Centralized Model and Exploration Policy for Multi-Agent RL

Qizhen Zhang
University of Toronto, Vector Institute
Toronto, Canada
qizhen@cs.toronto.edu

Chris Lu
University of Oxford
Oxford, UK
christopher.lu@exeter.ox.ac.uk

Animesh Garg
University of Toronto, Vector Institute, NVIDIA
Toronto, Canada
garg@cs.toronto.edu

Jakob Foerster
University of Oxford
Oxford, UK
jakob.foerster@eng.ox.ac.uk

Abstract

Reinforcement learning (RL) in partially observable, fully cooperative multi-agent settings (Dec-POMDPs) can in principle be used to address many real-world challenges such as controlling a swarm of rescue robots or a team of quadcopters. However, Dec-POMDPs are significantly harder to solve than single-agent problems, with the former being *NEXP-complete* and the latter, MDPs, being just P-complete. Hence, current RL algorithms for Dec-POMDPs suffer from poor sample complexity, which greatly reduces their applicability to practical problems where environment interaction is costly. Our key insight is that using just a *polynomial* number of samples, one can learn a *centralized* model that generalizes across different policies. We can then optimize the policy within the learned model instead of the true system, without requiring additional environment interactions. We also learn a centralized exploration policy within our model that learns to collect additional data in state-action regions with high model uncertainty. We empirically evaluate the proposed model-based algorithm, MARCO*, in three cooperative communication tasks, where it improves sample efficiency by up to 20x. Finally, to investigate the theoretical sample complexity, we adapt an existing model-based method for tabular MDPs to Dec-POMDPs, and prove that it achieves polynomial sample complexity.

Keywords

Model-Based Multi-Agent Reinforcement Learning; Dec-POMDPs

ACM Reference Format:
Qizhen Zhang, Chris Lu, Animesh Garg, and Jakob Foerster. 2022. Centralized Model and Exploration Policy for Multi-Agent RL. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 9 pages.

1 Introduction

Decentralized partially observable Markov Decision Processes (Dec-POMDPs) describe many real-world problems [22, 31], but they are significantly harder to solve than Markov Decision Processes (MDPs). This is because the policy of one agent in Dec-POMDPs effectively serves as the *observation function* of other agents and, hence, agents need to explore over *policies* rather than actions. As a consequence, solving Dec-POMDPs involves searching through the space of tuples of decentralized policies that map individual action-observation histories to actions. This space is double exponential [27]:

$$O\left[\left(|A|^{\frac{(|O|^H-1)}{|O|-1}}\right)^N\right], \tag{1}$$

where $|O|$ is size of the observation space, $|A|$ is size of the individual action space, H is the horizon, and N is the number of agents. Since finding an optimal solution is doubly exponential in the horizon, the problem falls into a class called non-deterministic exponential (NEXP)-Complete [4].

Solving problems in this class is much harder than solving MDPs, which is just P-complete [29]. Indeed, current deep multi-agent RL algorithms for learning approximate solutions in Dec-POMDPs, which mostly extend model-free approaches such as independent learning [43], suffer from high sample complexity [2, 34]. This limits their applicability in real world problems and other settings where interactions with the environment are costly.

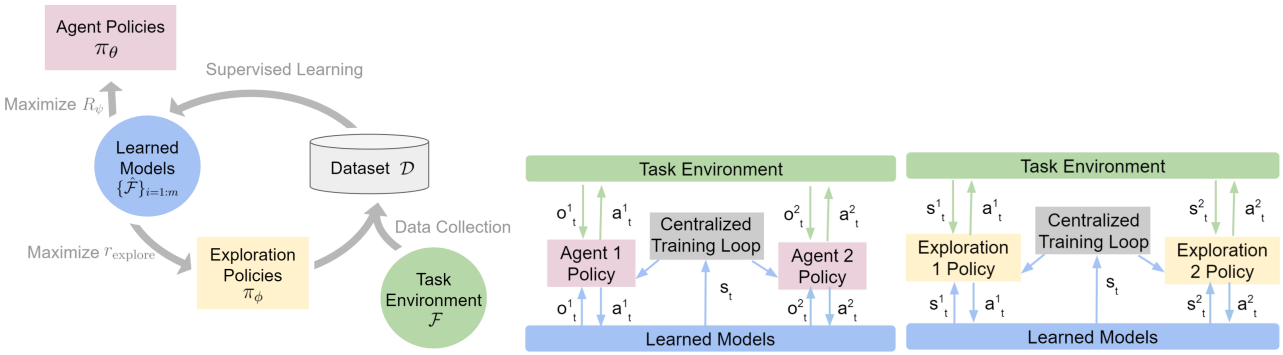
To address the problem of high environment sample complexity, MARCO uses a model-based approach. This is motivated by two reasons: 1) We take advantage of centralized training in Dec-POMDPs to learn a model of the environment that generalizes across different policies in just a polynomial number of samples (in the joint-action space and state space) like in single agent RL [38]. In contrast, as mentioned above, the sample complexity for learning an optimal policy is much larger in Dec-POMDPs (NEXP-complete vs P-complete). And 2) commonly in Dec-POMDPs, there are many possible optimal policies, where each of these only uses a small part of all possible states-action pairs during self-play. For learning an optimal policy, it is sufficient for the model to cover the state-action space associated with any *one* of these equilibria. Therefore in multi-agent settings, it is usually unnecessary to learn a good model of the entire environment.

Summary of contributions. This work makes four key contributions:

(1) First, we propose MARCO, a model-based learning algorithm for Dec-POMDPs. MARCO leverages centralized training to learn a model of the environment that generalizes across different policies. Within this model, we optimize the agents' policies using standard model-free methods, without using additional samples from the true environment.

*MARCO is short for **M**ulti-**A**gent **RL** with **C**entralized M**o**dels and Exploration Code available at `https://github.com/irenezhang30/MARCO/`.

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online.



(a) MARCO algorithm. (b) Agent policies (Observation as input). (c) Exploration policies (State as input).

Figure 1: Overview of the MARCO algorithm. Blue arrows indicate information flow during training, and green arrows indicate during test time. (a) MARCO alternates between policy optimization and model learning. During model learning, MARCO learns a centralized model that approximates the task environment. During policy optimization, MARCO updates the agents' policies using any model-free MARL algorithm of choice *within the learned model*. MARCO also learns a centralized exploration policy, which is used to collect data in the task environment for model learning. (b) Information flow between the agents' policies and the environment. The agents' policies are decentralized (i.e. they only take individual observations as input). (c) Information flow between the exploration policies and the environment. The exploration policies are centralized (i.e. they take the state as input). Instead of a single exploration policy across the joint-action space, which grows exponentially with the number of agents, MARCO uses one exploration policy per agent.

(2) In most existing Dyna-style RL algorithms [40, 41], data for model learning is collected using the agent's current policy. This is inefficient when data in the same state-action space is re-collected. To further improve sample complexity, MARCO uses a centralized exploration policy. This policy specifically collects data in parts of the state-action space with high model uncertainty and is trained *inside the model* to avoid consuming additional environment samples.
(3) To analyze the theoretical sample efficiency of model-based RL methods in Dec-POMDPs, we adapt R-MAX [5], a model-based dynamic programming algorithm for MDPs, to tabular Dec-POMDPs. Like MARCO, our adapted R-MAX also learns centralized models and performs exploration through "optimism in the face of uncertainty" [5]. We prove that this adapted R-MAX algorithm achieves a sample complexity polynomial in the size of the state and the joint-action space.
(4) And finally, we conduct empirical studies comparing MARCO with model-free state-of-the-art multi-agent RL (MARL) algorithms in three cooperative communication tasks, where MARCO improves sample efficiency by up to 20x.

2 Related Work

Single Agent Model-Based RL In RL problems, we generally do not assume prior knowledge of the environment. Model-free methods learn policies from interacting with the environment. In contrast, model-based RL (MBRL) methods first learn a model of the environment and use the model in turn for decision making. MBRL is well explored in the context of single-agent RL, and has recently shown promising results [13, 14, 35] across a variety of tasks [3, 44].

One problem of MBRL is that learning a perfect model is rarely possible, especially in environments with complex dynamics. In these settings, overfitting to model errors often hurt the test time performance. A popular method for addressing this problem is to learn an ensemble of models [21] and selecting one model randomly for each rollout step. Furthermore, the variance across the different models is a proxy for model uncertainty, which is used by e.g. Buckman et al. [7], Kalweit and Boedecker [18], Yu et al. [48]. During training in areas of high model-uncertainty these methods either penalize the agent or fall back to the real environment. A different approach is to actively explore in state-action space with high model uncertainty to learn better models [1, 37]. Our work uses the latter approach of active exploration, and closely aligns with [37], where we also explicitly learn an exploration policy within the learned model to perform data collection.

Model-free MARL for Dec-POMDPs Most deep RL work on learning in Dec-POMDPs uses model-free approaches. These methods can be roughly divided into two classes, value-based methods, e.g. [32, 39, 42], which build on DQN [25] and DRQN [15], and actor-critic methods, such as [10, 23]. These methods show good results in many tasks, but the number of samples required often goes into the millions or billions, even for environments with discrete or semantically abstracted state-action spaces [2, 33]. Orthogonal to our approach, few works have been proposed to address the sample complexity problem of MARL algorithms using off-policy learning methods [16, 45].

Model-based work in MARL A popular branch of work in the multi-agent setting studies opponent modelling. Instead of learning the dynamics of the environment, agents infer the opponents' policy from observing their behaviour to help decision making [6, 12, 24]. Along this line of work, Wang et al. trains an explicit model for each agent that predicts the goal conditioned motion of all agents in the partially observable environment.

There is little research in MARL that learns a model of the environment dynamics, as we do in our work (i.e. predicting the successor state from a state-action pair). Zhang et al. theoretically

analyse the sample complexity of model-based two-player zero-sum discounted Markov games, but do no present empirical studies. Krupnik et al. propose a multi-step generative model for two-player games, which does not predict the successor state, but the sequence of future joint-observation and joint-actions of all agents. Concurrent to our work, MAMBPO [47], most closely aligns with ours. Here, the authors learn a model of the environment, within which they perform policy optimization. While this work is concurrent to ours, there are also two key differences: 1) it does not learn a centralized exploration policy, and 2) MAMBPO uses the joint-observation and joint-action at the *current* timestep to predict the next joint-observation and reward. In contrast to our *fully centralized* model, which conditions on the central state, their model is not Markovian (see Figure 2). We illustrate this by a simple example: Suppose other agents in the environment can flip a light switch, but lights actually only turn on after a delay of 10 timesteps, which is reflected by a count-down value in the central state (not observed by any of the agents). The joint-observation and joint-action alone at the current timestep is insufficient for predicting the next joint-observation. In this example, the history of at least 10 past joint-observation and joint-action is required.

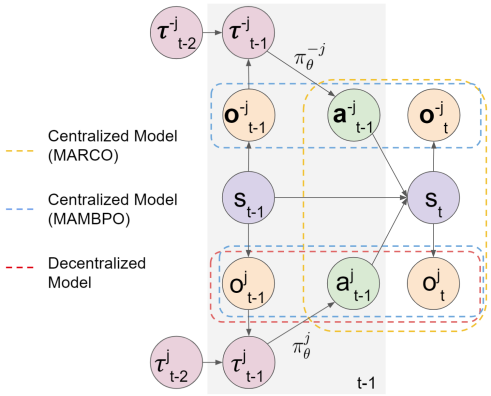


Figure 2: Probabilistic graphical models of the different possible learned Dec-POMDPs observation models. We use the notation $-j$ to denote the set of agents $\{1, .., n\} \setminus j$. The yellow box shows the centralized observation model $O_\psi(\mathbf{o}_t|s_t, \mathbf{a}_{t-1})$ learned by MARCO. The MARCO model is Markovian because the central state is used. It is also stationary since the environment dynamics are assumed to be stationary. The blue box shows the partially centralized model $O(\mathbf{o}_t|\mathbf{o}_{t-1}, \mathbf{a}_{t-1})$ learned by MAMBPO [47]. The model is non-Markovian because using the current timestep's joint-observation instead of the central state is insufficient to make the prediction. The red box shows the decentralized model $O^j(o^j_t|o^j_{t-1}, a^j_{t-1})$, which is also non-Markovian because the central state is not used. It is also non-stationary because the model depends on the policies of other agents π^{-j}_θ, which are updated throughout training.

3 Background

3.1 Dec-POMDPs

We consider a fully cooperative, partially observable task that is formalized as a decentralized partially observable Markov Decision Process (Dec-POMDP) [26] $\mathcal{F} = \langle S, A, P, R, Z, O, N, \gamma, d_0 \rangle$. $s \in S$ describes the central state of the environment, and d_0 is the initial state distribution. At each timestep, each agent $j \in J \equiv \{1, \ldots, N\}$ draws individual observations $o^j \in Z$ according to the observation function $O(s, \mathbf{a}) : S \times \mathbf{A} \rightarrow \mathbf{Z}$. Each agent then chooses an action $a^j \in A$, forming a joint-action $\mathbf{a} \in \mathbf{A} \equiv A^N$ [†]. This causes a transition in the environment according to the state transition function $P(s' \mid s, \mathbf{a}) : S \times \mathbf{A} \times S \rightarrow [0, 1]$. All agents share the same reward function $R(s, \mathbf{a}) : S \times \mathbf{A} \rightarrow \mathbb{R}$ and $\gamma \in [0, 1)$ is a discount factor.

Each agent has an action-observation history (AOH) $\tau^j \in T \equiv (Z \times A)^*$, on which it conditions a stochastic policy $\pi^j\left(a^j \mid \tau^j\right) : T \times A \rightarrow [0, 1]$. The joint-policy π induces a joint action-value function: $Q^\pi\left(s_t, \mathbf{a}_t\right) = \mathbb{E}_{s_{t+1:\infty}, \mathbf{a}_{t+1:\infty}}\left[R_t \mid s_t, \mathbf{a}_t\right]$, where $R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$ is the discounted return.

3.2 Dyna Style Model-Based RL

RL algorithms fall under two classes: model-free methods, where we directly learn value functions and/or policies by interacting with the environment, and model-based methods, where we use interactions with the environment to learn a model of it, which is then used for decision making. Dyna-style algorithms [40, 41] are a family of model-based algorithms for single-agent RL where training alternates between two steps: model learning and policy optimization. During model learning data is collected from the environment using the current policy and is used to learn the transition function. During policy optimization the policy is improved using a model-free RL algorithm of choice from data generated by the learned model.

3.3 Model-Free Multi-Agent Approaches

Most MARL methods for approximately solving Dec-POMDPs fall in the category of model-free methods. Many use the centralized training for decentralized execution (CTDE) framework [10, 19, 28], i.e., the learning algorithm has access to all global information, such as the joint-actions and the central state, but, at test time, each agent's learned policy conditions only on its own AOH τ^j.

A popular branch of multi-agent methods for partially observable, fully cooperative settings is based on Independent Q-Learning (IQL) [42, 43]. IQL treats the Dec-POMDP problem as simultaneous single-agent problems. Each agent learns its own Q-value that conditions only on the agent's own observation and action history, treating other agents as a part of the environment. MAPPO [8], another independent learning algorithm, extends PPO [36] to Dec-POMDPs. The advantage of independent learning is that it factorizes the exponentially large joint-action space. However, due to the non-stationarity in the environment induced by the learning of other agents, convergence is no longer guaranteed. Works like VDN and QMIX [32, 39] partially address this issue by learning joint Q-values. The former uses the sum of value functions of individual agents as the joint Q-values, while the latter learns a function parameterized by a neural network to map from individual Q-values to joint Q-values using the central state.

[†]Bold notation indicates joint quantity over all agents.

4 Methods

Optimally solving, or even finding an ϵ-approximate solution for, Dec-POMDPs is NEXP-complete [4, 30], which is significantly harder than solving MDPs with a complexity of P-complete [29]. This provides a strong motivation for using a model-based approach in Dec-POMDPs, as the number of samples required for learning a centralized model is polynomial in the state-action space, like in single-agent-RL [38].

To learn a policy in Dec-POMDPs, MARCO (see Figure 1) adapts Dyna-style model-based RL to the multi-agent setting, as shown in Algorithm 1[‡], which alternates between learning an approximation of the Dec-POMDP, $\hat{\mathcal{F}}$, and optimizing the policy within $\hat{\mathcal{F}}$. We refer to $\hat{\mathcal{F}}$ as *the model* for the remainder of the paper. The two key contributions of MARCO are 1) learning a stationary model via centralized training, and 2) actively collecting data using a separate centralized *exploration policy* trained inside the model not requiring additional environment samples.

4.1 Model-Based MARL with Centralized Models

The model $\hat{\mathcal{F}}$ is composed of the following components, each of which is a parameterized, learned approximation of the original Dec-POMDP $\mathcal{F}$:

$$\begin{aligned} \text{Reward model}: &\quad R_\psi(r_t|s_t, \mathbf{a_t}, s_{t-1}) \\ \text{Dynamics model}: &\quad T_\psi(s_{t+1}|s_t, \mathbf{a_t}) \\ \text{Observation model}: &\quad O_\psi(\mathbf{o_t}|s_t, \mathbf{a_{t-1}}) \\ \text{Termination model}: &\quad P_{\text{term}\psi}(\text{termination}|s_t, \mathbf{a_t}) \end{aligned}$$

In the single-agent, fully observable MBRL setting, the agent learns a dynamics model, and sometimes a reward and termination model. In MARCO, we also learn an observation model.

MARCO takes advantage of CTDE by letting all of our models condition on the central state as well as the joint-action. The importance of centralized model learning can be illustrated as follows: For example (see Figure 2), if the observation model is learned in a decentralized fashion (i.e. $O(o_t^j|o_{t-1}^j, a_{t-1}^j)$), then the actions of other agents, even when unobserved, can change the transition function of agent j (i.e. if another agent turns on a light, agent j will observe that the light transitioned from "off" to "on"). As agents' policies are changing throughout training, the observation model thus is non-stationary and would have to be re-learned at various stages during training. In contrast, MARCO learns a fully centralized model which approximates the stationary *ground truth* Dec-POMDP. Crucially, while in Dec-POMDPs agents have to explore over policies, MARCO allows agents to learn a single stationary model that is simultaneously accurate for *all different* policies being explored.

Each component of the model is parametrized by a separate neural network and is trained using supervised learning through maximizing the likelihood of the collected data. Similar to Kurutach et al., we train an ensemble of models to prevent the policy from overfitting to and exploiting model errors. When generating rollout data using the model, we randomly sample which one of the ensemble models to generate from.

[‡]For ease of notation, the input to all components $\hat{f} \in \hat{\mathcal{F}}$ in Algorithm 1 is written as $s_t, \mathbf{a_t}$. Actual inputs of $\hat{f}$ vary based on which model component $\hat{f}$ is.

Algorithm 1 MARCO: Multi-Agent RL with Centralized Models and Exploration

1: **Input**: Number of ensemble models m, and uncertainty hyper-parameter λ.
2: Initialize action-value functions Q_θ, Q_ϕ.
3: Initialize m ensemble environment models $\hat{\mathcal{F}}_i = \{R_{i,\psi}, T_{i,\psi}, O_{i,\psi}, P_{\text{term,i}_\psi}\}$ for $i = 1, .., m$.
4: Initialize a dataset D with samples collected using a random policy from the real environment $\mathcal{F}$.
5: **repeat**
6: Train model $\hat{\mathcal{F}}$ using dataset D.
7: Update Q_θ using model-free algorithm of choice within a randomly chosen model $\hat{\mathcal{F}}_i$.
8: Update Q_ϕ using model-free algorithm of choice in a centralized fashion within a randomly chosen model $\hat{\mathcal{F}}_i$ as follows:

$$\begin{aligned} \tilde{r}(s_t, \mathbf{a_t}) &= \sum_{\hat{f} \in \hat{\mathcal{F}}} \sum_{k=1}^{d(\hat{f})} Var(\{\hat{f}_{k,i,\psi}(s_t, \mathbf{a_t})\}_{i=1,..,m}) \\ r_{\text{explore}} &= R_\psi(s_t, \mathbf{a_t}, s_{t-1}) + \lambda \tilde{r}(s_t, \mathbf{a}_t) \\ y &= r_{\text{explore}} + \gamma \max_{\mathbf{a}'} \sum_{j=1}^{n} Q_{\text{target}}\left(s_{t+1}, a', j; \phi^-\right) \\ \mathcal{L}(\phi) &= \sum_{i=1}^{b} \left[\left(y_i - \sum_{j=1}^{n} Q(s, \mathbf{a}, j; \phi) \right)^2 \right], \end{aligned} \tag{2}$$

where $d(\hat{f})$ is the dimensionality of the model component $\hat{f}$.
9: Collect samples from environment $\mathcal{F}$ using centralized exploration policy π_ϕ and add them to D.
10: **until** Maximum environment samples is used.

4.2 Model-Based MARL with Centralized Exploration Policy

MARCO collects data for model learning from a separate exploration policy π_ϕ. Ideally, we want to collect data in regions of the state-action space with high model uncertainty. To quantify this *epistemic* uncertainty, we use the variance of the models in the ensemble, which we denote as $\tilde{r}$ in equation 2. To prevent the exploration policy from wandering off to regions irrelevant to the search space of the policies, the exploration policy should also optimize for the original objective. Hence, we set the reward of the exploration policy as the linear combination of $\tilde{r}$ and the reward generated by the reward model R_ψ. The hyper-parameter λ controls the trade-off between exploration and exploitation. The exploration policy is learned entirely in the model, without using additional samples from the ground truth environment. To train the exploration policy, we use VDN (or QMIX), and again fully exploit CTDE by using the central state inside the model. By conditioning on the central state, the exploration policy is able to more quickly return to the frontier, where high model uncertainty starts to occur. This avoids repeating

data collection in regions of the already known state-action space, allowing more sample efficient model learning.

Note that although MARCO's exploration policies uses centralized information, each agent's exploration policy outputs only their respective action. This is opposed to learning a single exploration policy that outputs the joint-action, which becomes intractable due to the exponential joint-action space.

5 Sample Complexity in Tabular Dec-POMDPs

We investigate the theoretical sample complexity of model-based Dec-POMDP methods. To do so, we make four additional assumptions that are not required for MARCO a) discrete and finite state, observation and action space b) at each timestep, the reward is bounded $0 \leq r(s, \mathbf{a}) \leq 1$, c) finite horizon, and d) deterministic observation function. Under these assumptions, we show that an idealized model-based method achieves a sample complexity polynomial in the size of the state and joint-action space.

We modify R-MAX [5], an MBRL algorithm for MDPs, to the Dec-POMDP setting (see Algorithm 2). We refer to our modified algorithm as the Adapted R-MAX for the remaining of the paper. Like MARCO, the Adapted R-MAX aims to learn a near-optimal decentralized joint-policy for a given Dec-POMDP D. Our adaptation of R-MAX also takes full advantage of CTDE by learning centralized models $\hat{P}(s, \mathbf{a})$, $\hat{R}(s, \mathbf{a})$, and $\hat{O}(s)$ using empirical estimates. Using the centralized models, the Adapted R-MAX constructs an approximate *K-known Dec-POMDP* $\hat{D}_K$ (see Definition 5.1), where K is the set of state-action pairs that has been visited at least m times. Within the model $\hat{D}_K$, we then evaluate all possible joint-policies $\pi \in \mathbf{\Pi}$ and choose the best one. Both MARCO and our adaptation of R-MAX encourage exploration in parts of the state-action space with high model uncertainty. The former performs exploration through a separate centralized exploration policy, while the latter performs exploration through optimistically setting the reward function of under-visited state-action pairs (i.e. those not in K).

Definition 5.1 (K-Known Dec-POMDP)**.** D_K is the expected version of $\hat{D}_K$ where:

$$P_K\left(s' \mid s, \mathbf{a}\right) = \begin{cases} P\left(s' \mid s, \mathbf{a}\right) & \text{if } (s, \mathbf{a}) \in K \\ \mathbb{1}\left[s' = s\right] & \text{otherwise} \end{cases}$$

$$\hat{P}_K\left(s' \mid s, \mathbf{a}\right) = \begin{cases} \frac{n(s,\mathbf{a},s')}{n(s,\mathbf{a})}, & \text{if } (s, \mathbf{a}) \in K \\ \mathbb{1}\left[s' = s\right], & \text{otherwise} \end{cases}$$

$$R_K\left(s, \mathbf{a}\right) = \begin{cases} R\left(s, \mathbf{a}\right) & \text{if } (s, \mathbf{a}) \in K \\ R_{\max} & \text{otherwise} \end{cases}$$

$$\hat{R}_K(s, \mathbf{a}) = \begin{cases} \frac{\sum_i^{n(s,\mathbf{a})} r(s,\mathbf{a})}{n(s,\mathbf{a})}, & \text{if } (s, \mathbf{a}) \in K \\ R_{\max}, & \text{otherwise} \end{cases}$$

$$O_K\left(s\right) = \begin{cases} O\left(s\right) & \text{if } (s, \cdot) \in K \\ \text{random observation} & \text{otherwise} \end{cases}$$

$$\hat{O}_K\left(s\right) = \begin{cases} \mathbf{o} \text{ where } n\left(s, \mathbf{o}\right) > 0 & \text{if } (s, \cdot) \in K \\ \text{random observation} & \text{otherwise} \end{cases}$$

To study the sample complexity of the Adapted R-MAX, we define the value of a joint-policy as follows:

Algorithm 2 Adapted R-MAX for Dec-POMDPs

Input: γ, m.
for all $(s, \mathbf{o})$ **do**
  $n(s, \mathbf{o}) \leftarrow 0$
for all $(s, \mathbf{a})$ **do**
  $r(s, \mathbf{a}) \leftarrow 0$
  $n(s, \mathbf{a}) \leftarrow 0$
  for all $s' \in S$ **do**
    $n(s, \mathbf{a}, s') \leftarrow 0$
for $t = 1, 2, 3, \cdots$ **do**
  Let $s, \mathbf{o}$ denote the state and observation at time t respectively.
  Choose action $\mathbf{a}$ according to $\pi^*_{\hat{D}_K}$
  Let r be the immediate reward and s' the next state after executing action $\mathbf{a}$ from state s.
  if $n(s, \mathbf{a}) < m$ **then**
    $n(s, \mathbf{o}) \leftarrow 1$ // Record observation
    $n(s, \mathbf{a}) \leftarrow n(s, \mathbf{a}) + 1$
    $r(s, \mathbf{a}) \leftarrow r(s, \mathbf{a}) + r$// Record immediate reward
    $n(s, \mathbf{a}, s') \leftarrow n(s, \mathbf{a}, s') + 1$// Record immediate next-state
  if $n(s, \mathbf{a}) = m$ **then**
    for all $\pi \in \mathbf{\Pi}$ **do**:
      Obtain $J_{\hat{D}_K}(\pi)$ using Monte Carlo rollouts in $\hat{D}_K$.
    $\pi^*_{\hat{D}_K} \leftarrow \arg\max_\pi J_{\hat{D}_K}(\pi)$

Definition 5.2. Given a decentralized joint-policy π, we estimate its value $J(\pi)$ in a Dec-POMDP D, defined as the expected reward obtained by following the joint-policy in D,

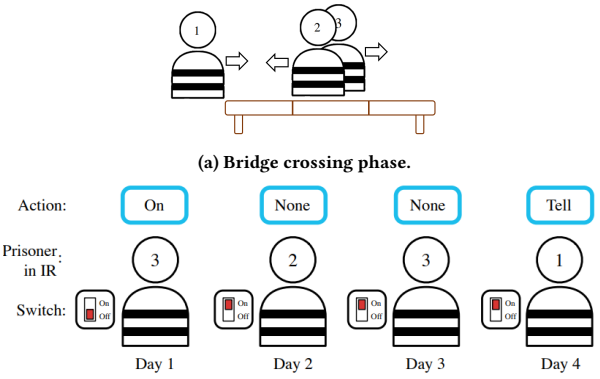
$$J_D(\pi) = \mathbb{E}_{s \sim d_0}\left[V^\pi(s)\right]. \tag{3}$$

Theorem 5.1. *Suppose that $0 \leq \varepsilon < 1$ and $0 \leq \delta < 1$ are two real numbers and D is any Dec-POMDP. There exists inputs $m = m\left(\frac{1}{\varepsilon}, \frac{1}{\delta}\right)$ and ε, satisfying $m\left(\frac{1}{\varepsilon}, \frac{1}{\delta}\right) = O\left(\frac{(S+\ln(SA/\delta))V_{\max}^2}{\varepsilon^2(1-\gamma)^2}\right)$ such that if the Adapted R-MAX algorithm is executed on D with inputs m and ε, then the following holds: Let $\pi^*_{\hat{D}_K}$ denote the Adapted R-MAX 's policy. With probability at least $1-\delta$, $J_D(\pi^*) - J_D(\pi^*_{\hat{D}_K}) \leq 2\epsilon$ is true for all but*

$$O\left(\frac{|S||\mathbf{A}|}{(1-\gamma)^2\epsilon^3}\left(|S| + \ln(\frac{|S||\mathbf{A}|}{\delta})\right)V_{\max}^3 \ln\frac{1}{\delta}\right)$$

episodes.

Theorem 5.1 shows that Adapted R-MAX acts near-optimally on all but a polynomial number of steps. These results confirm the motivation of MARCO, i.e. that an idealized MBRL method for Dec-POMDPs can indeed have polynomial sample complexity. The proof (see the appendix [50]) is heavily based on results from Jiang and Strehl et al., but for the first time extends them to the Dec-POMDP setting.



(a) Bridge crossing phase.

(b) Switch riddle playing phase.

Figure 3: *Switch riddle with bridge.* (a) The game begins with the *bridge crossing* phase. All agents start on the left side of the bridge. At each timestep, each agent chooses an action from {"Left", "Right", and "End episode"}. (b) Switch riddle *playing phase* [11]. After all agents arrive at the right side of the bridge, every timestep one agent gets sent to the interrogation room where they see the switch and choose an action from {"On", "Off", "Tell", "None", "Left", "Right", and "End episode"}.

6 Experiments

6.1 Environments

We evaluate the sample efficiency of MARCO against model-free MARL algorithms on three fully cooperative, partially observable communication tasks, the switch riddle [11], a variant of the switch riddle, and the simple reference game from the multi-agent particle environment (MPE) [23]. We explicitly chose communication tasks because one agent's belief is directly affected by other agents' policies, resulting in the larger policy search spaces typical for Dec-POMDPs.

Switch without Bridge [11] At each timestep t, a random agent $j \in \{1, 2, 3\}$ is sent into the interrogation room for one timestep. Each agent observes whether it is currently in the room, but only the current agent in the room observes whether the light switch in the room is "On" or "Off". If agent j is in the interrogation room, then its actions are $a_t^j \in$ { "None", "Tell", "Turn on lights", "Turn off lights"}; otherwise the only action is "None". The episode ends when an agent chooses "Tell" or when the maximum timestep, T, is reached. The reward r_t is 0 unless an agent chooses "Tell", in which case it is 1 if all agents have been to the interrogation room, and -1 otherwise. Finally, to keep the experiments computationally tractable we set the time horizon to $T = 6$.

Switch with Bridge To make the first task more challenging, we modify it as follows (see Figure 3). All agents start on the left side of a bridge, and the switch riddle only starts once all agents have crossed a bridge of length 3. If at timestep t not all agents have crossed the bridge, each agent observes its position $o_t^j \in \{0, 1, 2, 3\}$ on the bridge, and its actions are $a_t^j \in$ { "Left", "Right", "End episode"}. Selecting the action "Left" and "Right" increments the agent's position by -1 and +1 respectively. The episode ends and agents receive a reward of 0 if "End episode" is chosen. When all agents have crossed the bridge (i.e. all agents are at position 3 on the bridge), the switch riddle starts. Now agents proceeds like the above task, except that the agent currently in the room has access to additional actions $a_t^j \in$ { "None", "Tell", "Turn on lights", "Turn off lights", "Left", "Right", "End episode "}. Selecting "Left" or "Right" is equivalent to selecting "None", and selecting "End episode" terminates the episode early with a return of 0. We set the time horizon to $H = 9$.

The Simple Reference Game This task is a part of the multi-agent particle environment (MPE) [23], and consists of two agents, that are placed in an environment with three landmarks of differing colors. At the beginning of every episode, each agent is assigned to a landmark of a particular color. The closer the agents are to their assigned landmark, the higher their reward. However, agents themselves don't observe their own assigned color, only the other agent's assigned color. At each timestep, each agent chooses two actions: A movement action ∈ { "Left", "Right", "Up", "Down", "Do nothing"}, and one of the 10 possible messages that is sent to the other agent.

6.2 Experiment Details [§]

Model Learning The dynamics model in the two switch tasks is an auto-regressive model implemented using GRUs [9]. The remaining components of the model $\hat{\mathcal{F}}$ are implemented using fully connected neural networks. All model components are trained in supervised manner via maximum likelihood.

Dataset Collection The initial dataset is gathered with a random policy for all MARCO experiments. In the *switch without bridge* task, 5k samples are collected from the environment after every 10k training steps in the model. No further data is collected beyond 10k samples. For the *switch with bridge* and the MPE tasks, 10k samples are collected from the environment after every 50k training steps in the model. No further data is collected beyond 50k samples.

Policy Optimization The model-free baseline for each task is chosen by finding the most sample efficient algorithm between IQL, VDN, and QMIX. For each task, MARCO uses the same algorithm for policy optimization inside the model as the corresponding model-free baseline.

6.3 Results

The top row in Figure 4 displays results against *policy training steps* to show MARCO matches model-free performance, while the middle row displays results against *number of real environment interactions* to show sample efficiency of MARCO. The bottom row illustrates the performance against *number of real environment interactions in log scale*, where we see a 1-2 order of magnitude improvement of sample-efficiency in MARCO over model-free methods.

The left column in Figure 4 shows results for the *switch with bridge* task. Model-free IQL learns the optimal policy in roughly 200k samples. MARCO learns the optimal policy with 10k samples, which is a sample efficiency increase of 20x.

[§]See the appendix [50] for detailed experiment descriptions

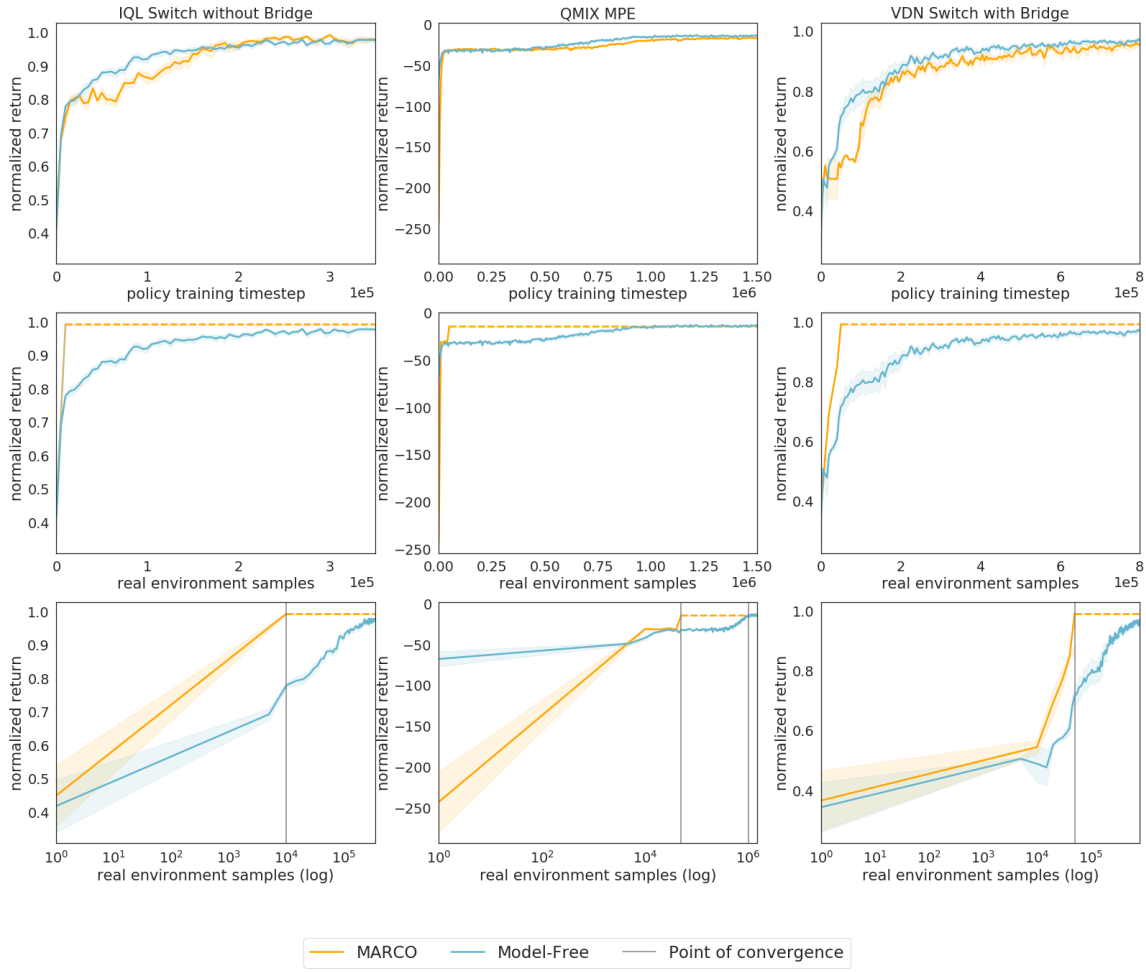


Figure 4: MARCO's test performance over 50 episodes with $\lambda = 2.0$ matches model-free performance with much less environment samples. The error bars reported are the standard error over 8 runs. Left: IQL in the *switch without bridge* task. MARCO is 20x more sample efficient. Middle: QMIX in MPE. MARCO is 20x more sample efficient. Right: VDN in the *switch with bridge* task. The MARCO model is learned with 50k samples. MARCO is 12x more sample efficient.

The middle column in Figure 4 shows results for the MPE task. Model-free QMIX learns the optimal policy with roughly 1m samples, while MARCO learns it in 50k, an efficiency increase of 20x.

The right column in Figure 4 shows results for the *switch with bridge crossing*. MARCO learns the optimal policy with only 50k samples. In comparison, model-free VDN requires roughly 600k samples, about 12x more than MARCO.

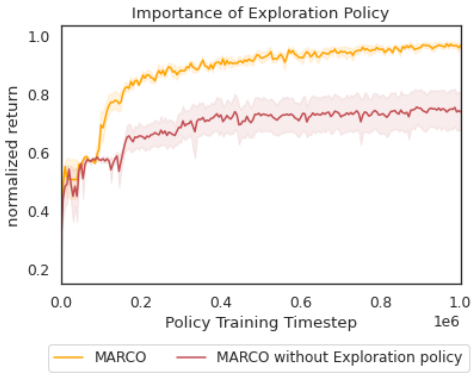
6.4 Ablation Studies

We evaluate the centralized exploration policy with two ablation studies in the *switch with bridge* task.

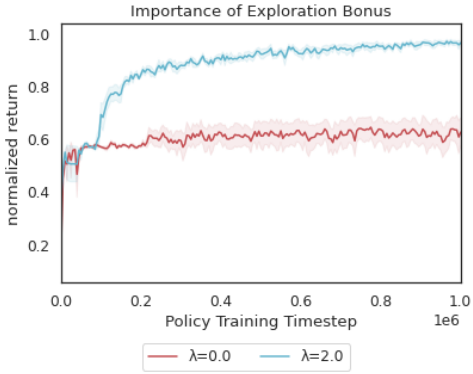
Effect of using exploration policy Figure 5a [¶] shows results comparing MARCO with an ablation without exploration policy. In the latter case, the model is learned with a dataset that is collected with the agents' current policy instead of a dedicated exploration policy. An ϵ-greedy policy is used for data collection, where with a probability of $\epsilon = 0.1$ the agent selects a random action from the available actions at the current step, and follows the data-collection policy otherwise.

MARCO without an exploration policy fails to solve this task because random exploration from the ϵ-greedy data collection policy is insufficient to cover the relevant state-action space. If a random policy is used, at every timestep during the bridge crossing phase, there is a probability of $1-(\frac{2}{3})^3 \approx 0.3$ that at least one agent chooses "End episode" among the three available actions. Hence, the episode is very likely to terminate before the switch-riddle playing even begins, and so collecting data with a random exploration for the switch riddle task is difficult at the initial learning phase. In contrast, MARCO with centralized exploration policy overcomes this problem by actively exploring in the state-action space with high model-uncertainty, which is where agents cross the bridge and play the switch riddle. This is illustrated in Figure 6, where we show that MARCO agents cover much more state-action space when using an exploration policy (Figure 6e-6h) than otherwise (Figure 6a -

[¶] The state-action space is displayed in 2D for ease of visualization



(a) Importance of exploration policy.



(b) Importance of exploration bonus.

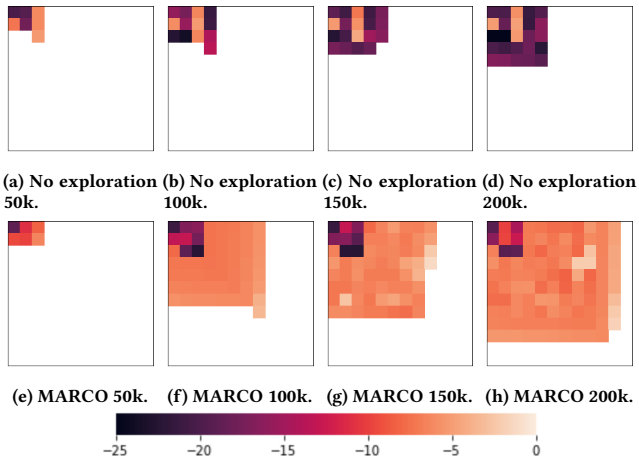
Figure 5: Ablation studies in the *switch with bridge* task. (a) Without the exploration policy, MARCO agents do not learn the optimal policy.(b) Without the exploration bonus term $\tilde{r}$ (i.e. setting λ=0), MARCO's final policy is sub-optimal. The error bars shown represent the standard error across 8 runs.

6d). The columns indicate the log of the uncertainty term of 5000 state-action pairs visited by MARCO's agents policies in the task environment after training for 50k, 100k, 150k, and 200k timesteps in the model. The rollout is done within the model that is available to the agents' policies at the time. Darker color indicates less model uncertainty, lighter color indicates more. No color (white) indicates an unvisited state-action pair.

Effect of using exploration bonus $\tilde{\mathbf{r}}$ Figure 5b shows that when we do not use the exploration bonus term $\tilde{\mathbf{r}}$ by setting $\lambda = 0$, MARCO no longer learns the optimal policy. This suggest that the exploration bonus term is essential.

7 Conclusion and Future Work

We presented MARCO, a model-based RL method adapted from the Dyna-style framework for sample-efficient learning in Dec-POMDPs. MARCO learns a centralized stationary model that in principle is entirely independent of the agents' policies. Within this model, policy optimization is performed using a model-free MARL algorithm of choice with no additional cost in environment samples (exploiting access to the *simulated central state*). To further improve sample complexity, MARCO also learns a centralized exploration policy to collect samples in parts of the state-action space with high model uncertainty. In addition, to investigate the theoretical sample



(a) No exploration 50k. **(b) No exploration 100k.** **(c) No exploration 150k.** **(d) No exploration 200k.**

(e) MARCO 50k. **(f) MARCO 100k.** **(g) MARCO 150k.** **(h) MARCO 200k.**

Figure 6: Log of the uncertainty term ($\log \tilde{r}$) of 5000 state-action pairs visited by the agents policies in the task environment either with (e-h) or without (a-d) the centralized exploration policy. The columns correspond to MARCO policies trained inside the model for 50k, 100k, 150k, and 200k timesteps, respectively. The $\log \tilde{r}$ term is calculated with the models that is available to the MARCO policy at the time of training. Darker color indicates less uncertainty, lighter color indicates more. No color (white) indicates an unvisited state-action pair. When the exploration policy is used, the agents cover much more of the state-action space starting at 100k than without the exploration policy.

efficiency of model-based Dec-POMDP methods, we introduced the Adapted R-MAX algorithm for Dec-POMDPs, and showed that it achieves polynomial sample complexity. Finally, we showed on three cooperative communication tasks that MARCO matches the performance of state-of-the-art model-free MARL methods requiring significantly fewer samples.

We discuss the limitations of the work in two aspects. First of all, the centralized exploration policy may deviate from the agents' decentralized policies due to having access to additional information in the central state. This may cause data to be collected in parts of the state-action space that are inaccessible to the agents due to partial observability. Secondly, model-based methods commonly require more wall-clock time than model-free methods due to the additional model-learning step. However, by assumption, in our setting compute-time is cheap compared to environment interactions.

Despite these limitations, we hope that this work brings MARL one step closer to being applicable to real-world problems. In future work we will investigate how to choose a better centralized data exploration policy, as well as how to combine existing work in image-input single-agent MBRL to MARL settings to enable good performance even on complicated, image-based environments.

8 Acknowledgements

Authors thank Wendelin Böhmer, Amir-massoud Farahmand, Keiran Paster, Claas Voelcker, and Stephen Zhao for insightful discussions and/or feedbacks on the drafts of the paper. QZ was supported by the Ontario Graduate Scholarship. AG and JF would like to acknowledge the CIFAR AI Chair award.

References

[1] Philip Ball, Jack Parker-Holder, Aldo Pacchiano, Krzysztof Choromanski, and Stephen Roberts. 2020. Ready policy one: World building through active learning. In *International Conference on Machine Learning*. PMLR, 591–601.

[2] Nolan Bard, Jakob N Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, et al. 2020. The hanabi challenge: A new frontier for ai research. *Artificial Intelligence* 280 (2020), 103216.

[3] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research* 47 (2013), 253–279.

[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] Ronen I Brafman and Moshe Tennenholtz. 2002. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research* 3, Oct (2002), 213–231.

[6] George W Brown. 1951. Iterative solution of games by fictitious play. *Activity analysis of production and allocation* 13, 1 (1951), 374–376.

[7] Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. 2018. Sample-efficient reinforcement learning with stochastic ensemble value expansion. *arXiv preprint arXiv:1807.01675* (2018).

[8] YU Chao, A VELU, E VINITSKY, et al. 2021. The surprising effectiveness of PPO in cooperative, multi-agent games. *arXiv preprint arXiv:2103.01955* (2021).

[9] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

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

[11] Jakob N Foerster, Yannis M Assael, Nando De Freitas, and Shimon Whiteson. 2016. Learning to communicate with deep multi-agent reinforcement learning. *arXiv preprint arXiv:1605.06676* (2016).

[12] Jakob N Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. 2017. Learning with opponent-learning awareness. *arXiv preprint arXiv:1709.04326* (2017).

[13] David Ha and Jürgen Schmidhuber. 2018. World models. *arXiv preprint arXiv:1803.10122* (2018).

[14] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. 2019. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603* (2019).

[15] Matthew Hausknecht and Peter Stone. 2015. Deep recurrent q-learning for partially observable mdps. *arXiv preprint arXiv:1507.06527* (2015).

[16] Wonseok Jeon, Paul Barde, Derek Nowrouzezahrai, and Joelle Pineau. 2020. Scalable and sample-efficient multi-agent imitation learning. In *Proceedings of the Workshop on Artificial Intelligence Safety, co-located with 34th AAAI Conference on Artificial Intelligence, SafeAI@ AAAI*.

[17] Nan Jiang. 2020. Notes on Rmax exploration. (2020).

[18] Gabriel Kalweit and Joschka Boedecker. 2017. Uncertainty-driven imagination for continuous deep reinforcement learning. In *Conference on Robot Learning*. PMLR, 195–206.

[19] Landon Kraemer and Bikramjit Banerjee. 2016. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing* 190 (2016), 82–94.

[20] Orr Krupnik, Igor Mordatch, and Aviv Tamar. 2020. Multi-agent reinforcement learning with multi-step generative models. In *Conference on Robot Learning*. PMLR, 776–790.

[21] Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. 2018. Model-ensemble trust-region policy optimization. *arXiv preprint arXiv:1802.10592* (2018).

[22] Miao Liu, Kavinayan Sivakumar, Shayegan Omidshafiei, Christopher Amato, and Jonathan P How. 2017. Learning for multi-robot cooperation in partially observable stochastic environments with macro-actions. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 1853–1860.

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

[24] Richard Mealing and Jonathan L Shapiro. 2015. Opponent modeling by expectation–maximization and sequence prediction in simplified poker. *IEEE Transactions on Computational Intelligence and AI in Games* 9, 1 (2015), 11–24.

[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), 529–533.

[26] Frans A Oliehoek and Christopher Amato. 2016. *A concise introduction to decentralized POMDPs*. Springer.

[27] Frans A Oliehoek, Julian FP Kooij, and Nikos Vlassis. 2008. The cross-entropy method for policy search in decentralized POMDPs. *Informatica* 32, 4 (2008).

[28] 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.

[29] Christos H Papadimitriou and John N Tsitsiklis. 1987. The complexity of Markov decision processes. *Mathematics of operations research* 12, 3 (1987), 441–450.

[30] Zinovi Rabinovich, Claudia V Goldman, and Jeffrey S Rosenschein. 2003. The complexity of multiagent systems: The price of silence. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*. 1102–1103.

[31] Shankarachary Ragi and Edwin KP Chong. 2014. Decentralized guidance control of UAVs with explicit optimization of communication. *Journal of Intelligent & Robotic Systems* 73, 1 (2014), 811–822.

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

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

[34] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philiph H. S. Torr, Jakob Foerster, and Shimon Whiteson. 2019. The StarCraft Multi-Agent Challenge. *CoRR* abs/1902.04043 (2019).

[35] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. 2020. Mastering atari, go, chess and shogi by planning with a learned model. *Nature* 588, 7839 (2020), 604–609.

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

[37] Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. 2020. Planning to explore via self-supervised world models. In *International Conference on Machine Learning*. PMLR, 8583–8592.

[38] Alexander L Strehl, Lihong Li, and Michael L Littman. 2009. Reinforcement Learning in Finite MDPs: PAC Analysis. *Journal of Machine Learning Research* 10, 11 (2009).

[39] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. 2017. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296* (2017).

[40] Richard S Sutton. 1990. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine learning proceedings 1990*. Elsevier, 216–224.

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

[42] 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.

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

[44] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 5026–5033.

[45] Bozhidar Vasilev, Tarun Gupta, Bei Peng, and Shimon Whiteson. 2021. Semi-On-Policy Training for Sample Efficient Multi-Agent Policy Gradients. *arXiv preprint arXiv:2104.13446* (2021).

[46] Rose E Wang, Chase Kew, Dennis Lee, Edward Lee, Brian Andrew Ichter, Tingnan Zhang, Jie Tan, and Aleksandra Faust. 2020. Model-based Reinforcement Learning for Decentralized Multiagent Rendezvous. In *Conference on robot learning*. PMLR.

[47] Daniël Willemsen, Mario Coppola, and Guido CHE de Croon. 2021. MAMBPO: Sample-efficient multi-robot reinforcement learning using learned world models. *arXiv preprint arXiv:2103.03662* (2021).

[48] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. 2020. Mopo: Model-based offline policy optimization. *arXiv preprint arXiv:2005.13239* (2020).

[49] Kaiqing Zhang, Sham M Kakade, Tamer Başar, and Lin F Yang. 2020. Model-based multi-agent rl in zero-sum markov games with near-optimal sample complexity. *arXiv preprint arXiv:2007.07461* (2020).

[50] Qizhen Zhang, Chris Lu, Animesh Garg, and Jakob Foerster. 2021. Centralized Model and Exploration Policy for Multi-Agent RL. *arXiv preprint arXiv:2107.06434* (2021).