Conference on Artificial Intelligence*. AAAI Press, New Orleans, Louisiana, USA, 2974–2982.

[12] Jakob N. Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip H. S. Torr, Pushmeet Kohli, and Shimon Whiteson. 2017. Stabilising Experience Replay for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70. PMLR, Sydney, Australia, 1146–1155.

[13] J. N. Foerster, C. A. Schroeder de Witt, G. Farquhar, P. H. S. Torr, W. Boehmer, and S. Whiteson. 2018. Multi-Agent Common Knowledge Reinforcement Learning. *ArXiv e-prints* (Oct. 2018). arXiv:cs.MA/1810.11702

[14] Jayesh K. Gupta, Maxim Egorov, and Mykel Kochenderfer. 2017. Cooperative multi-agent control using deep reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems*. Springer, Sao Paulo, Brazil, 66–83.

[15] Eric A Hansen, Daniel S Bernstein, and Shlomo Zilberstein. 2004. Dynamic programming for partially observable stochastic games. In *Proceedings of the 19th national conference on Artificial intelligence*. AAAI Press, San Jose, California, USA, 709–715.

[16] Steven R. Hirshorn, Linda D. Voss, and Linda K. Bromley. 2017. *NASA Systems Engineering Handbook*. MIT Research Lab Technical Report NASA/SP-2016-6105 Rev 2. National Aeronautics and Systems Administration.

[17] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[18] E. Jang, S. Gu, and B. Poole. 2016. Categorical Reparameterization with Gumbel-Softmax. *ArXiv e-prints* (Nov. 2016). arXiv:stat.ML/1611.01144

[19] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*. Ithaca, San Diego, USA.

[20] Vijay R. Konda and John N. Tsitsiklis. 2000. Actor-critic algorithms. In *Advances in Neural Information Processing Systems 12*. MIT Press, denver, Colorado, 1008–1014.

[21] Aravind S Lakshminarayanan, Sherjil Ozair, and Yoshua Bengio. 2016. Reinforcement learning with few expert demonstrations. In *NIPS Workshop on Deep Learning for Action and Interaction*. Barcelona, Spain.

[22] Hoang M Le, Yisong Yue, Peter Carr, and Patrick Lucey. 2017. Coordinated Multi-Agent Imitation Learning. In *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70. PMLR, Sydney, Australia, 1995–2003.

[23] J-H Lee and C-O Kim. 2008. Multi-agent systems applications in manufacturing systems and supply chain management: a review paper. *International Journal of Production Research* 46, 1 (2008), 233–265.

[24] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., Long beach, California, USA, 6379–6390.

[25] 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. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[26] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. 2018. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, Brisbane, QLD, Australia, 6292–6299.

[27] Ranjit Nair, Milind Tambe, Makoto Yokoo, David Pynadath, and Stacy Marsella. 2003. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proceedings of the 18th international joint conference on Artificial intelligence*. Morgan Kaufmann Publishers Inc., Acapulco, Mexico, 705–711.

[28] Frans Adriaan Oliehoek. 2013. Sufficient Plan-Time Statistics for Decentralized POMDPs. In *Proceedings of the 23rd international joint conference on Artificial Intelligence*. AAAI Press, Beijing, China, 302–308.

[29] Frans A Oliehoek and Christopher Amato. 2016. *A concise introduction to decentralized POMDPs* (1st ed.). Springer.

[30] Frans A Oliehoek, Matthijs TJ Spaan, Bas Terwijn, Philipp Robbel, and João V Messias. 2017. The MADP toolbox: an open source library for planning and learning in (multi-) agent systems. *The Journal of Machine Learning Research* 18, 1 (2017), 3112–3116.

[31] Frans A Oliehoek, Matthijs TJ Spaan, and Nikos Vlassis. 2008. Optimal and approximate Q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research* 32 (2008), 289–353.

[32] S. Omidshafiei, D.-K. Kim, M. Liu, G. Tesauro, M. Riemer, C. Amato, M. Campbell, and J. P. How. 2018. Learning to Teach in Cooperative Multiagent Reinforcement Learning. *ArXiv e-prints* (May 2018). arXiv:cs.MA/1805.07830

[33] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P How, and John Vian. 2017. Deep Decentralized Multi-task Multi-Agent Reinforcement Learning under Partial Observability. In *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70. PMLR, Sydney, Australia, 2681–2690.

[34] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. 2018. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. In *Proceedings of Robotics: Science and Systems*. Pittsburgh, Pennsylvania, USA. https://doi.org/10.15607/RSS.2018.XIV.049

[35] Sarvapali D Ramchurn, Feng Wu, Wenchao Jiang, Joel E Fischer, Steve Reece, Stephen Roberts, Tom Rodden, Chris Greenhalgh, and Nicholas R Jennings. 2016. Human–agent collaboration for disaster response. *Autonomous Agents and Multi-Agent Systems* 30, 1 (2016), 82–111.

[36] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2016. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In *International Conference on Learning Representations*. San Juan, Puerto Rico.

[37] Sven Seuken and Shlomo Zilberstein. 2007. Improved memory-bounded dynamic programming for decentralized POMDPs. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*. AUAI Press, Vancouver, BC, Votananda, 344–351.

[38] Daniel Szer, François Charpillet, and Shlomo Zilberstein. 2005. MAA*: a heuristic search algorithm for solving decentralized POMDPs. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*. AUAI Press, Edinburgh, Scotland, 576–583.

[39] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. 2017. Multiagent cooperation and competition with deep reinforcement learning. *PloS one* 12, 4 (2017), e0172395.

[40] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep Reinforcement Learning with Double Q-Learning.. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press, Phoenix, Arizona, USA, 2094–2100.

[41] Ruben Van Parys, Maarten Verbandt, Marcus Kotzé, Jan Swevers, Herman Bruyninckx, Johan Philips, and Goele Pipeleers. 2018. Flexible Multi-Agent System for Distributed Coordination, Transportation & Localisation. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, Stockholm, Sweden, 1832–1834.

[42] Shiyong Wang, Jiafu Wan, Daqiang Zhang, Di Li, and Chunhua Zhang. 2016. Towards smart factory for industry 4.0: a self-organized multi-agent system with big data based feedback and coordination. *Computer Networks* 101 (2016), 158–168.

[43] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3-4 (1992), 229–256.

[44] E. Zhan, S. Zheng, Y. Yue, L. Sha, and P. Lucey. 2018. Generative Multi-Agent Behavioral Cloning. *ArXiv e-prints* (March 2018). arXiv:1803.07612

[45] X. Zhang and H. Ma. 2018. Pretraining Deep Actor-Critic Reinforcement Learning Algorithms With Expert Demonstrations. *ArXiv e-prints* (Jan. 2018). arXiv:cs.AI/1801.10459