Deep Decentralized Multi-task Multi-Agent Reinforcement Learning under Partial Observability

Shayegan Omidshafiei [1] Jason Pazis [1] Christopher Amato [2] Jonathan P. How [1] John Vian [3]

Abstract

Many real-world tasks involve multiple agents with partial observability and limited communication. Learning is challenging in these settings due to local viewpoints of agents, which perceive the world as non-stationary due to concurrently-exploring teammates. Approaches that learn specialized policies for individual tasks face problems when applied to the real world: not only do agents have to learn and store distinct policies for each task, but in practice identities of tasks are often non-observable, making these approaches inapplicable. This paper formalizes and addresses the problem of multi-task multi-agent reinforcement learning under partial observability. We introduce a decentralized single-task learning approach that is robust to concurrent interactions of teammates, and present an approach for distilling single-task policies into a unified policy that performs well across multiple related tasks, without explicit provision of task identity.

1. Introduction

In multi-task reinforcement learning (MTRL) agents are presented several related *target* tasks (Taylor & Stone, 2009; Caruana, 1998) with shared characteristics. Rather than specialize on a single task, the objective is to generalize performance across all tasks. For example, a team of autonomous underwater vehicles (AUVs) learning to detect and repair faults in deep-sea equipment must be able to do so in many settings (varying water currents, lighting, etc.), not just under the circumstances observed during training.

Many real-world problems involve multiple agents with partial observability and limited communication (e.g., the AUV example) (Oliehoek & Amato, 2016), but generating accurate models for these domains is difficult due to complex interactions between agents and the environment. Learning is difficult in these settings due to partial observability and local viewpoints of agents, which perceive the environment as non-stationary due to teammates' actions. Efficient learners must extract knowledge from past tasks to accelerate learning and improve generalization to new tasks. Learning specialized policies for individual tasks can be problematic, as not only do agents have to store a distinct policy for each task, but in practice face scenarios where the identity of the task is often non-observable.

Existing MTRL methods focus on single-agent and/or fully observable settings (Taylor & Stone, 2009). By contrast, this work considers cooperative, independent learners operating in partially-observable, stochastic environments, receiving feedback in the form of local noisy observations and joint rewards. This setting is general and realistic for many multi-agent domains. We introduce the multi-task multi-agent reinforcement learning (MT-MARL) under partial observability problem, where the goal is to maximize execution-time performance on a set of related tasks, without explicit knowledge of the task identity. Each MT-MARL task is formalized as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) (Bernstein et al., 2002), a general formulation for cooperative decision-making under uncertainty. MT-MARL poses significant challenges, as each agent must learn to coordinate with teammates to achieve good performance, ensure policy generalization across *all* tasks, and conduct (implicit) execution-time inference of the underlying task ID to make sound decisions using local noisy observations. As typical in existing MTRL approaches, this work focuses on average asymptotic performance across all tasks (Caruana, 1998; Taylor & Stone, 2009) and sample-efficient learning.

We propose a two-phase MT-MARL approach that first uses cautiously-optimistic learners in combination with Deep Recurrent Q-Networks (DRQNs) (Hausknecht & Stone, 2015) for action-value approximation. We introduce Concurrent Experience Replay Trajectories (CERTs), a decentralized extension of ex-

[1]Laboratory for Information and Decision Systems (LIDS), MIT, Cambridge, MA, USA [2]College of Computer and Information Science (CCIS), Northeastern University, Boston, MA, USA [3]Boeing Research & Technology, Seattle, WA, USA. Correspondence to: Shayegan Omidshafiei <shayegan@mit.edu>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

perience replay (Lin, 1992; Mnih et al., 2015) targeting sample-efficient and stable MARL. This first contribution enables coordination in single-task MARL under partial observability. The second phase of our approach distills each agent's specialized action-value networks into a generalized recurrent multi-task network. Using CERTs and optimistic learners, well-performing distilled policies (Rusu et al., 2015) are learned for multi-agent domains. Both the single-task and multi-task phases of the algorithm are demonstrated to achieve good performance on a set of multi-agent target capture Dec-POMDP domains. The approach makes no assumptions about communication capabilities and is fully decentralized during learning and execution. To our knowledge, this is the first formalization of decentralized MT-MARL under partial observability.

2. Background

2.1. Reinforcement Learning

Single-agent RL under full observability is typically formalized using Markov Decision Processes (MDPs) (Sutton & Barto, 1998), defined as tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$. At timestep t, the agent with state $s \in \mathcal{S}$ executes action $a \in \mathcal{A}$ using policy $\pi(a|s)$, receives reward $r_t = \mathcal{R}(s) \in \mathbb{R}$, and transitions to state $s' \in \mathcal{S}$ with probability $P(s'|s,a) = \mathcal{T}(s,a,s')$. Denoting discounted return as $R_t = \sum_{t'=t}^{H} \gamma^{t'-t} r_t$, with horizon H and discount factor $\gamma \in [0,1)$, the action-value (or Q-value) is defined as $Q^\pi(s,a) = \mathbb{E}_\pi[R_t|s_t = s, a_t = a]$. Optimal policy π^* maximizes the Q-value function, $Q^{\pi^*}(s,a) = \max_\pi Q(s,a)$. In RL, the agent interacts with the environment to learn π^* *without* explicit provision of the MDP model. Model-based methods first learn $\mathcal{T}$ and $\mathcal{R}$, then use a planner to find Q^{π^*}. Model-free methods typically directly learn Q-values or policies, so can be more space and computation efficient.

Q-learning (Watkins & Dayan, 1992) iteratively estimates the optimal Q-value function using backups, $Q(s,a) = Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$, where $\alpha \in [0,1)$ is the learning rate and the term in brackets is the temporal-difference (TD) error. Convergence to Q^{π^*} is guaranteed in the tabular (no approximation) case provided sufficient state/action space exploration; however, tabulated learning is unsuitable for problems with large state/action spaces. Practical TD methods instead use function approximators (Gordon, 1995) such as linear combinations of basis functions or neural networks, leveraging inductive bias to execute similar actions in similar states. Deep Q-learning is a state-of-the-art approach using a Deep Q-Network (DQN) for Q-value approximation (Mnih et al., 2015). At each iteration j, experience tuple $\langle s, a, r, s' \rangle$ is sampled from replay memory $\mathcal{M}$ and DQN parameters θ are updated to minimize loss $L_j(\theta_j) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{M}}[(r + \gamma \max_{a'} Q(s',a';\hat{\theta}_j) - Q(s,a;\theta_j))^2]$. Replay memory $\mathcal{M}$ is a first-in first-out queue containing the set of latest experience tuples from ϵ-greedy policy execution. *Target network* parameters $\hat{\theta}_j$ are updated less frequently and, in combination with experience replay, are critical for stable deep Q-learning.

Agents in partially-observable domains receive observations of the latent state. Such domains are formalized as Partially Observable Markov Decision Processes (POMDPs), defined as $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, \mathcal{O}, \gamma \rangle$ (Kaelbling et al., 1998). After each transition, the agent observes $o \in \Omega$ with probability $P(o|s',a) = \mathcal{O}(o,s',a)$. Due to noisy observations, POMDP policies map observation histories to actions. As Recurrent Neural Networks (RNNs) inherently maintain an internal state h_t to compress input history until timestep t, they have been demonstrated to be effective for learning POMDP policies (Wierstra et al., 2007). Recent work has introduced Deep Recurrent Q-Networks (DRQNs) (Hausknecht & Stone, 2015), combining Long Short-Term Memory (LSTM) cells (Hochreiter & Schmidhuber, 1997) with DQNs for RL in POMDPs. Our work extends this single-task, single-agent approach to the multi-task, multi-agent setting.

2.2. Multi-agent RL

Multi-agent RL (MARL) involves a set of agents in a shared environment, which must learn to maximize their individual returns (Buşoniu et al., 2010). Our work focuses on cooperative settings, where agents share a joint return. Claus & Boutilier (1998) dichotomize MARL agents into two classes: Joint Action Learners (JALs) and Independent Learners (ILs). JALs observe actions taken by *all* agents, whereas ILs only observe *local* actions. As observability of joint actions is a strong assumption in partially observable domains, ILs are typically more practical, despite having to solve a more challenging problem (Claus & Boutilier, 1998). Our approach utilizes ILs that conduct both learning *and* execution in a decentralized manner.

Unique challenges arise in MARL due to agent interactions during learning (Buşoniu et al., 2010; Matignon et al., 2012). Multi-agent domains are non-stationary from agents' local perspectives, due to teammates' interactions with the environment. ILs, in particular, are susceptible to *shadowed equilibria*, where local observability and non-stationarity cause locally optimal actions to become a globally sub-optimal joint action (Fulda & Ventura, 2007). Effective MARL requires each agent to tightly coordinate with fellow agents, while also being robust against destabilization of its own policy due to environmental non-stationarity. Another desired characteristic is robustness to *alter-exploration*, or drastic changes in policies due to exploratory actions of teammates (Matignon et al., 2012).

2.3. Transfer and Multi-Task Learning

Transfer Learning (TL) aims to generalize knowledge from a set of *source tasks* to a *target task* (Pan & Yang, 2010). In single-agent, fully-observable RL, each task is formalized as a distinct MDP (i.e., MDPs and tasks are synonymous) (Taylor & Stone, 2009). While TL assumes sequential transfer, where source tasks have been previously learned and even may not be related to the target task, Multi-Task Reinforcement Learning (MTRL) aims to learn a policy that performs well on related target tasks from an underlying task distribution (Caruana, 1998; Pan & Yang, 2010). MTRL tasks can be learned simultaneously or sequentially (Taylor & Stone, 2009). MTRL directs the agent's attention towards pertinent training signals learned on individual tasks, enabling a *unified* policy to generalize well across all tasks. MTRL is most beneficial when target tasks share common features (Wilson et al., 2007), and most challenging when the task ID is not explicitly specified to agents during execution – the setting addressed in this paper.

3. Related Work

3.1. Multi-agent RL

Buşoniu et al. (2010) present a taxonomy of MARL approaches. Partially-observable MARL has received limited attention. Works include model-free gradient-ascent based methods (Peshkin et al., 2000; Dutech et al., 2001), simulator-supported methods to improve policies using a series of linear programs (Wu et al., 2012), and model-based approaches where agents learn in an interleaved fashion to reduce destabilization caused by concurrent learning (Banerjee et al., 2012). Recent scalable methods use Expectation Maximization to learn finite state controller (FSC) policies (Wu et al., 2013; Liu et al., 2015; 2016).

Our approach is most related to IL algorithms that learn Q-values, as their representational power is more conducive to transfer between tasks, in contrast to policy tables or FSCs. The majority of existing IL approaches assume full observability. Matignon et al. (2012) survey these approaches, the most straightforward being Decentralized Q-learning (Tan, 1993), where each agent performs independent Q-learning. This simple approach has some empirical success (Matignon et al., 2012). Distributed Q-learning (Lauer & Riedmiller, 2000) is an optimal algorithm for deterministic domains; it updates Q-values only when they are guaranteed to increase, and the policy only for actions that are no longer greedy with respect to Q-values. Bowling & Veloso (2002) conduct Policy Hill Climbing using the Win-or-Learn Fast heuristic to decrease (increase) each agent's learning rate when it performs well (poorly). Frequency Maximum Q-Value heuristics (Kapetanakis & Kudenko, 2002) bias action selection towards those consistently achieving max rewards. Hysteretic Q-learning (Matignon et al., 2007) addresses miscoordination using cautious optimism to stabilize policies while teammates explore. Its track record of empirical success against complex methods (Xu et al., 2012; Matignon et al., 2012; Barbalios & Tzionas, 2014) leads us to use it as a foundation for our MT-MARL approach. Foerster et al. (2016) present architectures to learn communication protocols for Dec-POMDP RL, noting best performance using a centralized approach with inter-agent backpropagation and parameter sharing. They also evaluate a model combining Decentralized Q-learning with DRQNs, which they call Reinforced Inter-Agent Learning. Given the decentralized nature of this latter model (called Dec-DRQN herein for clarity), we evaluate our method against it. Concurrent to our work, Foerster et al. (2017) investigated an alternate means of stabilizing experience replay for the centralized learning case.

3.2. Transfer and Multi-task RL

Taylor & Stone (2009) and Torrey & Shavlik (2009) provide excellent surveys of transfer and multi-task RL, which almost exclusively target single-agent, fully-observable settings. Tanaka & Yamamura (2003) use first and second-order statistics to compute a prioritized sweeping metric for MTRL, enabling an agent to maximize lifetime reward over task sequences. Fernández & Veloso (2006) introduce an MDP policy similarity metric, and learn a *policy library* that generalizes well to tasks within a shared domain. Wilson et al. (2007) consider TL for MDPs, learning a Dirichlet Process Mixture Model over source MDPs, used as an informative prior for a target MDP. They extend the work to multi-agent MDPs by learning characteristic agent roles (Wilson et al., 2008). Brunskill & Li (2013) introduce an MDP clustering approach that reduces negative transfer in MTRL, and prove reduction of sample complexity of exploration using transfer. Taylor et al. (2013) introduce *parallel transfer* to accelerate multi-agent learning using inter-agent transfer. Recent work extends the notion of neural network distillation (Hinton et al., 2015) to DQNs for single-agent, fully-observable MTRL, first learning a set of specialized teacher DQNs, then distilling teachers to a single multi-task network (Rusu et al., 2015). The efficacy of the distillation technique for single-agent MDPs with large state spaces leads our work to use it as a foundation for the proposed MT-MARL under partial observability approach.

4. Multi-task Multi-agent RL

This section introduces MT-MARL under partial observability. We formalize single-task MARL using the Decentralized Partially Observable Markov Decision Process (Dec-POMDP), defined as $\langle \mathcal{I}, \mathcal{S}, \boldsymbol{\mathcal{A}}, \mathcal{T}, \mathcal{R}, \boldsymbol{\Omega}, \mathcal{O}, \gamma \rangle$, where $\mathcal{I}$ is a set of n agents, $\mathcal{S}$ is the state space, $\boldsymbol{\mathcal{A}} =$

$\times_i \mathcal{A}^{(i)}$ is the joint action space, and $\boldsymbol{\Omega} = \times_i \Omega^{(i)}$ is the joint observation space (Bernstein et al., 2002).[1] Each agent i executes action $a^{(i)} \in \mathcal{A}^{(i)}$, where joint action $\boldsymbol{a} = \langle a^{(1)}, \ldots, a^{(n)} \rangle$ causes environment state $s \in \mathcal{S}$ to transition with probability $P(s'|s, \boldsymbol{a}) = \mathcal{T}(s, \boldsymbol{a}, s')$. At each timestep, each agent receives observation $o^{(i)} \in \Omega^{(i)}$, with joint observation probability $P(\boldsymbol{o}|s', \boldsymbol{a}) = \mathcal{O}(\boldsymbol{o}, s', \boldsymbol{a})$, where $\boldsymbol{o} = \langle o^{(1)}, \ldots, o^{(n)} \rangle$. Let local observation history at timestep t be $\vec{o}_t^{\,(i)} = (o_1^{(i)}, \ldots, o_t^{(i)})$, where $\vec{o}_t^{\,(i)} \in \vec{O}_t^{\,(i)}$. Single-agent policy $\pi^{(i)} : \vec{O}_t^{\,(i)} \mapsto A^{(i)}$ conducts action selection, and the joint policy is denoted $\boldsymbol{\pi} = \langle \pi^{(1)}, \ldots, \pi^{(n)} \rangle$. For simplicity, we consider only pure joint policies, as finite-horizon Dec-POMDPs have at least one pure joint optimal policy (Oliehoek et al., 2008). The team receives a joint reward $r_t = \mathcal{R}(s_t, \boldsymbol{a}_t) \in \mathbb{R}$ at each timestep t, the objective being to maximize the value (or expected return), $V = \mathbb{E}[\sum_{t=0}^{H} \gamma^t r_t]$. While Dec-POMDP *planning* approaches assume agents do not observe intermediate rewards, we make the typical RL assumption that they do. This assumption is consistent with prior work in MARL (Banerjee et al., 2012; Peshkin et al., 2000).

ILs provide a scalable way to learn in Dec-POMDPs, as each agent's policy maps local observations to actions. However, the domain appears *non-stationary* from the perspective of each Dec-POMDP agent, a property we formalize by extending the definition by Laurent et al. (2011).

Definition 1. *Let $\boldsymbol{a}^{-(i)} = \boldsymbol{a} \setminus \{a^{(i)}\}$. Local decision process for agent i is stationary if, for all timesteps $t, u \in \mathbb{N}$,*

$$\sum_{\boldsymbol{a}_t^{-(i)} \in \boldsymbol{\mathcal{A}}^{-(i)}} P(s'|s, \langle a^{(i)}, \boldsymbol{a}_t^{-(i)} \rangle) = \sum_{\boldsymbol{a}_u^{-(i)} \in \boldsymbol{\mathcal{A}}^{-(i)}} P(s'|s, \langle a^{(i)}, \boldsymbol{a}_u^{-(i)} \rangle), \quad (1)$$

and

$$\sum_{\boldsymbol{a}_t^{-(i)} \in \boldsymbol{\mathcal{A}}^{-(i)}} P(o^{(i)}|s', \langle a^{(i)}, \boldsymbol{a}_t^{-(i)} \rangle) = \sum_{\boldsymbol{a}_u^{-(i)} \in \boldsymbol{\mathcal{A}}^{-(i)}} P(o^{(i)}|s', \langle a^{(i)}, \boldsymbol{a}_u^{-(i)} \rangle). \quad (2)$$

Letting $\boldsymbol{\pi}^{-(i)} = \boldsymbol{\pi} \setminus \{\pi^{(i)}\}$, non-stationarity from the local perspective of agent i follows as in general $\boldsymbol{a}_t^{-(i)} = \boldsymbol{\pi}^{-(i)}(\vec{\boldsymbol{o}}_t) \neq \boldsymbol{\pi}^{-(i)}(\vec{\boldsymbol{o}}_u) = \boldsymbol{a}_u^{-(i)}$, which causes violation of (1) and (2). Thus, MARL extensions of single-agent algorithms that assume stationary environments, such as Dec-DRQN, are inevitably ill-fated. This motivates our decision to first design a single-task, decentralized MARL approach targeting non-stationarity in Dec-POMDP learning.

The MT-MARL problem in partially observable settings is now introduced by extending the single-agent, fully-observable definition of Fernández & Veloso (2006).

Definition 2. *A partially-observable MT-MARL Domain $\mathcal{D}$ is a tuple $\langle \mathcal{I}, \mathcal{S}, \boldsymbol{\mathcal{A}}, \boldsymbol{\Omega}, \gamma \rangle$, where $\mathcal{I}$ is the set of agents, $\mathcal{S}$ is the environment state space, $\boldsymbol{\mathcal{A}}$ is the joint action space, $\boldsymbol{\Omega}$ is the joint observation space, and γ is the discount factor.*

Definition 3. *A partially-observable MT-MARL Task T_j is a tuple $\langle \mathcal{D}, \mathcal{T}_j, \mathcal{R}_j, \mathcal{O}_j \rangle$, where $\mathcal{D}$ is a shared underlying domain; $\mathcal{T}_j$, $\mathcal{R}_j$, $\mathcal{O}_j$ are, respectively, the task-specific transition, reward, and observation functions.*

In MT-MARL, each episode $e \in \{1, \ldots, E\}$ consists of a randomly sampled Task T_j from domain $\mathcal{D}$. The team observes the task ID, j, during learning, but not during execution. The objective is to find a joint policy that maximizes average empirical execution-time return in all E episodes, $\bar{V} = \frac{1}{E} \sum_{e=0}^{E} \sum_{t=0}^{H_e} \gamma^t \mathcal{R}_e(s_t, \boldsymbol{a}_t)$, where H_e is the time horizon of episode e.

5. Approach

This section introduces a two-phase approach for partially-observable MT-MARL; the approach first conducts single-task specialization, and subsequently unifies task-specific DRQNs into a joint policy that performs well in all tasks.

5.1. Phase I: Dec-POMDP Single-Task MARL

As Dec-POMDP RL is notoriously complex (and solving for the optimal policy is NEXP-complete even with a known model (Bernstein et al., 2002)), we first introduce an approach for stable single-task MARL. This enables agents to learn coordination, while also learning Q-values needed for computation of a unified MT-MARL policy.

5.1.1. Decentralized Hysteretic Deep Recurrent Q-Networks (Dec-HDRQNs)

Due to partial observability and local non-stationarity, model-based Dec-POMDP MARL is extremely challenging (Banerjee et al., 2012). Our approach is model-free and decentralized, learning Q-values for each agent. In contrast to policy tables or FSCs, Q-values are amenable to the multi-task distillation process as they inherently measure quality of all actions, rather than just the optimal action.

Overly-optimistic MARL approaches (e.g., Distributed Q-learning (Lauer & Riedmiller, 2000)) completely ignore low returns, which are assumed to be caused by teammates' exploratory actions. This causes severe overestimation of Q-values in stochastic domains. Hysteretic Q-learning (Matignon et al., 2007), instead, uses the insight that low returns may also be caused by domain stochasticity, which should not be ignored. This approach uses two learning rates: nominal learning rate, α, is used when the TD-error is non-negative; a smaller learning rate, β, is used otherwise (where $0 < \beta < \alpha < 1$). The result is hysteresis (lag) of Q-value degradation for actions associated with positive past experiences that occurred due to successful

[1]Superscripts indicate *local* parameters for agent $i \in \mathcal{I}$.

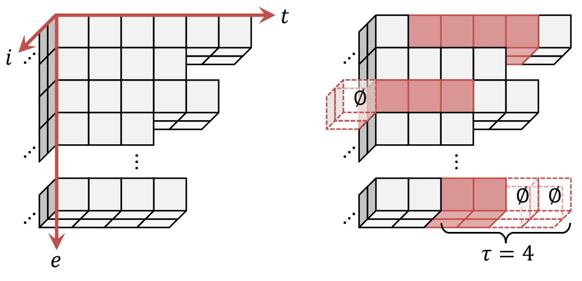
cooperation. Agents are, therefore, robust against negative learning due to teammate exploration and concurrent actions. Notably, unlike Distributed Q-learning, Hysteretic Q-learning permits eventual degradation of Q-values that were overestimated due to outcomes unrelated to their associated action.

Hysteretic Q-learning has enjoyed a strong empirical track record in fully-observable MARL (Xu et al., 2012; Matignon et al., 2012; Barbalios & Tzionas, 2014), exhibiting similar performance as more complex approaches. Encouraged by these results, we introduce Decentralized Hysteretic Deep Recurrent Q-Networks (Dec-HDRQNs) for partially-observable domains. This approach exploits the robustness of hysteresis to non-stationarity and alter-exploration, in addition to the representational power and memory-based decision making of DRQNs. As later demonstrated, Dec-HDRQN is well-suited to Dec-POMDP MARL, as opposed to non-hysteretic Dec-DRQN.

5.1.2. Concurrent Experience Replay Trajectories (CERTs)

Experience replay (sampling a memory bank of experience tuples $\langle s, a, r, s' \rangle$ for TD learning) was first introduced by Lin (1992) and recently shown to be crucial for stable deep Q-learning (Mnih et al., 2015). With experience replay, sampling cost is reduced as multiple TD updates can be conducted using each sample, enabling rapid Q-value propagation to preceding states *without* additional environmental interactions. Experience replay also breaks temporal correlations of samples used for Q-value updates—crucial for reducing generalization error, as the stochastic optimization algorithms used for training DQNs typically assume i.i.d. data (Bengio, 2012).

Despite the benefits in single-agent settings, existing MARL approaches have found it necessary to disable experience replay (Foerster et al., 2016). This is due to the non-concurrent (and non-stationary) nature of local experiences when sampled independently for each agent, despite the agents learning concurrently. A contributing factor is that inter-agent desynchronization of experiences compounds the prevalence of earlier-mentioned shadowed equilibria challenges, destabilizing coordination. As a motivating example, consider a 2 agent game where $\mathcal{A}^{(1)} = \mathcal{A}^{(2)} = \{a_1, a_2\}$. Let there be two optimal joint actions: $\langle a_1, a_1 \rangle$ and $\langle a_2, a_2 \rangle$ (e.g., only these joint actions have positive, equal reward). Given *independent* experience samples for each agent, the first agent may learn action a_1 as optimal, whereas the second agent learns a_2, resulting in arbitrarily poor joint action $\langle a_1, a_2 \rangle$. This motivates a need for *concurrent* (synchronized) sampling of experiences across the team in MARL settings. Concurrent experiences induce correlations in local policy updates, so



(a) CERT structure. (b) CERT minibatches.

Figure 1. Concurrent training samples for MARL. Each cube signifies an experience tuple $\langle o_t^{(i)}, a_t^{(i)}, r_t, o_{t+1}^{(i)} \rangle$. Axes e, t, i correspond to episode, timestep, and agent indices, respectively.

that given existence of multiple equilibria, agents tend to converge to the same one. Thus, we introduce Concurrent Experience Replay Trajectories (CERTs), visualized in Fig. 1a. During execution of each learning episode $e \in \mathbb{N}^+$, each agent i collects experience tuple $\langle o_t^{(i)}, a_t^{(i)}, r_t, o_{t+1}^{(i)} \rangle$ at timestep t, where o_t, a_t, and r_t are current observation, action, and reward, and o_{t+1} is the subsequent observation. Fig. 1a visualizes each experience tuple as a cube. Experiences in each episode are stored in a sequence (along time axis t of Fig. 1a), as Dec-HDRQN assumes an underlying RNN architecture that necessitates sequential samples for each training iteration. Importantly, as all agents are aware of timestep t and episode e, they store their experiences concurrently (along agent index axis i of Fig. 1a). Upon episode termination, a new sequence is initiated (a new row along episode axis e of Fig. 1a). No restrictions are imposed on terminal conditions (i.e., varying trajectory lengths are permitted along axis t of Fig. 1a). CERTs are a first-in first-out circular queue along the episode axis e, such that old episodes are eventually discarded.

5.1.3. Training Dec-HDRQNs using CERTs

Each agent i maintains DRQN $Q^{(i)}(o_t^{(i)}, h_{t-1}^{(i)}, a^{(i)}; \theta^{(i)})$, where $o_t^{(i)}$ is the latest local observation, $h_{t-1}^{(i)}$ is the RNN hidden state, $a^{(i)}$ is the action, and $\theta^{(i)}$ are the local DRQN parameters. DRQNs are trained on experience sequences (traces) with *tracelength* τ. Figure 1b visualizes the minibatch sampling procedure for training, with $\tau = 4$. In each training iteration, agents first sample a *concurrent* minibatch of episodes. All agents' sampled traces have the same starting timesteps (i.e., are coincident along agent axis i in Fig. 1b). Guaranteed concurrent sampling merely requires a one-time (offline) consensus of agents' random number generator seeds prior to initiating learning. This ensures our approach is fully decentralized and assumes no explicit communication, even during learning. Fig. 1b shows a minibatch of 3 episodes, e, sampled in red. To train DRQNs, Hausknecht & Stone (2015) suggest randomly sampling a timestep within each episode, and training using τ backward steps. However, this imposes a bias where experiences in each episode's final τ timesteps are used in fewer recurrent updates. Instead, we propose that for each sampled episode e, agents sample a concurrent start timestep t_0 for the trace from interval $\{-\tau+1,\ldots,H_e\}$, where H_e is the timestep of the episode's final experience. For example, the three sampled (red) traces in Fig. 1b start at timesteps $+1$, -1, and $+2$, respectively. This ensures all experiences have equal probability of being used in updates, which we found especially critical for fast training on tasks with only terminal rewards.

Sampled traces sometimes contain elements outside the episode interval (indicated as $\varnothing$ in Fig. 1b). We discard $\varnothing$ experiences and zero-pad the suffix of associated traces (to ensure all traces have equal length τ, enabling seamless use of fixed-length minibatch optimizers in standard deep learning libraries). Suffix (rather than prefix) padding ensures RNN internal states of non-$\varnothing$ samples are unaffected. In training iteration j, agent i uses the sampling procedure to collect a minibatch of traces from CERT memory $\mathcal{M}^{(i)}$,

$$\begin{aligned}\mathcal{B} = \{\langle\langle o^b_{t_0}, a^b_{t_0}, r^b_{t_0}, o^b_{t_0+1}\rangle, \ldots,\\ \langle o^b_{t_0+\tau-1}, a^b_{t_0+\tau-1}, r^b_{t_0+\tau-1}, o^b_{t_0+\tau}\rangle\rangle\}_{b=\{1,\ldots,B\}},\end{aligned} \tag{3}$$

where t_0 is the start timestep for each trace, b is trace index, and B is number of traces (minibatch size).[2] Each trace b is used to calculate a corresponding sequence of target values,

$$\{\langle\langle y^b_{t_0}\rangle, \ldots, \langle y^b_{t_0+\tau-1}\rangle\rangle\}_{b=\{1,\ldots,B\}}, \tag{4}$$

where $y^b_t = r^b_t + \gamma \max_{a'} Q(o^b_{t+1}, h^b_t, a'; \hat{\theta}^{(i)}_j)$. Target network parameters $\hat{\theta}^{(i)}_j$ are updated less frequently, for stable learning (Mnih et al., 2015). Loss over all traces is,

$$L_j(\theta^{(i)}_j) = \mathbb{E}_{(o^b_t, a^b_t, r^b_t, o^b_{t+1})\sim\mathcal{M}^{(i)}}[(\delta^b_t)^2], \tag{5}$$

where $\delta^b_t = y^b_t - Q(o^b_t, h^b_{t-1}, a^b_t; \theta^{(i)}_j)$. Loss contributions of suffix $\varnothing$-padding elements are masked out. Parameters are updated via gradient descent on Eq. (5), with the caveat of hysteretic learning rates $0<\beta<\alpha<1$, where learning rate α is used if $\delta^b_t \geq 0$, and β is used otherwise.

5.2. Phase II: Dec-POMDP MT-MARL

Following task specialization, the second phase involves distillation of each agent's set of DRQNs into a unified DRQN that performs well in all tasks without explicit provision of task ID. Using DRQNs, our approach extends the single-agent, fully-observable MTRL method proposed by Rusu et al. (2015) to Dec-POMDP MT-MARL. Specifically, once Dec-HDRQN specialization is conducted for each task, multi-task learning can be treated as a regression problem over Q-values. During multi-task learning, our approach iteratively conducts data collection and regression.

For data collection, agents use each specialized DRQN (from Phase I) to execute actions in corresponding tasks, resulting in a set of regression CERTs $\{\mathcal{M}_R\}$ (one per task), each containing sequences of *regression experiences* $\langle o^{(i)}_t, Q^{(i)}_t\rangle$, where $Q^{(i)}_t = Q^{(i)}_t(\vec{o}^{(i)}_t; \theta^{(i)})$ is the specialized DRQN's Q-value vector for agent i at timestep t. Supervised learning of Q-values is then conducted. Each agent samples experiences from its local regression CERTs to train a *single* distilled DRQN with parameters $\theta^{(i)}_R$. Given a minibatch of regression experience traces $\mathcal{B}_R = \{\langle\langle o^b_{t_0}, Q^b_{t_0}\rangle, \ldots, \langle o^b_{t_0+\tau-1}, Q^b_{t_0+\tau-1}\rangle\rangle\}_{b=\{1,\ldots,B\}}$, the following tempered Kullback-Leibler (KL) divergence loss is minimized for each agent,

$$\begin{aligned}&L_{KL}(\mathcal{B}_R, \theta^{(i)}_R; T) \\ &= \mathbb{E}_{(o^b_t, Q^b_t)\sim\{\mathcal{M}_{\mathcal{R}}{}^{(i)}\}} \sum_{a=1}^{|\mathcal{A}^{(i)}|} \text{softmax}_a\left(\frac{Q^b_t}{T}\right) \ln \frac{\text{softmax}_a(\frac{Q^b_t}{T})}{\text{softmax}_a(Q^b_{t,R})},\end{aligned} \tag{6}$$

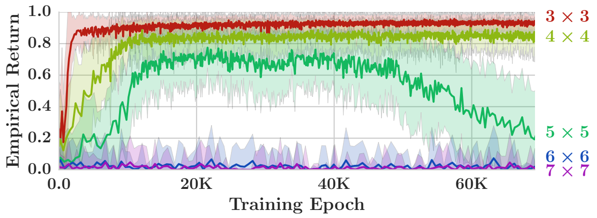
where $Q^b_{t,R} = Q^b_{t,R}(\vec{o_t}^b; \theta^{(i)}_R)$ is the vector of action-values predicted by distilled DRQN given the same input as the specialized DRQN, T is the softmax temperature, and softmax$_a$ refers to the a-th element of the softmax output. The motivation behind loss function (6) is that low temperatures ($0<T<1$) lead to sharpening of specialized DRQN action-values, Q^b_t, ensuring that the distilled DRQN ultimately chooses similar actions as the specialized policy it was trained on. We refer readers to Rusu et al. (2015) for additional analysis of the distillation loss. Note that concurrent sampling is not necessary during the distillation phase, as it is entirely supervised; CERTs are merely used for storage of the regression experiences.

6. Evaluation

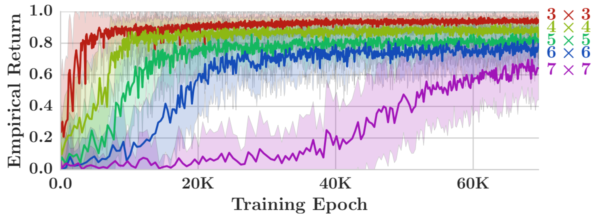
6.1. Task Specialization using Dec-HDRQN

We first evaluate single-task performance of the introduced Dec-HDRQN approach on a series of increasingly challenging domains. Domains are designed to support a large number of task variations, serving as a useful MT-MARL benchmarking tool. All experiments use DRQNs with 2 multi-layer perceptron (MLP) layers, an LSTM layer (Hochreiter & Schmidhuber, 1997) with 64 memory cells, and another 2 MLP layers. MLPs have 32 hidden units each and rectified linear unit nonlinearities are used throughout, with the exception of the final (linear) layer. Experiments use $\gamma = 0.95$ and Adam optimizer (Kingma & Ba, 2014) with base learning rate 0.001. Dec-HDRQNs use hysteretic learning rate $\beta = 0.2$ to 0.4. All results are reported for batches of 50 randomly-initialized episodes.

[2] For notational simplicity, agent superscripts (i) are excluded from local experiences $\langle o^{(i)}, a^{(i)}, r, o'^{(i)}\rangle$ in Eqs. (3) to (6).



(a) Learning via Dec-DRQN.



(b) Learning via Dec-HDRQN (our approach).

Figure 2. Task specialization for MAMT domain with $n = 2$ agents, $P_f = 0.3$. (a) Without hysteresis Dec-DRQN policies destabilize in the 5×5 task and fails to learn in the 6×6 and 7×7 tasks. (b) Dec-HDRQN (our approach) performs well in all tasks.

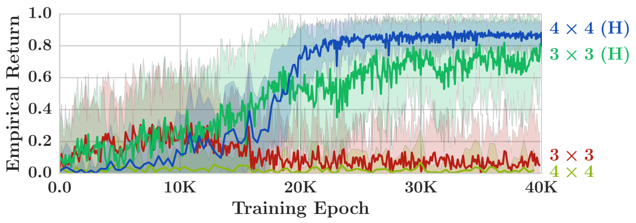


Figure 3. The advantage of hysteresis is even more pronounced for MAMT with $n = 3$ agents. $P_f = 0.6$ for 3×3 task, and $P_f = 0.1$ for 4×4 task. Dec-HDRQN indicated by (H).

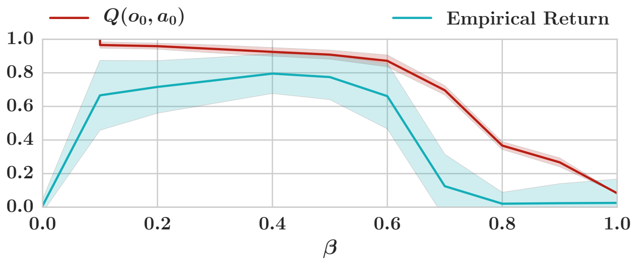


Figure 4. Dec-HDRQN sensitivity to learning rate β (6×6 MAMT domain, $n = 2$ agents, $P_f = 0.25$). Anticipated return $Q(o_0, a_0)$ upper bounds actual return due to hysteretic optimism.

Performance is evaluated on both multi-agent single-target (MAST) and multi-agent multi-target (MAMT) capture domains, variations of the existing meeting-in-a-grid Dec-POMDP benchmark (Amato et al., 2009). Agents $i \in \{1, \ldots, n\}$ in an $m \times m$ toroidal grid receive $+1$ terminal reward only when they *simultaneously* capture moving targets (1 target in MAST, and n targets in MAMT). Each agent always observes its own location, but only sometimes observes targets' locations. Target dynamics are unknown to agents and vary across tasks. Similar to the Pong POMDP domain of Hausknecht & Stone (2015), our domains include observation flickering: in each timestep, observations of targets are sometimes obscured, with probability P_f. In MAMT, each agent is assigned a unique target to capture, yet is unaware of the assignment (which also varies across tasks). Agent/target locations are randomly initialized in each episode. Actions are 'move north', 'south', 'east', 'west', and 'wait', but transitions are noisy (0.1 probability of moving to an unintended adjacent cell).

In the MAST domain, each task is specified by a unique grid size m; in MAMT, each task also has a unique agent-target assignment. The challenge is that agents must learn particular roles (to ensure coordination) and also discern aliased states (to ensure quick capture of targets) using local noisy observations. Tasks end after H timesteps, or upon simultaneous target capture. Cardinality of local policy space for agent i at timestep t is $O(|\mathcal{A}^{(i)}|^{\frac{|\Omega^{(i)}|^{t}-1}{|\Omega^{(i)}|-1}})$ (Oliehoek et al., 2008), where $|A^{(i)}| = 5$, $|\Omega^{(i)}| = m^4$ for MAST, and $|\Omega^{(i)}| = m^{2(n+1)}$ for MAMT. Across all tasks, non-zero reward signals are extremely sparse, appearing in the terminal experience tuple only if targets are *simultaneously* captured. Readers are referred to the supplementary material for domain visualizations.

The failure point of Dec-DRQN is first compared to Dec-HDRQN in the MAST domain with $n = 2$ and $P_f = 0$ (full observability) for increasing task size, starting from 4×4. Despite the domain simplicity, Dec-DRQN fails to match Dec-HDRQN at the 8×8 mark, receiving value 0.05 ± 0.16 in contrast to Dec-HDRQN's 0.76 ± 0.11 (full results reported in supplementary material). Experiments are then scaled up to a 2 agent, 2 target MAMT domain with $P_f = 0.3$. Empirical returns throughout training are shown in Fig. 2. In the MAMT tasks, a well-coordinated policy induces agents to capture targets simultaneously (yielding joint $+1$ reward). If any agent strays from this strategy during learning (e.g., while exploring), teammates receive no reward even while executing optimal local policies, leading them to deviate from learned strategies. Due to lack of robustness against alter-exploration/non-stationarity, the Dec-DRQN becomes unstable in the 5×5 task, and fails to learn altogether in the 6×6 and 7×7 tasks (Fig. 2a). Hysteresis affords Dec-HDRQN policies the stability necessary to consistently achieve agent coordination (Fig. 2b). A *centralized-learning* variation of Dec-DRQN with inter-agent parameter sharing (similar to RIAL-PS in Foerster et al. (2016)) was also tested, but was not found to improve performance (see supplementary material). These re-

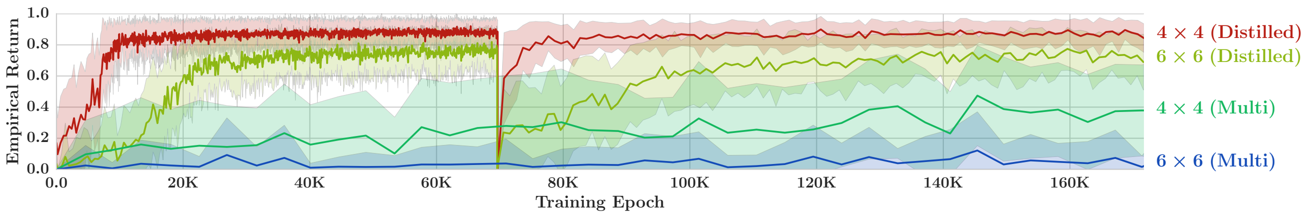


Figure 5. MT-MARL performance of the proposed Dec-HDRQN specialization/distillation approach (labeled as *Distilled*) and simultaneous learning approach (labeled as *Multi*). Multi-task policies for both approaches were trained on all MAMT tasks from 3×3 through 6×6. Performance shown only for 4×4 and 6×6 domains for clarity. Distilled approach shows specialization training (Phase I of approach) until 70K epochs, after which distillation is conducted (Phase II of approach). Letting the simultaneous learning approach run for up to 500K episodes did not lead to significant performance improvement. By contrast, the performance of our approach during the distillation phase (which includes task identification) is almost as good as its performance during the specialization phase.

sults further validate that, despite its simplicity, hysteretic learning significantly improves the stability of MARL in cooperative settings. Experiments are also conducted for the $n = 3$ MAMT domain (Fig. 3). This domain poses significant challenges due to reward sparsity. Even in the 4×4 task, only 0.02% of the joint state space has a non-zero reward signal. Dec-DRQN fails to find a coordinated joint policy, receiving near-zero return after training. Dec-HDRQN successfully coordinates the 3 agents. Note the high variance in empirical return for the 3×3 task is due to flickering probability being increased to $P_f = 0.6$.

Sensitivity of Dec-HDRQN empirical performance to hysteretic learning rate β is shown in Fig. 4, where lower β corresponds to higher optimism; $\beta = 0$ causes monotonic increase of approximated Q-values during learning, whereas $\beta = 1$ corresponds to Dec-DRQN. Due to the optimistic assumption, anticipated returns at the initial timestep, $Q(o_0, a_0)$, overestimate true empirical return. Despite this, $\beta \in [0.1, 0.6]$ consistently enables learning of a well-coordinated policy, with $\beta \in [0.4, 0.5]$ achieving best performance. Readers are referred to the supplementary material for additional sensitivity analysis of convergence trends with varying β and CERT tracelength τ.

6.2. Multi-tasking using Distilled Dec-HDRQN

We now evaluate distillation of specialized Dec-HDRQN policies (as learned in Section 6.1) for MT-MARL. A first approach is to forgo specialization and directly learn a Dec-HDRQN using a pool of experiences from all tasks. This approach, called Multi-DQN by Rusu et al. (2015), is susceptible to convergence issues even in single-agent, fully-observable settings. In Fig. 5, we compare these approaches (where we label ours as 'Distilled', and Multi-HDRQN as 'Multi'). Both approaches were trained to perform multi-tasking on 2-agent MAMT tasks ranging from 3×3 to 6×6, with $P_f = 0.3$. Our distillation approach uses no task-specific MLP layers, unlike Rusu et al. (2015), due to our stronger assumptions on task relatedness and lack of execution-time observability of task identity.

In Fig. 5, our MT-MARL approach first performs Dec-HDRQN specialization training on each task for 70K epochs, and then performs distillation for 100K epochs. A grid search was conducted for temperature hyperparameter in Eq. (6) ($T = 0.01$ was found suitable). Note that performance is plotted only for the 4×4 and 6×6 tasks, simply for plot clarity (see supplementary material for MT-MARL evaluation results on all tasks). Multi-HDRQN exhibits poor performance across all tasks due to the complexity involved in concurrently learning over multiple Dec-POMDPs (with partial observability, transition noise, non-stationarity, varying domain sizes, varying target dynamics, and random initializations). We experimented with larger and smaller network sizes for Multi-HDRQN, with no major difference in performance (we also include training results for 500K Multi-HDRQN iterations in the supplementary). By contrast, our proposed MT-MARL approach achieves near-nominal execution-time performance on all tasks using a single distilled policy for each agent – despite not explicitly being provided the task identity.

7. Contribution

This paper introduced the first formulation and approach for multi-task multi-agent reinforcement learning under partial observability. Our approach combines hysteretic learners, DRQNs, CERTs, and distillation, demonstrably achieving multi-agent coordination using a single joint policy in a set of Dec-POMDP tasks with sparse rewards, despite not being provided task identities during execution. The parametric nature of the capture tasks used for evaluation (e.g., variations in grid size, target assignments and dynamics, sensor failure probabilities) makes them good candidates for ongoing benchmarks of multi-agent multi-task learning. Future work will investigate incorporation of skills (macro-actions) into the framework, extension to domains with heterogeneous agents, and evaluation on more complex domains with much larger numbers of tasks.

Acknowledgements

The authors thank the anonymous reviewers for their insightful feedback and suggestions. This work was supported by Boeing Research & Technology, ONR MURI Grant N000141110688 and BRC Grant N000141712072.

References

Amato, Christopher, Dibangoye, Jilles Steeve, and Zilberstein, Shlomo. Incremental policy generation for finite-horizon DEC-POMDPs. In *ICAPS*, 2009.

Banerjee, Bikramjit, Lyle, Jeremy, Kraemer, Landon, and Yellamraju, Rajesh. Sample bounded distributed reinforcement learning for decentralized POMDPs. In *AAAI*, 2012.

Barbalios, Nikos and Tzionas, Panagiotis. A robust approach for multi-agent natural resource allocation based on stochastic optimization algorithms. *Applied Soft Computing*, 18:12–24, 2014.

Bengio, Yoshua. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pp. 437–478. Springer, 2012.

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

Bowling, Michael and Veloso, Manuela. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250, 2002.

Brunskill, Emma and Li, Lihong. Sample complexity of multi-task reinforcement learning. *arXiv preprint arXiv:1309.6821*, 2013.

Buşoniu, Lucian, Babuška, Robert, and De Schutter, Bart. Multi-agent reinforcement learning: An overview. In *Innovations in multi-agent systems and applications-1*, pp. 183–221. Springer, 2010.

Caruana, Rich. Multitask learning. In *Learning to learn*, pp. 95–133. Springer, 1998.

Claus, Caroline and Boutilier, Craig. The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI*, 1998:746–752, 1998.

Dutech, Alain, Buffet, Olivier, and Charpillet, François. Multi-agent systems by incremental gradient reinforcement learning. In *Proc. of the International Joint Conf. on Artificial Intelligence*, pp. 833–838, 2001.

Fernández, Fernando and Veloso, Manuela. Probabilistic policy reuse in a reinforcement learning agent. In *Proc. of the fifth international joint conf. on Autonomous agents and multiagent sys.*, pp. 720–727. ACM, 2006.

Foerster, Jakob, Nardelli, Nantas, Farquhar, Gregory, Torr, Philip, Kohli, Pushmeet, Whiteson, Shimon, et al. Stabilising experience replay for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1702.08887*, 2017.

Foerster, Jakob N, Assael, Yannis M, de Freitas, Nando, and Whiteson, Shimon. Learning to communicate to solve riddles with deep distributed recurrent Q-networks. *arXiv preprint arXiv:1602.02672*, 2016.

Fulda, Nancy and Ventura, Dan. Predicting and preventing coordination problems in cooperative Q-learning systems. In *IJCAI*, volume 2007, pp. 780–785, 2007.

Gordon, Geoffrey J. Stable function approximation in dynamic programming. In *Proc. of the twelfth international conf. on machine learning*, pp. 261–268, 1995.

Hausknecht, Matthew and Stone, Peter. Deep recurrent Q-learning for partially observable MDPs. *arXiv preprint arXiv:1507.06527*, 2015.

Hinton, Geoffrey, Vinyals, Oriol, and Dean, Jeff. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural comp.*, 9(8):1735–1780, 1997.

Kaelbling, Leslie Pack, Littman, Michael L, and Cassandra, Anthony R. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1): 99–134, 1998.

Kapetanakis, Spiros and Kudenko, Daniel. Reinforcement learning of coordination in cooperative multi-agent systems. *AAAI/IAAI*, 2002:326–331, 2002.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Lauer, Martin and Riedmiller, Martin. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proc. of the Seventeenth International Conf. on Machine Learning*. Citeseer, 2000.

Laurent, Guillaume J, Matignon, Laëtitia, Fort-Piat, Le, et al. The world of independent learners is not markovian. *International Journal of Knowledge-based and Intelligent Engineering Systems*, 15(1):55–64, 2011.

Lin, Long-Ji. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.

Liu, Miao, Amato, Christopher, Liao, Xuejun, Carin, Lawrence, and How, Jonathan P. Stick-breaking policy learning in Dec-POMDPs. In *Proc. of the International Joint Conf. on Artificial Intelligence*, 2015.

Liu, Miao, Amato, Christopher, Anesta, Emily, Griffith, J. Daniel, and How, Jonathan P. Learning for decentralized control of multiagent systems in large partially observable stochastic environments. In *AAAI*, 2016.

Matignon, Laëtitia, Laurent, Guillaume J, and Le Fort-Piat, Nadine. Hysteretic Q-learning: an algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In *IROS*, 2007.

Matignon, Laetitia, Laurent, Guillaume J, and Le Fort-Piat, Nadine. Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems. *The Knowledge Engineering Review*, 27(01): 1–31, 2012.

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Rusu, Andrei A, Veness, Joel, Bellemare, Marc G, Graves, Alex, Riedmiller, Martin, Fidjeland, Andreas K, Ostrovski, Georg, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Oliehoek, Frans A. and Amato, Christopher. *A Concise Introduction to Decentralized POMDPs*. Springer, 2016.

Oliehoek, Frans A, Spaan, Matthijs TJ, Vlassis, Nikos A, et al. Optimal and approximate q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research (JAIR)*, 32:289–353, 2008.

Pan, Sinno Jialin and Yang, Qiang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

Peshkin, Leonid, Kim, Kee-Eung, Meuleau, Nicolas, and Kaelbling, Leslie Pack. Learning to cooperate via policy search. In *Proc. of the Sixteenth conf. on Uncertainty in artificial intelligence*, pp. 489–496. Morgan Kaufmann Publishers Inc., 2000.

Rusu, Andrei A, Colmenarejo, Sergio Gomez, Gulcehre, Caglar, Desjardins, Guillaume, Kirkpatrick, James, Pascanu, Razvan, Mnih, Volodymyr, Kavukcuoglu, Koray, and Hadsell, Raia. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

Sutton, Richard S and Barto, Andrew G. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Tan, Ming. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proc. of the tenth international conf. on machine learning*, pp. 330–337, 1993.

Tanaka, Fumihide and Yamamura, Masayuki. Multitask reinforcement learning on the distribution of mdps. In *Computational Intelligence in Robotics and Automation, 2003. Proceedings. 2003 IEEE International Symposium on*, volume 3, pp. 1108–1113. IEEE, 2003.

Taylor, Adam, Dusparic, Ivana, Galván-López, Edgar, Clarke, Siobhán, and Cahill, Vinny. Transfer learning in multi-agent systems through parallel transfer. In *Workshop on Theoretically Grounded Transfer Learning at the 30th International Conf. on Machine Learning (Poster)*, volume 28, pp. 28. Omnipress, 2013.

Taylor, Matthew E and Stone, Peter. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.

Torrey, Lisa and Shavlik, Jude. Transfer learning. *Handbook of Research on Machine Learning Applications and Trends: Algs., Methods, and Techniques*, 1:242, 2009.

Watkins, Christopher JCH and Dayan, Peter. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Wierstra, Daan, Foerster, Alexander, Peters, Jan, and Schmidhuber, Juergen. Solving deep memory POMDPs with recurrent policy gradients. In *International Conf. on Artificial Neural Networks*, pp. 697–706. Springer, 2007.

Wilson, Aaron, Fern, Alan, Ray, Soumya, and Tadepalli, Prasad. Multi-task reinforcement learning: a hierarchical bayesian approach. In *Proc. of the 24th international conf. on Machine learning*, pp. 1015–1022. ACM, 2007.

Wilson, Aaron, Fern, Alan, Ray, Soumya, and Tadepalli, Prasad. Learning and transferring roles in multi-agent reinforcement. In *Proc. AAAI-08 Workshop on Transfer Learning for Complex Tasks*, 2008.

Wu, Feng, Zilberstein, Shlomo, and Chen, Xiaoping. Rollout sampling policy iteration for decentralized POMDPs. *arXiv preprint arXiv:1203.3528*, 2012.

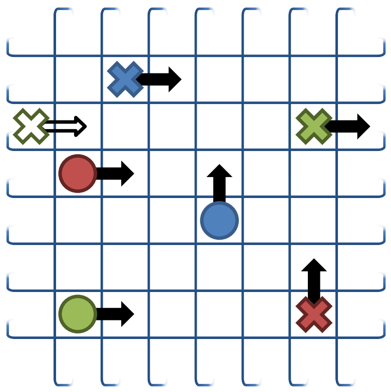
Wu, Feng, Zilberstein, Shlomo, and Jennings, Nicholas R. Monte-carlo expectation maximization for decentralized POMDPs. In *Proc. of the International Joint Conf. on Artificial Intelligence*, pp. 397–403, 2013.

Xu, Yinliang, Zhang, Wei, Liu, Wenxin, and Ferrese, Frank. Multiagent-based reinforcement learning for optimal reactive power dispatch. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1742–1751, 2012.

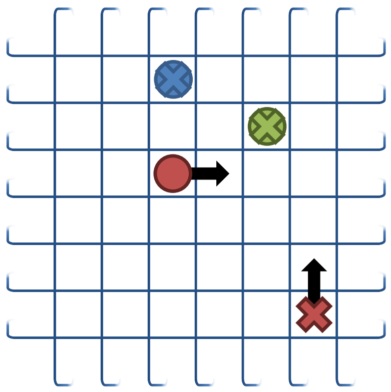
Supplemental: Deep Decentralized Multi-task Multi-agent RL under Partial Observability

The following provides additional empirical results. Some plots are reproduced from the main text to ease comparisons.

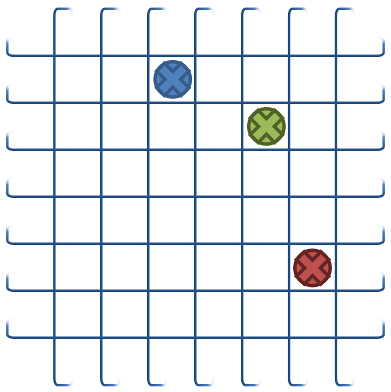
A. Multi-agent Multi-target (MAMT) Domain Overview



(a) Agents must learn inherent toroidal transition dynamics in the domain for fast target capture (e.g., see green target).



(b) In MAMT tasks, no reward is given to the team above, despite two agents successfully capturing their targets.

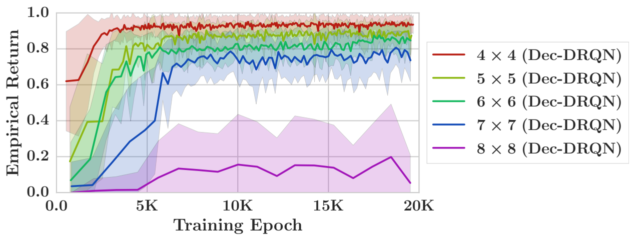


(c) Example successful simultaneous capture scenario, where the team is given $+1$ reward.

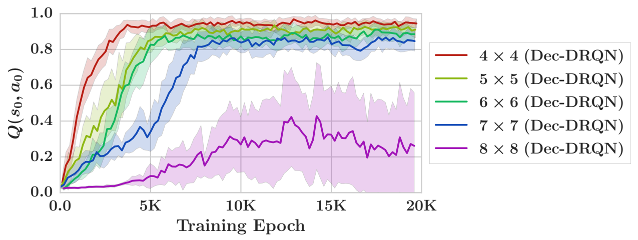
Figure 6. Visualization of MAMT domain. Agents and targets operate on a toroidal $m \times m$ gridworld. Each agent (circle) is assigned a unique target (cross) to capture, but does not observe its assigned target ID. Targets' states are fully occluded at each timestep with probability P_f. Despite the simplicity of gridworld transitions, reward sparsity makes this an especially challenging task. During both learning and execution, the team receives no reward unless all targets are captured *simultaneously* by their corresponding agents.

B. Empirical Results: Learning on Multi-agent Single-Target (MAST) Domain

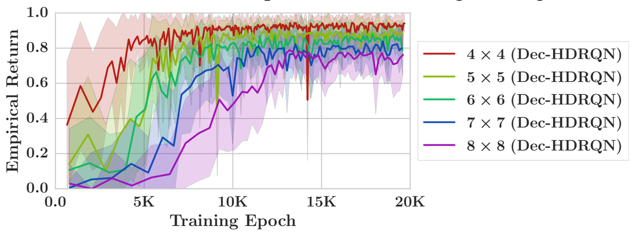
Multi-agent Single-target (MAST) domain results for Dec-DRQN and Dec-HDRQN, with 2 agents and $P_f = 0.0$ (observation flickering disabled). These results mainly illustrate that Dec-DRQN sometimes has *some* empirical success in low-noise domains with small state-space. Note that in the 8×8 task, Dec-HDRQN significantly outperforms Dec-DRQN, which converges to a sub-optimal policy despite domain simplicity.



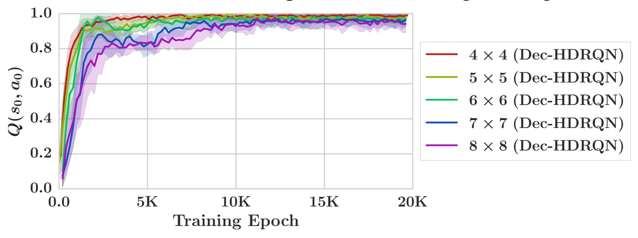
(a) Dec-DRQN empirical returns during training.



(b) Dec-DRQN anticipated values during training.



(c) Dec-HDRQN empirical returns during training.

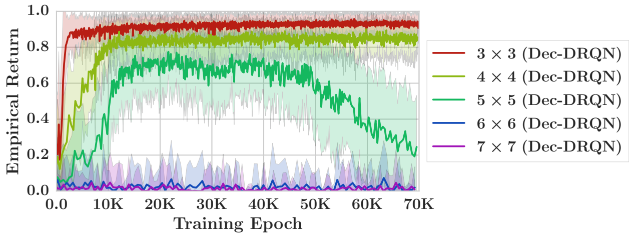


(d) Dec-HDRQN anticipated values during training.

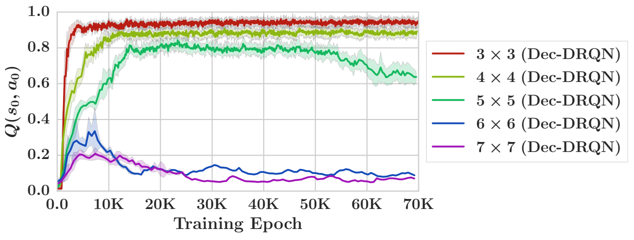
Figure 7. Multi-agent Single-target (MAST) domain results for Dec-DRQN and Dec-HDRQN, with 2 agents and $P_f = 0.0$ (observation flickering disabled). All plots conducted (at each training epoch) for a batch of 50 randomly-initialized episodes. Anticipated value plots (on right) were plotted for the exact starting states and actions undertaken for the episodes used in the plots on the left.

C. Empirical Results: Learning on MAMT Domain

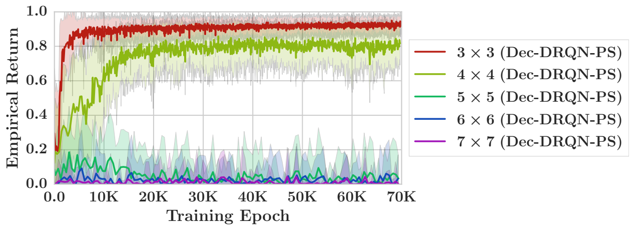
Multi-agent Single-target (MAMT) domain results, with 2 agents and $P_f = 0.3$ (observation flickering disabled). We also evaluated performance of inter-agent parameter sharing (a centralized approach) in Dec-DRQN (which we called Dec-DRQN-PS). Additionally, performance of a Double-DQN was deemed to have negligible impacts (Dec-DDRQN).



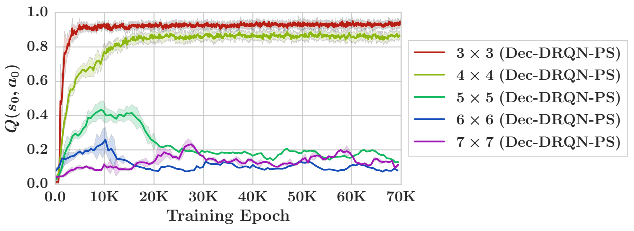
(a) Dec-DRQN empirical returns during training.



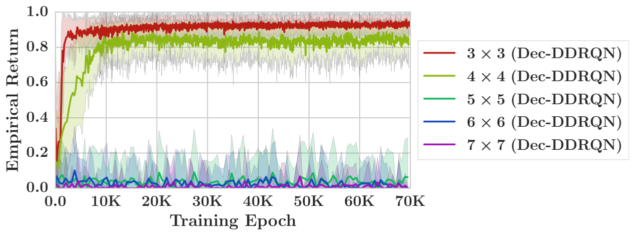
(b) Dec-DRQN anticipated values during training.



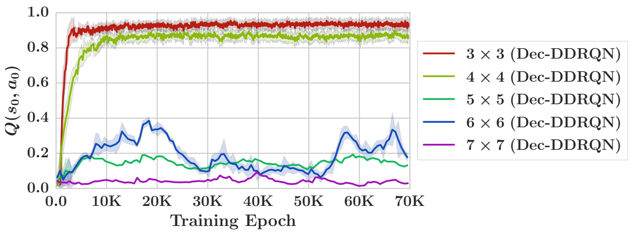
(c) Dec-DRQN-PS (with parameter sharing), empirical returns during training.



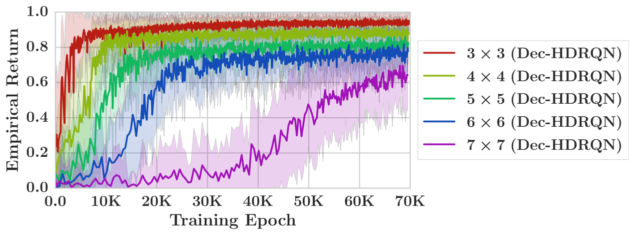
(d) Dec-DRQN-PS (with parameter sharing), anticipated values during training.



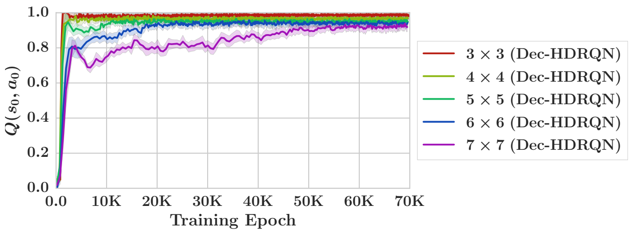
(e) Dec-DDRQN (double DRQN) empirical returns during training.



(f) Dec-DDRQN (double DRQN) anticipated values during training.



(g) Dec-HDRQN (our approach) empirical returns during training.



(h) Dec-HDRQN (our approach) anticipated values during training.

Figure 8. MAMT domain results for Dec-DRQN and Dec-HDRQN, with 2 agents and $P_f = 0.3$. All plots conducted (at each training epoch) for a batch of 50 randomly-initialized episodes. Anticipated value plots (on right) were plotted for the exact starting states and actions undertaken for the episodes used in the plots on the left.

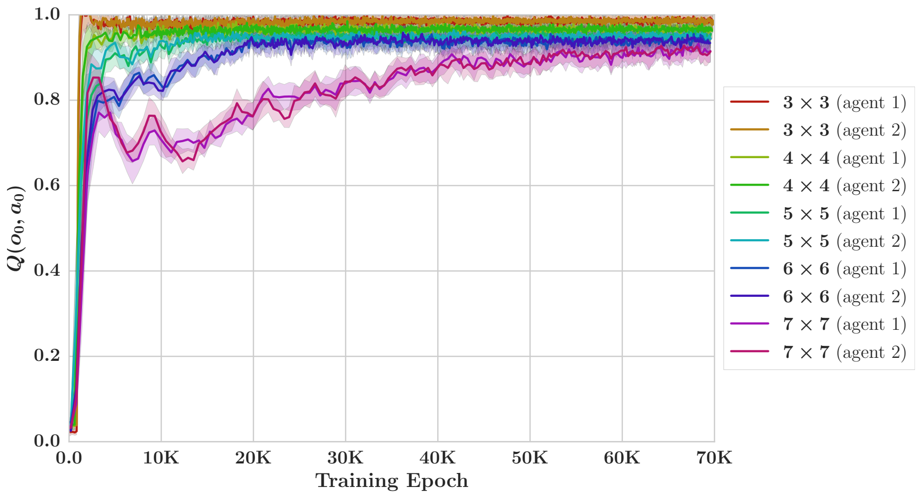
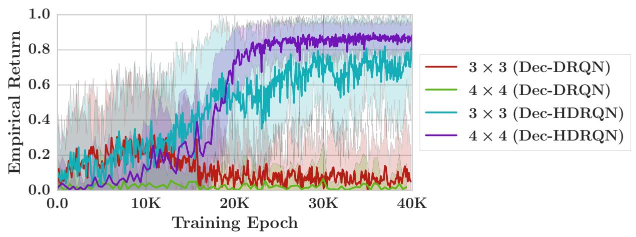
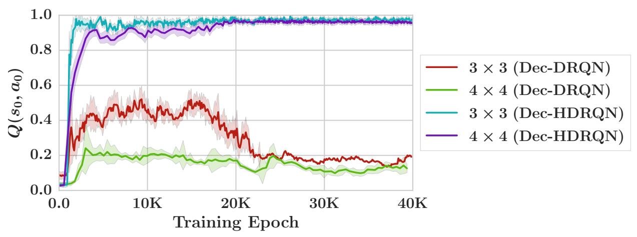


Figure 9. Comparison of agents' anticipated value plots using Dec-HDRQN during training. MAMT domain, with 2 agents and $P_f = 0.3$. All plots conducted (at each training epoch) for a batch of 50 randomly-initialized episodes. For a given task, agents have similar anticipated value convergence trends due to shared reward; differences are primarily caused by random initial states and independently sampled target occlusion events for each agent.



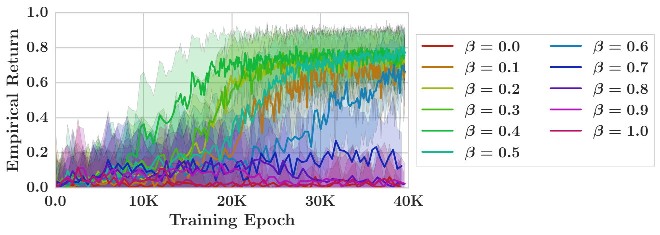
(a) Empirical returns during training. For batch of 50 randomly-initialized games.



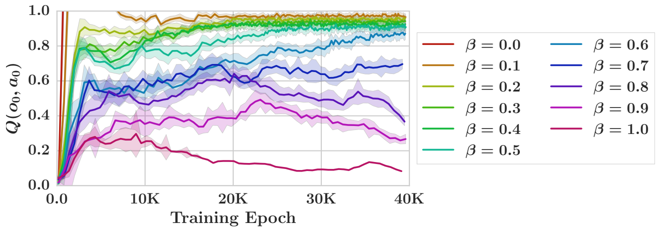
(b) Anticipated values during training. For specific starting states and actions undertaken in the same 50 randomly-initialized games as Fig. 10a.

Figure 10. MAMT domain results for Dec-DRQN and Dec-HDRQN, with $n = 3$ agents. $P_f = 0.6$ for the 3×3 task, and $P_f = 0.1$ for the 4×4 task.

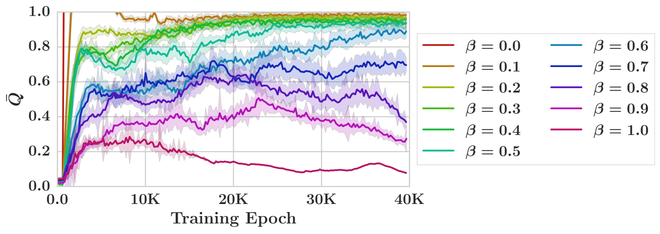
D. Empirical Results: Learning Sensitivity to Dec-HDRQN Negative Learning Rate β



(a) Sensitivity of Dec-HDRQN empirical returns to β during training. For batch of 50 randomly-initialized games.



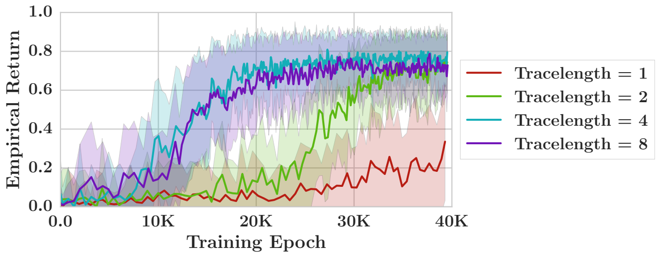
(b) Sensitivity of Dec-HDRQN predicted action-values to β during training. For specific starting states and actions undertaken in the same 50 randomly-initialized games of Fig. 11a.



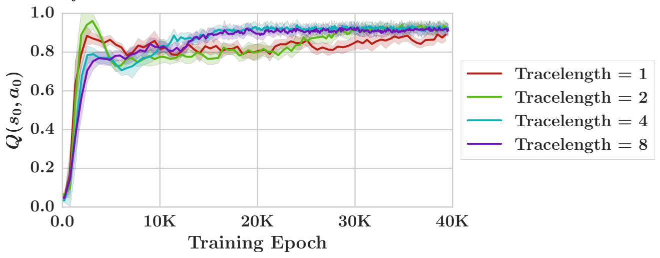
(c) Sensitivity of Dec-HDRQN average Q values to β during training. For random minibatch of 32 experienced observation inputs.

Figure 11. Learning sensitivity to β for 6×6, 2 agent MAMT domain with $P_f = 0.25$. All plots for agent $i = 0$. $\beta = 1$ corresponds to Decentralized Q-learning, $\beta = 0$ corresponds to Distributed Q-learning (not including the distributed policy update step).

E. Empirical Results: Learning Sensitivity to Dec-HDRQN Recurrent Training Tracelength Parameter τ



(a) Dec-HDRQN sensitivity to tracelength τ. 6x6 MAMT domain with $P_f = 0.25$.

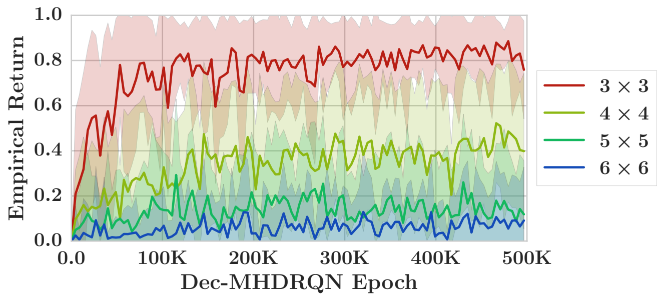


(b) Sensitivity of Dec-HDRQN predicted action-values to recurrent training tracelength parameter. For specific starting states and actions undertaken in the same 50 randomly-initialized games of Fig. 12a.

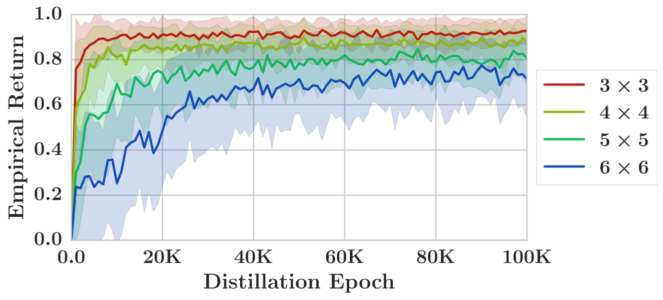
Figure 12. Learning sensitivity to recurrent training tracelength parameter for 6×6, 2 agent MAMT domain with $P_f = 0.25$. All plots for agent $i = 0$. $\beta = 1$ corresponds to Decentralized Q-learning, $\beta = 0$ corresponds to Distributed Q-learning (not including the distributed policy update step).

F. Empirical Results: Multi-tasking Performance Comparison

The below plots show multi-tasking performance of both the distillation and Multi-HDRQN approaches. Both approaches were trained on the 3×3 through 6×6 MAMT tasks. Multi-DRQN failed to achieve specialized-level performance on all tasks, despite 500K training epochs. By contrast, the proposed MT-MARL distillation approach achieves nominal performance after 100K epochs.



(a) MT-MARL via Multi-HDQRN.



(b) MT-MARL via specialized and distilled Dec-HDRQN.

Figure 13. Multi-task performance on MAMT domain, $n = 2$ agents and $P_f = 0.3$.