Centralized Model and Exploration Policy for Multi-Agent RL

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

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

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

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

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 single-agent 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]:

$$O \left( \left| A \right| \left( \left| O \right| H - 1 \right) \right|^N \right),$$

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.
(a) MARCO algorithm. Blue arrows indicate information flow during training, and green arrows indicate information flow 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
We consider a fully cooperative, partially observable task that is not Markovian because the central state is not used. It is also non-stationary because the environment dynamics are assumed to be stationary. 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 central-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.

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] $\mathcal{F} = (S, A, P, R, Z, O, N, f, o_0)$. $s \in S$ describes the central state of the environment, and $o_0$ 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 $O(s, a) : S \times A \rightarrow Z$. Each agent then chooses an action $a_j \in A$, forming a joint-action $a \in A \equiv A^N$.

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

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.
4 Methods

Optimally solving, or even finding an $\epsilon$-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, $\hat{F}$, and optimizing the policy within $\hat{F}$. We refer to $\hat{F}$ as the model for the remainder of the paper. The two key contributions of MARCO are 1) learning a stationary model via intrinsic motivation, 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 $\hat{F}$ is composed of the following components, each of which is a parameterized, learned approximation of the original Dec-POMDP $F$:

- **Reward model**: $R_{\phi}(r_t|s_t, a_{t-1}, s_{t-1})$
- **Dynamics model**: $T_{\phi}(s_{t+1}|s_t, a_t)$
- **Observation model**: $O_{\phi}(o_t|s_t, a_{t-1})$
- **Termination model**: $P_{term}\phi(\text{termination}|s_t, a_t)$

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. $O(\alpha_j'|\alpha_{j-1}', a_{j-1}')$), 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\footnote{For ease of notation, the input to all components $\hat{f} \in \hat{F}$ in Algorithm 1 is written as $s_t, a_t$. 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 $\lambda$.
2. Initialize action-value functions $Q_0, Q_\phi$.
3. Initialize $m$ ensemble environment models $\hat{F}_i = \{R_{i,\phi}, T_{i,\phi}, O_{i,\phi}, P_{term,j,i}\}$ for $i = 1, \ldots, m$.
4. Initialize a dataset $D$ with samples collected using a random policy from the real environment $F$.
5. **repeat**
6. Train model $\hat{F}$ using dataset $D$.
7. Update $Q_\phi$ using model-free algorithm of choice within a randomly chosen model $\hat{F}_i$.
8. Update $Q_{\phi}$ using model-free algorithm of choice in a centralized fashion within a randomly chosen model $\hat{F}_i$ as follows:

\[
\hat{r}(s_t, a_t) = \sum_{j=1}^d \hat{r}(\hat{F}_i) = \sum_{k=1}^{n} \sum_{j=1}^{m} V_a((\hat{f}_{j,k},\psi)(s_t, a_t))
\]

\[
r_{\text{exploit}} = R_{\phi}(s_t, a_t, s_{t-1}) + \lambda \hat{r}(s_t, a_t)
\]

\[
y = r_{\text{exploit}} + y \max_{a' \in A} \sum_{j=1}^n Q(j, a', s_{j+1}, a'')
\]

\[
L(\phi) = \sum_{i=1}^h \left( y_i - \sum_{j=1}^n Q(s, a, j, \psi) \right)^2,
\]

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

rollout data using the model, we randomly sample which one of the ensemble models to generate from.

4.2 Model-Based MARL with Centralized Exploration Policy

MARCO collects data for model learning from a separate exploration policy $\pi_{\phi}$. 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 $\hat{r}$ in equation 2. To prevent the exploration policy from exploring 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 $\hat{r}$ and the reward generated by the reward model $R_{\phi}$. The hyper-parameter $\lambda$ 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
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 $0 \leq r(s,a) \leq 1$, 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 $\hat{P}(s,a), \hat{R}(s,a)$, 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 $\pi \in \Pi$ 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). $\hat{D}_K$ is the expected version of $\hat{D}_K$ where:

- $P_K(s' | s, a) = \begin{cases} P(s' | s, a) & \text{if } (s, a) \in K \\ \mathbb{1} [s' = s] & \text{otherwise} \end{cases}$
- $\hat{P}_K(s' | s, a) = \begin{cases} n(s,a,s') / n(s,a) & \text{if } (s, a) \in K \\ \mathbb{1} [s' = s] & \text{otherwise} \end{cases}$
- $R_K(s, a) = \begin{cases} R(s, a) & \text{if } (s, a) \in K \\ R_{\max} & \text{otherwise} \end{cases}$
- $\hat{R}_K(s, a) = \begin{cases} r(s,a) / n(s,a) & \text{if } (s, a) \in K \\ R_{\max} & \text{otherwise} \end{cases}$
- $O_K(s) = \begin{cases} O(s) & \text{if } (s, o) \in K \\ \text{random observation} & \text{otherwise} \end{cases}$
- $\hat{O}_K(s) = \begin{cases} o & \text{if } (s, o) > 0 \\ \text{random observation} & \text{otherwise} \end{cases}$

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

1. **Input:** $\gamma, m$.
2. **for** all $(s, o)$ do
3. $n(s, o) \leftarrow 0$
4. **for** all $(s, a)$ do
5. $r(s, a) \leftarrow 0$
6. $n(s, a) \leftarrow 0$
7. **for** all $s' \in S$ do
8. $n(s, a, s') \leftarrow 0$
9. **for** $t = 1, 2, 3, \ldots$ do
10. Let $s, o$ denote the state and observation at time $t$ respectively.
11. Choose action $a$ according to $\pi^*_D$
12. Let $r$ be the immediate reward and $s'$ the next state after executing action $a$ from state $s$.
13. if $n(s, a) < m$ then
14. $n(s, o) \leftarrow 1$ // Record observation
15. $n(s, a) \leftarrow n(s, a) + 1$
16. $r(s, a) \leftarrow r(s, a) + r$ // Record immediate reward
17. $n(s, a, s') \leftarrow n(s, a, s') + 1$ // Record immediate next-state
18. if $n(s, a) = m$ then
19. **for** all $\pi \in \Pi$ do
20. Obtain $\hat{J}_{DK}(\pi)$ using Monte Carlo rollouts in $\hat{D}_K$.
21. $\pi^*_D \leftarrow \arg \max_{\pi} \hat{J}_{DK}(\pi)$

**Definition 5.2.** Given a decentralized joint-policy $\pi$, we estimate its value $J(\pi)$ in a Dec-POMDP $D$, defined as the expected reward obtained by following the joint-policy in $D$, $J_D(\pi) = \mathbb{E}_{s \sim d_\pi} [V^\pi(s)]$. (3)

**Theorem 5.1.** Suppose that $0 \leq \epsilon < 1$ and $0 \leq \delta < 1$ are two real numbers and $D$ is any Dec-POMDP. There exists inputs $m = m(\frac{1}{\gamma}, \frac{1}{\delta})$ and $\epsilon$, satisfying $m \geq \frac{1}{\gamma} \frac{1}{\delta}$, such that if the Adapted R-MAX algorithm is executed on $D$ with inputs $m$ and $\epsilon$, then the following holds: Let $\pi^*_D$ denote the Adapted R-MAX’s policy. With probability at least $1 - \delta$, $J_D(\pi^*) - J_D(\pi^*_D) \leq 2\epsilon$ is true for all but $O \left( \frac{|S||A|}{(1 - \gamma)^2 \epsilon^3} \left( |S| + \ln \left( \frac{|S||A|}{\delta} \right) \frac{V^3 \ln 1}{\delta} \right) \right)$ 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) is heavily based on results from Jiang and Strehl et al., but for the first time extends them to the Dec-POMDP setting.
We evaluate the sample efficiency of MARCO against model-free Dec-POMDPs. 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 \(a_j^t \in \{“None”, “Tell”, “Turn on lights”, “Turn off lights”\}\); otherwise the only action is “None”. The episode ends when an agent chooses “Tell” or when the maximum timestep, \(T\), is reached. The reward \(r_j\) 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 \(o_j^t \in \{0, 1, 2, 3\}\) on the bridge, and its actions are \(a_j^t \in \{“Left”, “Right”, “End episode”\}\). 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 \(a_j^t \in \{“None”, “Tell”, “Turn on lights”, “Turn off lights”, “Left”, “Right”, “End episode”\}\). 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 \(\in \{“Left”, “Right”, “Up”, “Down”, “Do nothing”\}\), 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 \(\hat{F}\) 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 training.

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 $\epsilon$-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 $\epsilon$-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-6d).
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 $\lambda = 0$), 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 $\tilde{r}$ Figure 5b shows that when we do not use the exploration bonus term $\tilde{r}$ 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 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.
A Theoretical Results

A.1 Setting

We operate in the same Dec-POMDP setup as described in Section 3.1, but with four additional assumptions:

- Bounded rewards: \(\forall (s, a), 0 \leq R(s, a) \leq 1\). This allows us to introduce the notion of maximum reward and maximum value function as \(R_{\text{max}} = 1\) and \(V_{\text{max}} = \frac{1}{1-\gamma}\) respectively.
- Finite horizon setting: All trajectories end in \(H\) steps.
- Tabular setting: Discrete and finite number of states, actions and observations.
- Deterministic observation function.

A.2 Notation

In addition to the setting described in Section 3.1, we introduce a few more notations for our sample complexity analysis:

- \(\hat{s} \in \hat{S}\) is the joint state-action history, i.e. \(\hat{s}_2 = \{s_0, a_0, s_1, a_1, s_2\}\).
- \(\hat{r}\) is the trajectory of joint state-action histories, i.e. \(\hat{r}_2 = \{\hat{s}_0, a_0, \hat{s}_1, a_1, \hat{s}_2\}\).
- \(\pi : \hat{S} \rightarrow (O \times A)^* \rightarrow A\) is a deterministic "joint-policy" (see Figure 7). For simplicity, we absorb the deterministic observation function within the "joint-policy" (i.e. the first mapping \(\hat{S} \rightarrow (O \times A)^*\) in \(\pi\) is the deterministic observation function, and the second mapping \((O \times A)^* \rightarrow A\) in \(\pi\) is the deterministic joint-policy we actually learn). The second mapping is a set of decentralized policies that each maps from an individual AOH to an individual action. For the remainder of this section, when we talk about learning a near optimal joint-policy, we are referring to the actual set of decentralized joint-policies \((O \times A)^* \rightarrow A\). Hence, even though we denote \(\pi\) as a function that takes \(\hat{s}\) as input, we are still learning decentralized policies.

![Figure 7: Illustration of the joint-policy \(\pi\) at timestep \(t\) for a Dec-POMDP with \(N\) agents.](image)

- "\(D\)" denotes the Dec-POMDP, and "\(M\)" denotes the centralized MDP.
- The relationship between reward functions in a Dec-POMDP and in its corresponding centralized MDP can be expressed as follows due to the Markov property in MDPs:
  \[ R_D(s, a) = R_M(s, a). \]
- Similarly, the relationship between the transition functions in a Dec-POMDP and in its corresponding centralized MDP can be expressed as follows due to the Markov property in MDPs:
  \[ P_D(s' | s, a) = P_M(s' | s, a) \mathbb{1}_{\text{check history}}. \]

The indicator function \(\mathbb{1}_{\text{check history}}\) checks whether \(s'\)’s previous joint state-action history is the same as \(s\), and whether the joint-action \(a\) is the same as \(s'\)’s last joint-action. If the two conditions are not satisfied, the transition cannot happen because we can neither alter nor erase history (a rule that applies to both RL and life in general). For example, if \(\hat{s}' = (s'_0, a'_0, s'_1, a'_1, s'_2)\) and \(\hat{s} = (s'_0, a'_0, s'_1, a'_1, s'_2, s'_3)\), then,

\[ P_D(\hat{s}' | \hat{s}, a) = P_M(s'_1 | s'_0, a'_0) \mathbb{1}_{(s'_0 = s'_0)} \mathbb{1}_{(a'_0 = a'_0)} \mathbb{1}_{(s'_1 = s'_1)} \mathbb{1}_{(a'_1 = a'_1)} \mathbb{1}_{(s'_2 = s'_2)} \mathbb{1}_{(a = a'_1)} \]

- The value of a joint-policy \(\pi\) in a Dec-POMDP \(D\) is defined as the expected reward obtained by following the joint-policy in \(D\):
  \[ f(\pi) = \mathbb{E}_{s, a, \hat{s}} \left[ V^\pi(\hat{s}) \right] = \mathbb{E}_{s, a, \hat{s}} \left[ V^\pi(s) \right]. \]

For the remainder of the derivation, we will overload notation for simplicity. When \(P\) takes \(\hat{s}\) as input, we mean \(P_D\), and when \(P\) takes \(s\) as input, we mean \(P_M\). The same convention follows for \(\hat{P}\), \(R\), and \(\hat{R}\).
A.3 Algorithm
R-MAX [5] is a model-based RL (MBRL) algorithm for learning in MDPs. In this work, we adapt R-MAX to Dec-POMDPs (see Algorithm 2). We learn centralized models \( \hat{P}_M, \hat{R}_M, \hat{O} \) using empirical estimates. Using the learned models, we construct an approximate K-known Dec-POMDP \( \hat{D}_K \) (see Definition 5.1). Within \( \hat{D}_K \), we evaluate all possible joint-policies \( \pi \in \Pi \) and choose the best performing one. Our adapted R-MAX algorithm optimistically assigns rewards for all under-visited state action pairs to encourage exploration.

A.4 Basic Lemmas and Definitions
The rest of the derivation is heavily based on results from Jiang and Strehl et al., but for the first time extends them to the Dec-POMDP setting.

Lemma A.1 (Coin Flip [38]). Suppose a weighted coin, when flipped, has probability \( p > 0 \) of landing with heads up. Then, for any positive integer \( h \) and real number \( \delta \in (0, 1) \), there exists a number \( W = O((h/p) \ln(1/\delta)) \) such that after \( W \) tosses, with probability at least \( 1 - \delta \), we will observe \( h \) or more heads.

Proof. Results follow from the Chernoff-Hoeffding bound.

Definition A.1 (k-Known Dec-POMDP). \( D_K \) is the expected version of \( \hat{D}_K \) where:

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

\[
\hat{P}_K(s',s,a) = \begin{cases} \frac{n(s,a,s')}{n(s,a)} & \text{if } (s,a) \in K \\ \mathbb{I}[s' = s] & \text{otherwise} \end{cases}
\]

\[
R_K(s,a) = \begin{cases} R(s,a) & \text{if } (s,a) \in K \\ R_{\text{max}} & \text{otherwise} \end{cases}
\]

\[
\hat{R}_K(s,a) = \begin{cases} \frac{\sum n(s,a)}{n(s,a)} r(s,a) & \text{if } (s,a) \in K \\ R_{\text{max}} & \text{otherwise} \end{cases}
\]

\[
O_K(s) = \begin{cases} O(s) & \text{random observation} \\ \mathbb{0} & \text{otherwise} \end{cases}
\]

\[
\hat{O}_K(s) = \begin{cases} \mathbb{O} & \text{random observation} \\ \mathbb{0} & \text{otherwise} \end{cases}
\]

Fact A.1. Because the observation function is deterministic, we only need to see the observation once for a given \( (s,a) \) to learn an accurate model. When a given state-action pair \( \not\in K \), the observation model can be anything since it does not affect the outcome. This is because whatever action the agent performs, the K-known Dec-POMDP transition function will be a self-loop, and the reward will be 1 (i.e. \( R_{\text{max}} \)).

Table 1: Relationship between \( D, D_K \) and \( \hat{D}_K \)

| Known (K) | \( D \) | \( D_K \) | \( \hat{D}_K \) |
|---|---|---|---|
| Known (K) | \( = D \) | \( = D \) | \( = D \) |
| Unknown | \( \geq D \) | Self-loop, max reward, random observation | Self-loop, max reward, random observations |

Lemma A.2 (Reward Model Error [38]). Suppose \( m \) rewards are drawn independently from the reward distribution, \( R(s,a) \), for state-action pair \( (s,a) \). Let \( \hat{R}(s,a) \) be the empirical (maximum-likelihood) estimate of \( R(s,a) \). Let \( \delta_R \) be any positive real number less than 1. Then, with probability at least \( 1 - \delta_R \), we have that \( |\hat{R}(s,a) - R(s,a)| \leq \epsilon_R \), where

\[
\epsilon_R := \sqrt{\frac{\ln(2/\delta_R)}{2m}}.
\]

Proof. Results follows directly from Hoeffding’s bound.

Lemma A.3 (Transition Model Error [38]). Suppose \( m \) transitions are drawn independently from the transition distribution, \( P(s,a) \), for state-action pair \( (s,a) \). Let \( \hat{P}(s,a) \) be the empirical (maximum-likelihood) estimate of \( P(s,a) \). Let \( \delta_P \) be any positive real number less than 1. Then, with probability at least \( 1 - \delta_P \), we have that \( \|P(s,a) - \hat{P}(s,a)\|_1 \leq \epsilon_P \), where

\[
\epsilon_P := \sqrt{\frac{2}{m} \ln \left( \frac{2^2 - 2}{\delta_P} \right)}.
\]

Proof. The result follows immediately from an application of Theorem 2.1 of Weissman et al. (2003).
Corollary A.1. Suppose for all state-action pairs \((s, a) \in S \times A\), \(m\) transitions and \(m\) rewards are drawn independently from the transition distribution \(P(s, a)\) and the reward distribution \(R(s, a)\) respectively. Let \(\delta\) be any positive real number less than 1. Then, with probability at least \(1 - \delta\), for all state-action pairs \((s, a)\), we have \(|\hat{R}(s, a) - R(s, a)| \leq \varepsilon_R\) and \(\|P(s, a) - \hat{P}(s, a)\|_1 \leq \varepsilon_P\), where:

\[
\varepsilon_R := \sqrt{\frac{\ln (4S|A|)}{2m}}.
\]  

and

\[
\varepsilon_P := \sqrt{\frac{2}{m} \ln \left(\frac{2^{|S|} - 2}{2|S||A|}\right) / \delta}.
\]

Proof. The result is obtained directly by applying the union bound to Lemma A.2 and Lemma A.3, where we set \(\delta_P = \delta_R = \frac{\delta}{2|S||A|}\). We first split the failure probability evenly between the reward estimation events and transition estimation events. This results in the division by a factor of 2. Then, for the transition and the reward, we split \(\delta/2\) evenly among all state action pairs. This results in a further division by a factor of \(|S||A|\).

Fact A.2 (Optimism). By construction of the the Adapted R-MAX algorithm, \(\forall \pi : \hat{S} \rightarrow A, \quad J_{D_k} (\pi) \geq J_D (\pi)\).

Lemma A.4 (Simulation Lemma for Dec-POMDPs, Part 1). Let \(D\) and \(D^\prime\) be two Dec-POMDPs that only differ in \((P, R)\) and \((\hat{P}, \hat{R})\).

Let \(\varepsilon_R \geq \max_{s,a} |\hat{R}(s, a) - R(s, a)|\) and \(\varepsilon_P \geq \max_{s,a} |\hat{P}(s, a) - P(s, a)|\). Then \(\forall \pi : \hat{S} \rightarrow A,\)

\[
\frac{V^\pi_D - V^\pi_D}{\infty} \leq \varepsilon_R + 2\varepsilon_P \max_{s,a} (P(s, a) - \hat{P}(s, a)).
\]

Proof. For all \(\hat{s} \in \hat{S}\),

\[
\left| V^\pi_D (\hat{s}) - V^\pi_D (\hat{s}) \right| = \left| \hat{R}(\hat{s}, \pi) + \gamma \left( \hat{P}(\hat{s}, \pi) \cdot V^\pi_D (\hat{s}) - R(\hat{s}, \pi) - \gamma \left( P(\hat{s}, \pi), V^\pi_D (\hat{s}) \right) \right) \right|
\]

\[
= |\hat{R}(\hat{s}, \pi) - R(\hat{s}, \pi)| + \gamma \left| \left( \left( \hat{P}(\hat{s}, \pi) \cdot V^\pi_D (\hat{s}) \right) - \left( P(\hat{s}, \pi), V^\pi_D (\hat{s}) \right) \right) \right| \quad \text{(re-arranging terms)}
\]

\[
\leq \varepsilon_R + \gamma \left| \left( \left( \hat{P}(\hat{s}, \pi) \cdot V^\pi_D (\hat{s}) \right) - \left( P(\hat{s}, \pi), V^\pi_D (\hat{s}) \right) \right) \right| \quad \text{(add & subtract)}
\]

\[
\leq \varepsilon_R + \gamma \left| \left( \hat{P}(\hat{s}, \pi) - P(\hat{s}, \pi) \right) \cdot V^\pi_D (\hat{s}) \right| \quad \text{(add & subtract)}
\]

Because of Fact A.1, we need not deal with the observation function when writing out the value function, as we simply absorb the observation function as apart of the deterministic joint-policy.

Since Equation 9 holds for all \(\hat{s} \in \hat{S}\), we can take the infinite-norm on the left hand side:

\[
\left| V^\pi_D - V^\pi_D \right|_{\infty} \leq \varepsilon_R + \gamma \left| \left( \hat{P}(\hat{s}, \pi) - P(\hat{s}, \pi) \right) \cdot V^\pi_D \right| + \gamma \left| V^\pi_D - V^\pi_D \right|_{\infty}.
\]

We then expand the blue term as follows:

\[
\left| \left( \hat{P}(\hat{s}, \pi) - P(\hat{s}, \pi) \right) \cdot V^\pi_D \right| = \left| \left( P(\hat{s}, \pi) - P(\hat{s}, \pi) \right) \cdot V^\pi_D \right| \quad \text{(where 1 is a vector of ones \(\in \mathbb{R}^{|S|}\))}
\]

\[
\leq \left| \left( P(\hat{s}, \pi) - P(\hat{s}, \pi) \right) \cdot 1 \cdot \frac{R_{\text{max}}}{2(1 - \gamma)} \right| \quad \text{(Holder’s inequality)}
\]

\[
\leq \left| \left( P(s, \pi(\hat{s})) \cdot \text{check history} - P(s, \pi(\hat{s})) \cdot \text{check history} \right) \cdot 1 \cdot \frac{R_{\text{max}}}{2(1 - \gamma)} \right| \quad \text{(by definition of P)}
\]

\[
= \left| \left( P(s, \pi(\hat{s})) - P(s, \pi(\hat{s})) \right) \cdot \text{check history} \right| \cdot \left| \left( V^\pi_D - 1 \cdot \frac{R_{\text{max}}}{2(1 - \gamma)} \right) \right|_{\infty}
\]

\[
\leq \varepsilon_P \cdot \frac{R_{\text{max}}}{2(1 - \gamma)}
\]

\[
= \frac{\varepsilon_P \cdot \max_{s,a} (P(s, a) - \hat{P}(s, a))}{2}.
\]

In Equation 11, the step in line 1 shifts the range of \( V \) from \([0, \frac{R_{\text{max}}}{1 - \gamma}]\) to \([-\frac{R_{\text{max}}}{2(1 - \gamma)} \cdot \frac{R_{\text{max}}}{2(1 - \gamma)}]\) to obtain a tighter bound by a factor of 2. The equality in line 1 holds because of the following, where \( C \) is any constant:
\[
\langle \hat{P} - P, C \cdot 1 \rangle = C(\hat{P} - P)
\]
\[
= C \left( (\hat{P}, 1) - (P, 1) \right)
\]
\[
= C(1 - 1) \quad \text{because } P \text{ and } \hat{P} \text{ are probability distributions}
\]
\[
= 0
\]

From equation 12, we observe the equality in line 1 holds:

\[
(\hat{P} - P, V - C \cdot 1) = (\hat{P} - P, V) - (\hat{P} - P, C \cdot 1)
\]
\[
= (\hat{P} - P, V) - 0
\]
\[
= (\hat{P} - P, V).
\]

Finally, we plug equation 11 into equation 10 to obtain the bound.

\[
\left\| V^*_D - V^*_D \right\|_\infty \leq \epsilon_R + \gamma \epsilon_P \cdot \frac{V_{\text{max}}}{2} + \gamma \left\| V^*_D - V^*_D \right\|_\infty
\]
\[
= \epsilon_R + \gamma \frac{\epsilon_P V_{\text{max}}}{1 - \gamma} \cdot 2(1 - \gamma).
\]

**Lemma A.5 (Simulation Lemma for Dec-POMDPs, Part 2).** Let \( D \) and \( \hat{D} \) be two Dec-POMDPs that only differ in \((P, R)\) and \((\hat{P}, \hat{R})\), then the following holds:

Let \( \epsilon_R \geq \max_{s,a} |\hat{R}(s,a) - R(s,a)| \) and \( \epsilon_P \geq \max_{s,a} \|\hat{P}(\cdot | s, a) - P(\cdot | s, a)|\|_1 \), then \( \forall \pi : \hat{S} \rightarrow A \),

\[
\left\| V^*_D - V^*_D \right\|_\infty \leq \epsilon_R + \gamma \epsilon_P V_{\text{max}} / (2(1 - \gamma)).
\]

**Proof.** Let \( T_D, \hat{T}_D \) be the Bellman update operator of \( D \) and \( \hat{D} \) respectively.

\[
\left\| V^*_D - V^*_D \right\|_\infty = \left\| T_D V^*_D - \hat{T}_D V^*_D \right\|_\infty
\]
\[
= \max_{s,a \in S \times A} \left[ \hat{R}(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot | s,a)} \left[ V^*_{\hat{D}}(s') \right] - \hat{R}(s,a) - \gamma \mathbb{E}_{s' \sim \hat{P}(\cdot | s,a)} \left[ V^*_D(s') \right] \right]
\]
\[
= \max_{s,a \in S \times A} \left[ \hat{R}(s,a) - \hat{R}(s,a) + \max_{s,a \in S \times A} \left[ \mathbb{E}_{s' \sim P(\cdot | s,a)} \left[ V^*_{\hat{D}}(s') \right] - \gamma \mathbb{E}_{s' \sim \hat{P}(\cdot | s,a)} \left[ V^*_D(s') \right] \right] \right]
\]
\[
= \epsilon_R + \gamma \max_{s,a \in S \times A} \left\| P(\cdot | s,a) - \hat{P}(\cdot | s,a) \right\|_\infty \| V^*_D - V_{\text{max}} / 2 \cdot 1 \|_\infty
\]
\[
\leq \epsilon_R + \gamma \max_{s,a \in S \times A} \left\| P(\cdot | s,a) - \hat{P}(\cdot | s,a) \|_1 \right\| V^*_D - V_{\text{max}} / 2 \cdot 1 \|_\infty
\]
\[
\leq \epsilon_R + \gamma \epsilon_P \cdot V_{\text{max}} / 2.
\]

Therefore,

\[
\left\| V^*_D - V^*_D \right\|_\infty \leq \left\| T_D V^*_D - \hat{T}_D V^*_D \right\|_\infty
\]
\[
= \left\| T_D V^*_D - T_D V^*_D + T_D V^*_D - \hat{T}_D V^*_D \right\|_\infty \quad \text{(add and subtract)}
\]
\[
\leq \epsilon_R + \frac{\gamma \epsilon_P}{2(1 - \gamma)} \cdot V_{\text{max}} / 2 + \left\| T_D V^*_D - \hat{T}_D V^*_D \right\|_\infty \quad \text{(by equation 15)}
\]
\[
\leq \epsilon_R + \frac{\gamma \epsilon_P}{2(1 - \gamma)} \cdot V_{\text{max}} / 2 + \gamma \left\| V^*_D - V^*_D \right\|_\infty \quad \text{(by property of } V^*)
\]

**Corollary A.2.** Following from the two Simulation Lemmas for Dec-POMDPs, we have the following:

1. \( \forall \pi : J_{D_k} (\pi) - J_{D_k} (\pi) \leq \frac{\epsilon_R}{1 - \gamma} + \frac{\gamma \epsilon_P \epsilon_{D_k} \cdot V_{\text{max}}}{2(1 - \gamma)} \).
2. \( \forall \pi : J_{D_k} (\pi) - J_{D_k} (\pi) \leq \frac{\epsilon_R}{1 - \gamma} + \frac{\gamma \epsilon_P \epsilon_{D_k} \cdot V_{\text{max}}}{2(1 - \gamma)} \).

**Proof.** We give a proof for (2) as follows:

By definition, \( J(\pi) = \mathbb{E}_{s \sim d_\phi} [V^*_\pi] = \langle d_\phi, V^*_\pi \rangle \).
\[
\left| JD_K (\pi^*_D) - JD_K (\pi^*_K) \right| = \left| \langle d_0, V^*_D \rangle - \langle d_0, V^*_K \rangle \right|
\leq \langle d_0, 1 \cdot \| V^*_D - V^*_K \|_\infty \rangle \quad \text{where } 1 \text{ is the all ones vector}
\leq \| V^*_D - V^*_K \|_\infty \langle d_0, 1 \rangle
\]
(17)

The proof for (1) follows from the same steps as above using Lemma A.4.

\[ \square \]

**Lemma A.6 (Induced Inequality).** Suppose Dec-POMDPs $D$ and $D_K$ has centralized transition/reward/observation functions that agree exactly on $K \subseteq S \times A$. Let escape $\hat{\kappa} (\hat{\tau})$ be 1 if the trajectory $\hat{\tau}$ contains at least one $(s,a) \not\in K$, and 0 otherwise. $\forall \pi : \hat{S} \rightarrow A$,

\[
JD_K (\pi) - JD_K (\pi) \leq V_{\max} \cdot \mathbb{P}_D \left[ \text{escape}_K (\hat{\tau}) \mid \pi \right].
\]

Proof. Let $R_D (\hat{\tau})$ denote the sum of discounted rewards in $\hat{\tau}$ according to the reward function of $D$. We can write $V^*_D = \sum_{\hat{\tau}} \mathbb{P}_D [\hat{\tau} \mid \pi] R_D (\hat{\tau})$ (and similarly for $D_K$). For $\hat{\tau}$ such that escape $\hat{\kappa} (\hat{\tau}) = 1$, define prefix $\hat{\tau}$ as the prefix of $\hat{\tau}$ where every state-action pair in $\hat{\tau}$ is also in $K$, and that the next $\hat{\tau}$ contains one that escapes. Similarly define suffix $\hat{\tau}$ as the remainder of the episode. See Figure 8 for an illustration of an example. Let $R [\text{prefix}_K (\hat{\tau})]$ be the sum of discounted rewards within the prefix (or suffix), and $\mathbb{P}_D [\text{prefix}_K (\hat{\tau}) \mid \pi]$ be the marginal probability of the prefix (or suffix) assigned by $D_K$ under the joint-policy $\pi$. Because $JD_K \geq JD$ by optimism (Fact A.2), its sufficient to upper bound $JD_K (\pi) - JD (\pi)$.

$K = \{ \bullet, \bullet, \bullet, \bullet \}$

Trajectory = \[
[ ( \bullet, \bullet, \bullet, \bullet ), ( \bullet, \bullet, \bullet, \bullet ), ( \bullet, \bullet, \bullet, \bullet ), ( \bullet, \bullet, \bullet, \bullet ) ]
\]

prefix suffix

Figure 8: An example of partitioning a trajectory $\hat{\tau}$ into the prefix and the suffix. Different colours of a circle denote different state-action pairs, and each bracket in the trajectory is a state. Because purple in $s_2$ is the first time the trajectory sees a state-action pair not in the set $K$, so prefix = $[s_0, a_0, s_1, a_1]$, and suffix = $[s_2, a_2, s_3]$.

First, we give an upper bound on $JD_K$ using the fact that the cumulative discounted reward is upper-bounded by $V_{\max}$:

\[
\begin{align*}
JD_K (\pi) &= \sum_{\hat{\tau} : \text{prefix}_K (\hat{\tau})} \mathbb{P}_D [\hat{\tau} \mid \pi] R_D (\hat{\tau}) + \sum_{\hat{\tau} : \text{suffix}_K (\hat{\tau})} \mathbb{P}_D [\hat{\tau} \mid \pi] R_D (\hat{\tau}) \\
&= \sum_{\hat{\tau} : \text{prefix}_K (\hat{\tau})} \mathbb{P}_D [\hat{\tau} \mid \pi] (R_D (\text{prefix}_K (\hat{\tau})) + R_K (\text{suf}_K (\hat{\tau}))) + \sum_{\hat{\tau} : \text{suffix}_K (\hat{\tau})} \mathbb{P}_D [\hat{\tau} \mid \pi] R_D (\hat{\tau}) \\
&\leq \sum_{\hat{\tau} : \text{prefix}_K (\hat{\tau})} \mathbb{P}_D [\hat{\tau} \mid \pi] (R_D (\text{prefix}_K (\hat{\tau})) + V_{\max}) + \sum_{\hat{\tau} : \text{prefix}_K (\hat{\tau})} \mathbb{P}_D [\hat{\tau} \mid \pi] R_D (\hat{\tau}) \\
&\leq \sum_{\text{prefix}_K (\hat{\tau})} \mathbb{P}_D [\hat{\tau} \mid \pi] (R_D (\text{prefix}_K (\hat{\tau})) + V_{\max}) + \sum_{\hat{\tau} : \text{prefix}_K (\hat{\tau})} \mathbb{P}_D [\hat{\tau} \mid \pi] R_D (\hat{\tau}).
\end{align*}
\]
(18)

First line uses the fact that trajectories falls in either category - those that contain an escape, and those that do not. The second line follows from the fact that we can split each trajectory containing at least one escape into a prefix and suffix. The third line uses the fact that the prefix trajectory’s value is the same as the true Dec-POMDP $D$, and that the value of the suffix trajectory cannot be greater than $V_{\max}$. The last step uses the fact that for any $\hat{\tau}$ that shares the same prefix $\hat{\tau}$, we can combine their probabilities (because $R (\text{prefix}_K (\hat{\tau})) + V_{\max}$ does not depend on the suffix), and we get the marginal probability of the prefix.

Similarly, we give a lower bound on $JD_K$, this time using the fact that the cumulative discounted reward is lower-bounded by 0:

\[
\begin{align*}
JD (\pi) \geq \sum_{\text{prefix}_K (\hat{\tau})} \mathbb{P}_D [\hat{\tau} \mid \pi] R_D (\text{prefix}_K (\hat{\tau}) + \sum_{\hat{\tau} : \text{prefix}_K (\hat{\tau})} \mathbb{P}_D [\hat{\tau} \mid \pi] R_D (\hat{\tau}).
\end{align*}
\]
(19)

By the definition of $D_K$ and $D$, we have the following:
(1) \( \mathbb{P}_{D_K} [\text{pre}_K(\hat{t}) \mid \pi] = \mathbb{P}_D [\text{pre}_K(\hat{t}) \mid \pi] \).
(2) \( R_{D_K}(\text{pre}_K(\hat{t})) = R_D(\text{pre}_K(\hat{t})) \).
(3) \( \text{escape}_K(\hat{t}) = 0, \mathbb{P}_{D_K} [\hat{t} \mid \pi] = \mathbb{P}_D [\hat{t} \mid \pi] \).
(4) \( \text{escape}_K(\hat{t}) = 0, R_{D_K}(\hat{t}) = R_D(\hat{t}). \)

Combining the above, we obtain the bound.

\[
J_{D_K}(\pi) - J_D(\pi) \\
\leq \text{UpperBound}(J_{D_K}(\pi)) - \text{LowerBound}(J_D(\pi)) \\
= \sum_{\text{pre}_K(\hat{t})} \mathbb{P}_{D_K} [\text{pre}_K(\hat{t}) \mid \pi] (R_{D_K}(\text{pre}_K(\hat{t})) + V_{\max}) + \sum_{\hat{t} \text{escape}_K(\hat{t})=0} \mathbb{P}_D [\hat{t} \mid \pi] R_{D_K}(\hat{t}) \\
- \sum_{\text{pre}_K(\hat{t})} \mathbb{P}_D [\text{pre}_K(\hat{t}) \mid \pi] R_D(\text{pre}_K(\hat{t})) - \sum_{\hat{t} \text{escape}_K(\hat{t})=0} \mathbb{P}_D [\hat{t} \mid \pi] R_D(\hat{t}) \\
= \sum_{\text{pre}_K(\hat{t})} \mathbb{P}_{D_K} [\text{pre}_K(\hat{t}) \mid \pi] (R_{D_K}(\text{pre}_K(\hat{t})) + V_{\max}) - \sum_{\text{pre}_K(\hat{t})} \mathbb{P}_D [\text{pre}_K(\hat{t}) \mid \pi] (R_D(\text{pre}_K(\hat{t}))) \quad \text{(by (3) and (4))} \\
= \sum_{\text{pre}_K(\hat{t})} \mathbb{P}_D [\text{pre}_K(\hat{t}) \mid \pi] V_{\max} \quad \text{(by (1) and (2))} \\
= \mathbb{P}_D [\text{escape}(\hat{t}) \mid \pi] V_{\max}.
\]

A.5 Sample Complexity Analysis

**Theorem A.3.** Suppose that \( 0 < \varepsilon < 1 \) 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 \( \varepsilon \), satisfying \( m \left( \frac{1}{\varepsilon}, \frac{1}{\delta} \right) = O \left( \frac{(S|A|S|A|V_{\max}^2)}{\varepsilon^2(1-\gamma)^{2\varepsilon}} \right) \) such that if the Adapted R-MAX is executed on \( D \) with inputs \( m \) and \( \varepsilon \), then the following holds: Let \( \pi_{{D_K}}^* \) denote the Adapted R-MAX's joint-policy. With probability at least \( 1 - \delta \), \( J_D(\pi^*) - J_D(\pi_{{D_K}}^*) \leq 2\varepsilon \) is true for all but \( O \left( \frac{|S||A|}{(1-\gamma)^2\varepsilon^3} \left( |S| + \ln \left( \frac{|S||A|}{\varepsilon^3} \right) \right) V_{\max}^3 \ln \frac{1}{\delta} \right) \) episodes.

**Proof.**

\[
J_D(\pi_{{D_K}}^*) - J_D(\pi_{{D_K}}^*) \\
\leq J_{D_K}(\pi_{{D_K}}^*) - J_D(\pi_{{D_K}}^*) \quad \text{(by optimism in lemma A.2)} \\
\leq J_{D_K}(\pi_{{D_K}}^*) - J_D(\pi_{{D_K}}^*) \quad \text{(by definition of greedy policy)} \\
\leq J_{D_K}(\pi_{{D_K}}^*) + \frac{\varepsilon R}{1-\gamma} + \frac{\varepsilon}{1-\gamma} V_{\max}^3 - J_D(\pi_{{D_K}}^*) \quad \text{(by (2) in Corollary A.2)} \\
\leq J_{D_K}(\pi_{{D_K}}^*) + 2 \frac{\varepsilon R}{1-\gamma} + \frac{\varepsilon}{1-\gamma} V_{\max}^3 - J_D(\pi_{{D_K}}^*) \quad \text{(by (1) in corollary A.2)} \\
\leq 2 \frac{\varepsilon R}{1-\gamma} + \frac{\varepsilon}{1-\gamma} V_{\max}^3 + \mathbb{P}_D [\text{escape}(\hat{t}) \mid \pi] V_{\max} \quad \text{(by Lemma A.6)}.
\]

In corollary A.2, we essential say \( |J_1 - J_2| \leq \Delta \). We then have the following upper bound \( J_1 \leq J_2 + \Delta \). This is because either of the two cases happens:

1. Case 1: \( J_1 \geq J_2 \):

\[
|J_1 - J_2| \leq \Delta \\
\Rightarrow J_1 - J_2 \leq \Delta \\
\Rightarrow J_1 \leq J_2 + \Delta
\]

2. Case 2: \( J_1 < J_2 \Rightarrow J_1 \leq J_2 + \Delta \) because \( \Delta \leq 0 \).

In line 3 and 4 of equation 21, we essential just apply the upper bound \( J_1 \leq J_2 + \Delta \).

To choose \( m \) large enough such that \( 2 \frac{\varepsilon R}{1-\gamma} + \frac{\varepsilon}{1-\gamma} V_{\max}^3 \leq \varepsilon \) with high probability, by Corollary A.1, we need the following two conditions to hold:

\[
\sqrt{\frac{\ln (4|S||A|/\varepsilon R)}{2m}} \frac{2}{1-\gamma} \leq \varepsilon \quad \text{(22)}
\]

\[
\sqrt{\frac{2^m \ln (2^2 - 2)}{2|S||A|/\varepsilon R}} \frac{V_{\max}}{(1-\gamma)} \leq \varepsilon \quad \text{(23)}
\]
Solving for \( m \), we see that when \( m \geq CV_{\max}^2 \left( \frac{S\ln(S/A)}{\gamma^2(1-\gamma)} \right) \), both conditions are satisfied (\( C \) is a constant).

There are two possible cases for the remaining term \( P_D [\text{escape}(\pi)] \):

1. Suppose that \( P_D [\text{escape}(\pi)] < \frac{\epsilon}{V_{\max}} \), then the joint-policy is \( 2\epsilon \)-optimal.
2. Suppose \( P_D [\text{escape}(\pi)] \geq \frac{\epsilon}{V_{\max}} \), then the following occurs: Because successful exploration occurs at most \( m|S||A| \) times, by Lemma A.1, we need \( W \) episodes with attempts of exploration,
\[
W = O \left( \frac{m|S||A|}{P_D [\text{escape}(\pi)]} \ln \frac{1}{\delta} \right)
= O \left( \frac{|S||A|}{(1-\gamma)^2 \epsilon^3} \left[ |S| + \ln \left( \frac{|S||A|}{\delta} \right) \right] V_{\max}^3 \ln \frac{1}{\delta} \right).
\] (24)

\[\square\]

### B Experiment Details

#### B.1 Policy Optimization

Implementation of model-free baseline and MARCO is based on the pymarl codebase [34].

In each experiment, all agents share the same network. To distinguish between each agent, a one-hot vector of agent ID is concatenated on to each agent’s observation. The architecture of all agent networks is a DRQN with a recurrent layer consisting of a fully-connected layer, followed by a GRU with hidden dimension of 64 units, then another fully-connected layer. We use a \( \gamma = 0.99 \) for all experiments. The replay buffer contains the most recent 5000 episodes. During training, we sample a batch of 32 episodes uniformly from the buffer each time, and the target network is updated every 200 episodes. When training the agent’s policy, we encourage exploration by using an \( \epsilon \)-greedy policy, where \( \epsilon \) anneals linearly from 1.0 to 0.05 over 100k training timestamps. after 100k timesteps, \( \epsilon \) remains constant at 0.05. All neural nets during policy optimization is trained using RMSprop with learning rate of \( 1 \times 10^{-4} \).

The exploration policy is trained in the same manner as described above. Whenever a new model is learned, the exploration policy is trained for 100k timesteps within the model in the switch tasks, and for 50k in the MPE task. The input to the exploration policy uses the state instead of the agent’s observation.

#### B.2 Data Collection

The dataset for model learning is always initialized with data collected using a randomly behaving policy. After the initialization, we use an \( \epsilon \)-greedy data-collection policy, where \( \epsilon \) is constant with value of 0.1.

#### B.3 Model Learning

The dynamics model for both switch tasks is an auto-regressive model. An encoder with a single hidden layer of 500 units encodes the past state-action pair. The encoded state-action pair is then used to generate the next state using a recurrent layer composed of a fully connected layer, followed by a GRU with hidden dimension of 500 units, and two more fully connected layers. The dynamics model for the MPE task uses a fully connected neural network of two hidden layers of 500 units to predict the categorical state features and continuous state features separately. All other model components are parameterized by neural networks with two fully connected layers of 500 units.

All model components are trained with the Adam optimizer, learning rate of 0.001, and batch size of 1000. To prevent overfitting, we use early stopping with a maximum of 700 epochs. The training and validation dataset is split between 3:7 respectively.

**Switch Tasks** We learn an ensemble of 5 models over the dynamics model, and 1 model for all other models components. We also learn a model for the available actions function to narrow down the joint-policy search space. At each timestep, the available actions function outputs a set of available actions.

\[
\text{Available actions model} : \quad P_{\text{available}}(\{a\}_{j=1:N}|s_t)
\]

The model-free baseline have access to the ground truth available actions function during both training and testing for a fair comparison.

**MPE** We learn an ensemble of 3 models over the dynamics model and the reward model.

#### B.4 Compute

All experiments are done on a GPU of either NVIDIA T4 or NVIDIA P100. Experiments took < 12 hours to run.