Successor Features Based Multi-Agent RL for Event-Based Decentralized MDPs

Tarun Gupta[1] [2], Akshat Kumar[1]†, Praveen Paruchuri[2]†
[1]School of Information Systems, Singapore Management University, Singapore
[2]Machine Learning Lab, Kohli Center on Intelligent Systems, IIIT Hyderabad, India
{tarung, akshatkumar}@smu.edu.sg, praveen.p@iiit.ac.in

Abstract

Decentralized MDPs (Dec-MDPs) provide a rigorous framework for collaborative multi-agent sequential decision-making under uncertainty. However, their computational complexity limits the practical impact. To address this, we focus on a class of Dec-MDPs consisting of independent collaborating agents that are tied together through a global reward function that depends upon their entire histories of states and actions to accomplish joint tasks. To overcome scalability barrier, our main contributions are: (a) We propose a new actor-critic based Reinforcement Learning (RL) approach for event-based Dec-MDPs using *successor features (SF)* which is a value function representation that decouples the dynamics of the environment from the rewards; (b) We then present Dec-ESR (Decentralized Event based Successor Representation) which generalizes learning for event-based Dec-MDPs using SF within an end-to-end deep RL framework; (c) We also show that Dec-ESR allows useful transfer of information on related but different tasks, hence bootstraps the learning for faster convergence on new tasks; (d) For validation purposes, we test our approach on a large multi-agent coverage problem which models schedule coordination of agents in a real urban subway network and achieves better quality solutions than previous best approaches.

1 Introduction

Sequential multi-agent decision-making allows multiple agents operating in an uncertain and partially observable environment to take a coordinated decision towards a long-term goal (Durfee and Zilberstein 2013). Decentralized partially observable MDPs (Dec-POMDPs) have emerged as a rich framework for multi-agent planning (Bernstein et al. 2002), and are applicable to several domains such as multi-agent robotics (Amato et al. 2016) and urban system optimization (Varakantham, Adulyasak, and Jaillet 2014). However, scalability remains challenging due to NEXP-Hard complexity even for two agents (Bernstein et al. 2002). To address such complexity issues, various models are explored where agent interactions are structured using various conditional and contextual independencies such as transition and observation independence among agents (Becker et al. 2004; Kumar, Zilberstein, and Toussaint 2011), event-driven interactions (Becker, Zilberstein, and Lesser 2004), collective interactions among agents (Varakantham, Adulyasak, and Jaillet 2014; Nguyen, Kumar, and Lau 2017a; 2017b) and weakly coupled agents (Spaan and Melo 2008; Witwicki and Durfee 2010).

We focus on multi-agent decision-making problems where agents are primarily coupled with complex joint-rewards which may depend upon the entire state-action trajectory of multiple agents (Becker et al. 2004). Such event-driven models better capture the asynchronous nature of agents when executing their policies in the real-world, and can encode the notion of agents accomplishing high-level tasks which is crucial for modeling real-world problems such as multi-agent coverage (Yehoshua and Agmon 2016). Several previous approaches have been developed for such transition independent decentralized MDP (TIDec-MDP) model. Dibangoye et al. (2013) develop an occupancy measure based approach over joint state-space of agents to compute agent policies. However, they do not explicitly address event-based rewards. Petrik and Zilberstein (2011) develop a bilinear programming based approach that can model event-based rewards, however, it is limited to two agents only. Gupta, Kumar, and Paruchuri (2018) develop a nonlinear math programming formulation for TIDec-MDPs with event-based rewards which work well for problems with small to medium state-space. For larger problems, they developed a policy gradient-based multi-agent reinforcement learning (MARL) approach for this model. Policy gradient-based approaches have recently become popular for general Dec-POMDPs (Dibangoye and Buffet 2018; Foerster et al. 2017), however, such formulations do not explicitly take into account event-based rewards which depend on entire state-action trajectories of multiple agents. In addition, a key unaddressed problem is that of doing transfer learning when some aspects of the model change.

In our work, we develop a new actor-critic (Konda and Tsitsiklis 2003) based MARL method for event-based Dec-MDPs using *successor features (SF)* which is a value function representation that decouples the dynamics of the environment from the rewards. Previous SF-based approaches are mainly for single agent problems (Dayan 1993; Kulkarni et al. 2016; Barreto et al. 2017); our work extends them to event-based multi-agent models; (b) We then present

†Equal advising.

Dec-ESR, which generalizes learning for event-based Dec-MDPs using SF within an end-to-end deep RL framework. Previous approach of Gupta, Kumar, and Paruchuri (2018) is not an end-to-end training approach and requires Monte-Carlo estimates of probability of events, which suffers from high variance; (c) Thanks to decoupling of rewards from environment dynamics using SFs, we also show that the proposed method allows useful transfer of information on related but different tasks. Therefore, it bootstraps the learning and makes convergence much faster on new tasks. We test our approach on a large multi-agent coverage problem, show its effectiveness against previous approaches, and the ability to do transfer learning.

2 Model Definition

We define an n-agent transition independent Dec-MDP (TIDec-MDP) using the tuple $\langle S, A, P, R\rangle$ (Becker et al. 2004):

- Factored state space defined as $S = \times_{i=1}^{n} S^i$, where S^i is the state space for agent i.
- Factored action space $A = \times_{i=1}^{n} A^i$, where A^i is the action space for agent i.
- Given the joint state $s = \langle s^i \rangle_{i=1}^{n}$ and joint-action $a = \langle a^i \rangle_{i=1}^{n}$, the transition to next state $\overline{s}$ has probability $P(\overline{s}|s,a) = \times_{i=1}^{n} P^i(\overline{s}^i|s^i,a^i)$, where P^i is agent i's local state transition function. This factorization of the transition function results in the *transition independence* property of the model.
- Local observability: Each agent fully observes its own local state s_t^i at each time step t. Agent i does not observe the local state of any other agent during execution time.
- Local rewards: Each agent has its own local reward function $r^i(s^i,a^i)$, and the global reward is additively defined as $r(s,a) = \sum_{i=1}^{n} r^i(s^i,a^i)$.

The above model defines a set of n-*independent* MDPs as agent's transition, observation, and the reward functions are all independent. We next describe how *joint-rewards* are defined that depend on the actions of multiple agents. Such event-based joint-rewards are the key to defining a rich class of non-linear interactions among agents that can model several practical scenarios.

Event-Driven Interactions: We now introduce further structure into the global reward function using detailed treatment presented in Becker et al. (2004). To define it, we need to introduce the notion of an occurrence of an event during the execution of a local policy.

Definition 1. *A* ***history*** Φ^i *denotes a valid local state-action execution sequence* $[s_1^i, a_1^i, s_2^i, a_2^i, \ldots]$ *for an agent i, starting with the local initial state for that agent (subscripts denote time). A* ***primitive event*** *for an agent i,* $e = (\hat{s}^i, a^i, \hat{s}^{i\prime})$*, is a tuple that includes the agent's local state, an action, and an outcome state. An* ***event*** $E = \{e_1, \ldots, e_h\}$ *is a set of primitive events.*

Definition 2. *A primitive event* $e = (\hat{s}^i, a^i, \hat{s}^{i\prime})$ *occurs in history* Φ^i*, denoted* $\Phi^i \models e$*, iff the triplet* $(\hat{s}^i, a^i, \hat{s}^{i\prime})$ *appears as a subsequence of* Φ^i*. An event E occurs in the history* Φ^i*, denoted* $\Phi^i \models E$ *iff* $\exists e \in E : \Phi^i \models e$

Events are used to signify the accomplishment of some task by an agent. Although a single local state-action transition may be sufficient to signify the completion of a task, we generally need a set of primitive events to account for the uncertainty in the domain as well as the fact that tasks could be accomplished in many different ways.

Definition 3. *A primitive event e is* ***proper*** *if it can occur at most once in each possible history of a given MDP. An event E is* ***proper*** *if it consists of mutually exclusive proper primitive events w.r.t. a given MDP. That is:*

$$\forall \Phi^i \neg\exists j \neq k : (e_j \in E \wedge e_k \in E \wedge \Phi^i \models e_j \wedge \Phi^i \models e_k)$$

Becker et al. show how non-proper events can be cast as proper events using techniques such as making time part of the state or including additional bits in the state to memorize the occurrence of some primitive events. We limit the discussion in this paper to *proper events* because they are sufficient to express the desired behavior and because they simplify the discussion.

Joint Reward: The joint reward structure can be viewed as a list of multiple constraints between the agents that describe how interactions between their local policies affect the global value of the system. Let Φ^1 through Φ^n denote histories for all the agents. A constraint k exists among a subset of agents G_k ($|G_k| \geq 2$). It is defined as a tuple $\langle\langle E_k^j\rangle_{j\in G_k}, c_k\rangle$. Semantically, the constraint k specifies that if *at least* one agent involved in G_k satisfies its part of the constraint, then the global reward c_k is given. Formally, constraint $\langle\langle E_k^j\rangle_{j\in G_k}, c_k\rangle$ specifies that the reward c_k is added to the global value iff $\Phi^j \models E_k^j$ for *at least* one agent $j \in G_k$. Let ρ be the set of all constraints; the same logic is followed for each constraint $k \in \rho$.

Policy and Joint Value function: A joint-policy $\boldsymbol{\pi} = (\pi^1, \ldots, \pi^n)$ is a set of individual policies for each agent. For TIDec-MDPs, the optimal local policy depends on agent's local observed state (Goldman and Zilberstein 2004). We represent agent i's stochastic policy as mapping from local state to a distribution over actions or $\pi^i(a^i|s^i)$. We have fixed-horizon histories, say H. Given local policies π^i, the probability $P(e;\pi^i)$ of a proper, primitive event $e = (\hat{s}^i, a^i, \hat{s}^{i\prime})$ occurring during any execution of π^i is:

$$P(e;\pi^i) = \sum_{t=1}^{H} P(s_t^i = \hat{s}^i;\pi^i)\pi^i(a^i|\hat{s}^i)P^i(\hat{s}^{i\prime}|\hat{s}^i,a^i) \quad (1)$$

As all primitive events in a proper event E are mutually exclusive, we have $P(E;\pi^i) = \sum_{e\in E} P(e;\pi^i)$. Given the starting state s_1^i for an agent i, ρ as the set of constraints, the global value function is defined as:

$$GV(\boldsymbol{s}_1;\boldsymbol{\pi}) = \sum_{i=1}^{n} V^i(s_1^i;\pi^i) + JV(\rho;\boldsymbol{\pi}) \quad (2)$$

where V^i is the value function of agent i's local MDP and the joint value function $JV(\rho;\boldsymbol{\pi})$ is defined as:

$$JV(\rho;\boldsymbol{\pi}) = \sum_{k\in\rho} c_k\Big[1 - \prod_{j\in G_k}\Big(1 - P(E^j_k;\pi^j)\Big)\Big] \quad (3)$$

The joint value function uses the fact that the probability of at least one event happening is one minus the probability that none of the events happen. Our goal is to compute the joint-policy $\boldsymbol{\pi}$ that optimizes the global value function (2).

Expressiveness: As per the above constraint semantics, global reward c_k is given if *at least one* event in a constraint occur. Similarly, the global reward can be given at the occurrence of *all events*, *at most x events*, *exactly x events*. The global and joint value function for *all event* semantics is shown in the longer version of the paper.

Brief Domain Definition: We experiment on a multi-agent coverage domain (Yehoshua and Agmon 2016; Galceran and Carreras 2013) under uncertainty and partial observability. The multi-agent coverage problem involves multiple agents inspecting locations on multiple lines within a mass rapid transit (MRT) network. Agents get local rewards for successfully inspecting locations. Shared locations correspond to *interchange* stations where multiple lines meet. Thus, shared locations can be inspected by multiple agents. The *joint reward* is modeled using *at least one event* semantics where at least one agent must successfully inspect a shared location once every hour (or at some pre-specified interval) for the joint reward to be added to the global value function. Thus, agents are also incentivized to coordinate with each other to avoid multiple agents inspecting the same shared location within a fixed time interval (say an hour).

Recent Work: Gupta, Kumar, and Paruchuri show how to calculate and backpropagate gradients for event-based TIDec-MDPs. Given the start state $\boldsymbol{s}_1$ at time step 1 for all the agents; each agent i's policy parameterized using θ^i (which represents NN parameters), the goal here is to compute the gradient of global-value function (2):

$$\begin{aligned}&\nabla_{\theta^i} GV(\boldsymbol{s}_1;\boldsymbol{\pi})\\ &=\nabla_{\theta^i} V^i(s^i_1;\pi^i) + \sum_{k\in\rho^i} c_k \nabla_{\theta^i} P(E^i_k) \prod_{j\in G_k\setminus\{i\}}\Big(1 - P(E^j_k)\Big)\end{aligned} \quad (4)$$

where ρ^i denotes the set of joint-rewards in which agent i participates. The gradient of local MDP value function $\nabla_{\theta^i} V^i(s^i_1;\pi^i)$ can be computed using REINFORCE with baseline method as explained in Sutton et al. (1999). The gradient of proper event can be computed as:

$$\nabla_{\theta^i} P(E;\pi^i) = \frac{1}{|\xi|}\sum_{e\in E}\sum_{\Phi^i\in\xi:\Phi^i\models e}\Big[\sum_{t'=1}^{t(e,\Phi^i)} \nabla_{\theta^i}\log\pi^i(a_{t'}|s_{t'})\Big]$$

ξ is the set of complete state-action samples from $P^{\pi}(s^i_{1:H+1}, a^i_{1:H})$; and $t(e,\Phi^i)$ denotes the time at which event e occurs in Φ^i. Given sample set ξ for each agent, we can also empirically compute the probability estimates of primitive events e, and use them to compute empirical estimate of events $P(E^j_k)$, which can be used in (4).

2.1 Markov Modeling of Event Probabilities

Becker et al. and Gupta, Kumar, and Paruchuri use $P(E;\pi^i)$ in the joint value function (Eq. 3) to evaluate the probability of events, which does not follow the *Markov* property. $P(E;\pi^i)$ is estimated using Monte-Carlo estimation, however like all Monte-Carlo methods it tends to be slow to learn (high variance) and inconvenient to implement online as explained in (Sutton et al. 1999). Moreover, the value of $P(E;\pi^i)$ does **not** provide any information about experiences of previous state-action trajectories while doing RL. That is, it cannot be used as a critic for bootstrapping (updating a state from the estimated values of subsequent states), but can only be used as a baseline for the state being updated.

Our Contribution: Our first contribution to tackle above problems is that we compute $P^{\pi^i}_t(E\,|\,s^i_t, a^i_t)$, that is the probability of occurence of event E, given action a^i_t is taken in state s^i_t by agent i at time t following policy π^i. $P^{\pi^i}_t(E\,|\,s^i_t, a^i_t)$ follows the *Markov* property as shown below, and can be easily trained with dynamic programming using TD (Temporal Difference) learning. Therefore, it serves as a *true bootstrapping critic* and allows actor critic to be more sample efficient via TD updates at every step.

Definition 4. *Consider a proper event E which consists of multiple proper primitive events. Given an experience tuple $(s^i_t, a^i_t, s^i_{t+1})$, we define an indicator function for agent i, $\varphi^i_E(s^i_t, a^i_t, s^i_{t+1})$ which is 1 iff $(s^i_t, a^i_t, s^i_{t+1})$ is one of the primitive events of E and otherwise 0. Formally,*

$$\varphi^i_E(s^i_t, a^i_t, s^i_{t+1}) = \left\{\begin{array}{cc} 1 & (s^i_t, a^i_t, s^i_{t+1}) \in E \\ 0 & \textit{otherwise}\end{array}\right\} \quad (5)$$

We next define how $P^{\pi^i}_t(E|s^i_t, a^i_t)$ can be written in a recursive manner:

$$\begin{aligned}&P^{\pi^i}_t(E\,|\,s^i_t, a^i_t) = \mathbb{E}_{s^i_{t+1}|s^i_t,a^i_t}\Big[\varphi^i_E(s^i_t, a^i_t, s^i_{t+1}) +\\ &\tilde{\varphi}^i_E(s^i_t,a^i_t,s^i_{t+1})\sum_{a^i_{t+1}}\pi^i(a^i_{t+1}|s^i_{t+1})P^{\pi^i}_{t+1}(E|s^i_{t+1},a^i_{t+1})\Big]\end{aligned} \quad (6)$$

where $\tilde{\varphi}^i_E(s^i_t,a^i_t,s^i_{t+1}) = 1 - \varphi^i_E(s^i_t, a^i_t, s^i_{t+1})$. Notice that since our primitive events are proper, only one of the primitive events $e \in E$ can happen in any given history Φ^i. Therefore, the above equation considers if $(s^i_t, a^i_t, s^i_{t+1})$ occurs in history Φ^i, that is $\varphi^i_E(s^i_t, a^i_t, s^i_{t+1}) = 1$, then all future expectation will be *zero* using $\tilde{\varphi}^i_E(s^i_t, a^i_t, s^i_{t+1})$. Given the starting state s^i_1 for an agent i, ρ as the set of constraints, the global value function $GV(\boldsymbol{s}_1;\boldsymbol{\pi})$ is now defined as:

$$= \mathbb{E}_{\boldsymbol{s}_1,\boldsymbol{a}_1}\Big[\sum_{i=1}^{n} Q^{\pi^i}_1(s^i_1, a^i_1) + JV(\rho;\boldsymbol{s}_1,\boldsymbol{a}_1,\boldsymbol{\pi})\Big] \quad (7)$$

where $Q^{\pi^i}_1(s^i_1, a^i_1)$ is the Q value function of agent i when action a^i_1 is taken in state s^i_1 under policy π^i. The joint value $JV(\rho;\boldsymbol{s}_1,\boldsymbol{a}_1,\boldsymbol{\pi})$ function now becomes:

$$= \mathbb{E}_{\boldsymbol{s}_1,\boldsymbol{a}_1}\left[\sum_{k\in\rho} c_k\Big[1 - \prod_{j\in G_k}\Big(1 - P^{\pi^j}_1(E^j_k|s^j_1,a^j_1)\Big)\Big]\right]$$

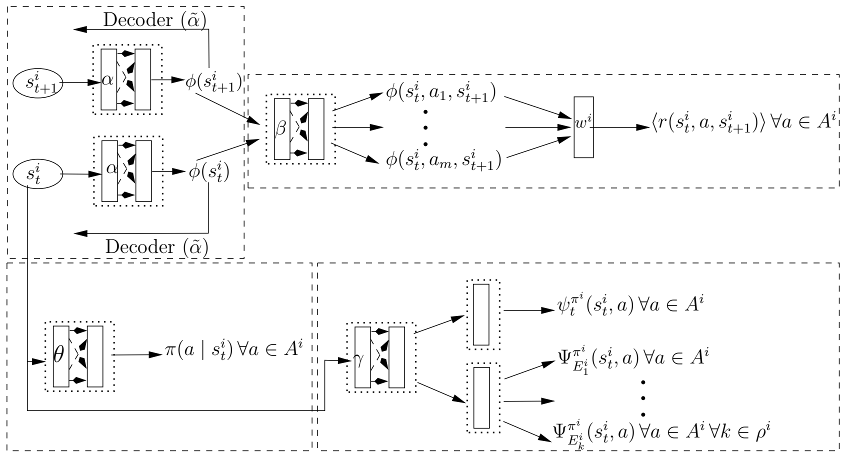

Decoder ($\tilde{\alpha}$)
s^i_{t+1}
α
$\phi(s^i_{t+1})$
β
$\phi(s^i_t, a_1, s^i_{t+1})$
$\phi(s^i_t, a_m, s^i_{t+1})$
w^i
$\langle r(s^i_t, a, s^i_{t+1})\rangle \; \forall a \in A^i$
s^i_t
α
$\phi(s^i_t)$
Decoder ($\tilde{\alpha}$)
θ
$\pi(a \mid s^i_t) \; \forall a \in A^i$
γ
$\psi^{\pi^i}_t(s^i_t, a) \; \forall a \in A^i$
$\Psi^{\pi^i}_{E^i_1}(s^i_t, a) \; \forall a \in A^i$
$\Psi^{\pi^i}_{E^i_k}(s^i_t, a) \; \forall a \in A^i \; \forall k \in \rho^i$

Figure 1: **Dec-ESR Model Architecture for agent i:** (1) State Feature Encoder f_α, (2) Feature Encoder f_β, (3) Feature Decoder $f_{\tilde{\alpha}}$ which produces the input reconstruction s^i_t from state features, (4) A linear regressor to predict instantaneous rewards $r(s^i_t, a, s^i_{t+1}) \; \forall a \in A^i$, (5) A successor features network f_γ and, (6) A policy neural network f_θ. The same architecture is used for all agents.

Our Actor-Critic Approach: We follow an actor-critic (AC) based policy gradient approach (Konda and Tsitsiklis 2003). The joint-policy $\boldsymbol{\pi}$ is parameterized using $\boldsymbol{\theta}$. The parameters are adjusted to maximize the objective $GV(\boldsymbol{s}_1; \boldsymbol{\pi})$ by taking steps in the direction of $\nabla_{\boldsymbol{\theta}} GV(\boldsymbol{s}_1; \boldsymbol{\pi})$. In the AC approach, the policy π is termed as an *actor*. We can estimate $Q^{\pi^i}_t$ (the action-value function) and $P^{\pi^i}_t$ (event probabilities) for each agent i using empirical returns, but it has high variance. To remedy this, AC methods often use a function approximator for $Q^{\pi^i}_t$ and $P^{\pi^i}_t$, which we denote as the *local* and *event* critic respectively. The critic can be learned from empirical returns using TD learning. $P^{\pi^i}_t(E|s^i_t, a^i_t)$ serves as an excellent *event-based critic* for bootstrapping the policy gradient for joint value function, as empirically noted.

3 Successor Features

We will next define successor features (SFs) and show how we can use an AC method based on both *local* and *event* based critic using SFs to get better global rewards, and also to transfer knowledge across related but different tasks.

3.1 Successor Features for MDPs

Successor features for MDPs have been introduced multiple times in (Dayan 1993; Barreto et al. 2017; Kulkarni et al. 2016) and therefore, we will present them briefly. Assume that an agent i experiences the following tuple $(s^i_t, a^i_t, s^i_{t+1}, r^i_{t+1})$. Let us assume that the function $\phi^i_t(s^i_t, a^i_t, s^i_{t+1}) \in \Re^d$ gives d-dimensional features such that $\phi^i_t(s^i_t, a^i_t, s^i_{t+1}) \cdot w^i = r^i_{t+1}$, where $w^i \in \Re^d$. For a given policy of the agent π^i, the $Q^{\pi^i}_t$ function is given as:

$$
\begin{aligned}
&Q^{\pi^i}_t(s^i_t, a^i_t) \\
&= \mathbb{E}_{s^i_{t+1:H+1}, a^i_{t+1:H}} \left[r^i_{t+1} + r^i_{t+2} + \ldots + r^i_{H+1} \mid s^i_t, a^i_t \right] \\
&= \mathbb{E}_{s^i_{t+1:H+1}, a^i_{t+1:H}} \left[\phi^i_t + \phi^i_{t+1} + \ldots \mid s^i_t, a^i_t \right]^\intercal w^i \\
&= \psi^{\pi^i}_t(s^i_t, a^i_t)^\intercal w^i
\end{aligned}
$$

where $\psi^{\pi^i}_t(s^i_t, a^i_t)$ are the successor features of (s^i_t, a^i_t) under policy π^i. Successor features decomposes the value function into two components — a reward predictor and a successor map. The successor map $\psi^{\pi^i}_t(s^i_t, a^i_t)$ represents the expected future state occupancy from an action taken in any given state and the reward predictor maps transitions to scalar rewards. The value function of a state can be computed as the inner product between the successor map and the reward weights. This decomposition is at the core of transfer learning in our Dec-ESR approach where $\psi^{\pi^i}_t(s^i_t, a^i_t)$ will serve as the *local* critic function.

3.2 Successor Features for Events

We next show how to compute SFs for events using the definition in (6). The probability that the event E happens given the current state s^i_t and action a^i_t, that is $P^{\pi^i}_t(E|s^i_t, a^i_t)$:

$$
\begin{aligned}
&= \mathbb{E}\big[\varphi^i_E(s^i_t, a^i_t, s^i_{t+1}) + \varphi^i_E(s^i_{t+1}, a^i_{t+1}, s^i_{t+2}) + .. \mid s^i_t, a^i_t \big] \\
&= \Psi^{\pi^i}_E(s^i_t, a^i_t)
\end{aligned}
$$

where the expectation $\mathbb{E}$ is over samples $s^i_{t+1:H+1}, a^i_{t+1:H}$ and the indicator function $\varphi^i_E(s^i_t, a^i_t, s^i_{t+1})$ gives 1 iff $(s_t, a_t, s_{t+1}) \in E$, otherwise zero. We call $\Psi^{\pi^i}_E(s^i_t, a^i_t)$ the successor features of event E, given (s^i_t, a^i_t) under policy π^i. In our Dec-ESR approach, $\Psi^{\pi^i}_E(s^i_t, a^i_t)$ will serve as the *event* based critic function. Notice that since our primitive events are proper, the summation $\big[\varphi^i_E(s^i_t, a^i_t, s^i_{t+1}) + \varphi^i_E(s^i_{t+1}, a^i_{t+1}, s^i_{t+2}) + \ldots \mid s^i_t, a^i_t \big]$ would be either *zero* or *one*. Therefore, the successor feature $\Psi^{\pi^i}_E(s^i_t, a^i_t)$ should give us the probability of event E happening given the current state and action. We will drop the superscript denoting agent i to simplify the notation in the next two subsections.

3.3 Dec-ESR

In this section, we will present Dec-ESR : Decentralized Event based Successor Representation, which generalizes learning for event-based Dec-MDPs using SF within an end-to-end deep reinforcement learning framework. First, we will describe the neural network (NN) architecture for each agent i shown in Figure 1. For large state spaces, representing and learning the SR can become intractable; therefore, we use a state feature encoder which is a non-linear function approximation parameterized by α to represent and learn a D-dimensional feature vector $\phi(s_t)$ and $\phi(s_{t+1})$ to further learn a d-dimensional feature vector $\phi_t(s_t, a, s_{t+1})\ \forall a \in A$ which is the output of a deep NN parameterized by β.

$$f_\alpha : s_t \rightarrow \phi(s_t) \in \Re^D$$
$$f_\beta : \langle \phi(s_t), \phi(s_{t+1}) \rangle \rightarrow \phi_t(s_t, a, s_{t+1}) \in \Re^d\ \forall a \in A$$

For a feature vector $\phi_t(s_t, a_t, s_{t+1})$, we define a feature-based SR as the expected future occupancy of the features and denote it by $\psi_t(s_t, a_t)$. We approximate our *local critic* i.e. $\psi_t(s_t, a)\ \forall a \in A$ by NN parameterized by γ. We also approximate our *event* based *critic* for all events of the agent i.e. $\Psi_E(s_t, a_t)$ by same NN parameterized by γ.

$$f_\gamma : s_t \rightarrow \langle \psi_t(s_t, a), \Psi_E(s_t, a) \rangle \in \Re^d\ \forall a \in A\ \forall E$$

Finally, we also approximate the immediate reward $R(s_t, a_t, s_{t+1})$ as a linear function of the feature vector $\phi_t(s_t, a_t, s_{t+1})$ as $R(s_t, a_t, s_{t+1}) \approx \phi_t(s_t, a_t, s_{t+1}).w$. We have another neural network which uses non-linear function approximation to learn the policy parameterized by θ.

$$f_\theta : s_t \rightarrow \pi(s_t, a)\ \forall a \in A \text{ s.t.} \sum_{a \in A} \pi(s_t, a) = 1$$

The SF for the optimal policy in the non-linear function approximation case can then be obtained from the following Bellman equations:

$$\psi_t(s_t, a_t) = \mathbb{E}_{s_{t+1}|s_t,a_t}\Big[\phi_t(s_t, a_t, s_{t+1}) + \sum_{a_{t+1}} \pi(a_{t+1}|s_{t+1})\psi_{t+1}(s_{t+1}, a_{t+1})\Big] \quad (8)$$

$$\Psi_E(s_t, a_t) = \mathbb{E}_{s_{t+1}|s_t,a_t}\Big[\varphi_E(s_t, a_t, s_{t+1}) + \tilde{\varphi}_E(s_t,a_t,s_{t+1}) \sum_{a_{t+1}} \pi(a_{t+1}|s_{t+1})\Psi_E(s_{t+1}, a_{t+1})\Big] \quad (9)$$

where the terminal cases for time horizon H are defined as $\Psi_E(s_H, a_H) = \varphi_E(s_H, a_H, s_{H+1})$ and $\psi_H(s_H, a_H) = \phi_H(s_H, a_H, s_{H+1})$.

3.4 Learning

We use centralized learning to learn decentralized policies for all agents. The centralized training of decentralized policies is a standard paradigm for multi-agent planning (Oliehoek, Spaan, and Vlassis 2008; Kraemer and Banerjee 2016; Foerster et al. 2017). The parameters $(\alpha, \tilde{\alpha}, \beta, \gamma, w, \theta)$ can be learned online through stochastic gradient descent. The loss function for α and $\tilde{\alpha}$ is given by:

$$L_1(\alpha, \tilde{\alpha}) = (\phi(s_t) - s_t)^2$$

For learning w, the weights for the reward approximation function, we use the following squared loss function:

$$L_2(w, \alpha, \beta) = (r(s_t, a_t, s_{t+1}) - \phi_t(s_t, a_t, s_{t+1}).w)^2$$

An ideal $\phi_t(s_t, a_t, s_{t+1})$ should be: (1) A good discriminator for the states; this condition is handled by using a decoder which produces the input reconstruction s_t by minimizing equation $L_1(\alpha, \tilde{\alpha})$ and (2) A good predictor for reward signal $r(s_t, a_t, s_{t+1})$; this condition is handled by minimizing equation $L_2(w, \alpha, \beta)$. For training parameter γ, we define the following loss functions derived from (8) and (9).

$$L_3(\gamma, \beta, \theta) = \mathbb{E}_{s_{t+1}|s_t,a_t}\Big[\phi_t(s_t, a_t, s_{t+1}) + \sum_{a_{t+1}} \pi(a_{t+1}|s_{t+1})\psi^{pr}_{t+1}(s_{t+1}, a_{t+1}) - \psi_t(s_t, a_t)\Big]^2$$

$$L_4(\gamma,\beta,\theta) = \mathbb{E}_{s_{t+1}|s_t,a_t}\Big[\varphi_E(s_t,a_t,s_{t+1}) + \tilde{\varphi}_E(s_t,a_t,s_{t+1}) \times \sum_{a_{t+1}} \pi(a_{t+1}|s_{t+1})\Psi^{pr}_E(s_{t+1}, a_{t+1}) - \Psi_E(s_t, a_t)\Big]^2$$

The usage of $\psi^{pr}_{t+1}(s_{t+1}, a_{t+1})$ and $\Psi^{pr}_E(s_{t+1}, a_{t+1})$ denotes previously cached values, which are set to current values periodically. This is essential for stable Q-learning with function approximations (Mnih et al. 2015). Finally, let $L_5(\theta)$ denotes the loss for the policy network. We will derive this loss function in the next section. The composite loss function is the sum of the five loss functions given above:

$$L(\alpha, \tilde{\alpha}, \beta, \gamma, \theta, w) = \sum_{p=1}^{5} L_p(.) \quad (10)$$

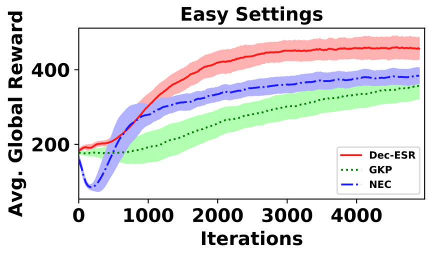
For optimizing the composite loss function in equation 10, with respect to the parameters $(w, \alpha, \tilde{\alpha}, \beta, \gamma, \theta)$, we iteratively update $(w, \alpha, \tilde{\alpha}, \beta)$, γ and θ. That is, we learn a feature representation by minimizing $L_1(\alpha, \tilde{\alpha}) + L_2(w, \alpha, \beta)$; then given $(w, \alpha, \tilde{\alpha}, \beta)$, we find the optimal γ by minimizing $L_3(\gamma, \beta, \theta) + L_4(\gamma, \beta, \theta)$. Since the SFs $\psi_t(s_t, a_t)$ and $\Psi_E(s_t, a_t)$ depend on $\phi_t(s_t, a_t, s_{t+1})$ and $\varphi_E(s_t, a_t, s_{t+1})$ respectively, learning the former while refining the latter can clearly lead to undesirable solutions. Therefore iteration is important to ensure that the successor branch does not back-propagate gradients to affect α and β. Finally given $(w, \alpha, \tilde{\alpha}, \beta, \gamma)$ for all agents, we find the optimal θ by optimizing $L_5(\theta)$. Algorithm 1 in the longer version of the paper highlights the learning algorithm in greater detail.

4 Policy Gradient

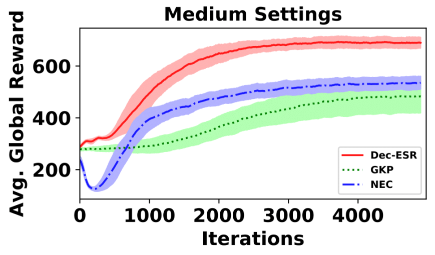
In this section, we show how to backpropagate gradients to optimize the policy loss function and in turn optimize the global value function defined in (7).

$$L_5(\theta) = \nabla_\theta GV(\boldsymbol{s}_1; \boldsymbol{\pi})$$
$$= \nabla_\theta\ \mathbb{E}_{\boldsymbol{s}_1,\boldsymbol{a}_1|\boldsymbol{\pi}}\Big[\sum_{i=1}^{n} Q^{\pi^i}_1(s^i_1, a^i_1) + JV(\rho; \boldsymbol{s}_1, \boldsymbol{a}_1, \boldsymbol{\pi})\Big]$$

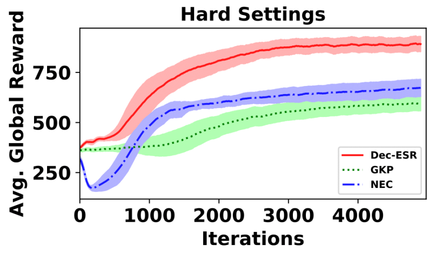
The policy gradient for local Q-value function $\mathbb{E}_{s^i_1,a^i_1|\pi^i}[Q^{\pi^i}_1(s^i_1, a^i_1)]$ with respect to policy parameters of agent i has been computed in Asadi et al. (2017) and



(a) $\langle$GainA, GainB$\rangle$ = $\langle 28\%, 18.9\% \rangle$



(b) $\langle$GainA, GainB$\rangle$ = $\langle 42.8\%, 28.7\% \rangle$



(c) $\langle$GainA, GainB$\rangle$ = $\langle 49.8\%, 32.5\% \rangle$

Figure 2: Solution quality comparisons between GKP, Dec-ESR and No Event Critic (NEC) approach. GainA and GainB are % quality improvement by Dec-ESR over GKP and NEC respectively upon convergence. Shaded regions are one standard deviation over 5 runs.

we will discuss it briefly. The complete derivation can be seen in the longer version of the paper.

$$\nabla_{\theta^i}\mathbb{E}_{s_1^i,a_1^i|\pi^i}\Big[Q_1^{\pi^i}(s_1^i,a_1^i)\Big]$$
$$=\sum_{t=1}^{H}\mathbb{E}_{s_t^i}\Big[\sum_{a\in A^i}\Big(\psi_t^{\pi^i}(s_t^i,a).w^i\Big)\nabla_{\theta^i}\pi^i(a|s_t^i)\Big] \qquad (11)$$

Equation 11 computes the gradient by summing over all actions, rather than just the sampled actions. This helps in reducing variance while performing gradient updates. We now show how to backpropagate gradients through the joint value function with respect to policy parameters of agent i, i.e. θ^i.

Theorem 1. *The policy gradient for joint-value function* $\mathbb{E}_{\boldsymbol{s}_1,\boldsymbol{a}_1|\boldsymbol{\pi}}\Big[JV(\rho;\boldsymbol{s}_1,\boldsymbol{a}_1,\boldsymbol{\pi})\Big]$ *with respect to policy parameters of agent i is given by:*

$$\nabla_{\theta^i}\mathbb{E}_{\boldsymbol{s}_1,\boldsymbol{a}_1|\boldsymbol{\pi}}\Big[JV(\rho;\boldsymbol{s}_1,\boldsymbol{a}_1,\boldsymbol{\pi})\Big]$$
$$=\sum_{k\in\rho^i}c_k\Big[g_k^{\pi^i}(s_1^i,a_1^i)+h_k^{\pi^i}(s_1^i,a_1^i)\Big]q_k^{\boldsymbol{\pi}^{-i}}(\boldsymbol{s}_1^{-i},\boldsymbol{a}_1^{-i}) \qquad (12)$$

where ρ^i is the set of constraints in which agent i participates. The functions g, h, and q are defined as follows:

$$g_k^{\pi^i}(s_1^i,a_1^i)=\mathbb{E}_{s_1^i,a_1^i}\Big[\nabla_{\theta^i}\log\pi^i(a_1^i|s_1^i)\Psi_{E_k^i}^{\pi^i}(s_1^i,a_1^i)\Big]$$
$$q_k^{\boldsymbol{\pi}^{-i}}(\boldsymbol{s}_1^{-i},\boldsymbol{a}_1^{-i})=\mathbb{E}_{\substack{s_1^j,a_1^j\\ j\in G_k\setminus\{i\}}}\Big[\prod_{j\in G_k\setminus\{i\}}\Big(1-\Psi_{E_k^j}^{\pi^j}(s_1^j,a_1^j)\Big)\Big]$$
$$h_k^{\pi^i}(s_1^i,a_1^i)=\nabla_{\theta^i}\Psi_{E_k^i}^{\pi^i}(s_1^i,a_1^i) \qquad (13)$$

The proof is in the longer version of the paper. We will next focus on the gradient $\nabla_{\theta^i}\Psi_{E_k^i}^{\pi^i}(s_1^i,a_1^i)$ from Equation 13:

$$\nabla_{\theta^i}\Psi_{E_k^i}^{\pi^i}(s_1^i,a_1^i)=\sum_{T=2}^{H}\mathbb{E}\Bigg[\Bigg(\prod_{r=1}^{T-1}\tilde{\varphi}_{E_k^i}^{i}(s_r^i,a_r^i,s_{r+1}^i)\Bigg)\times$$
$$\Psi_{E_k^i}^{\pi^i}(s_T^i,a_T^i)\nabla_{\theta^i}\log\pi^i(a_T^i|s_T^i)\Bigg] \qquad (14)$$

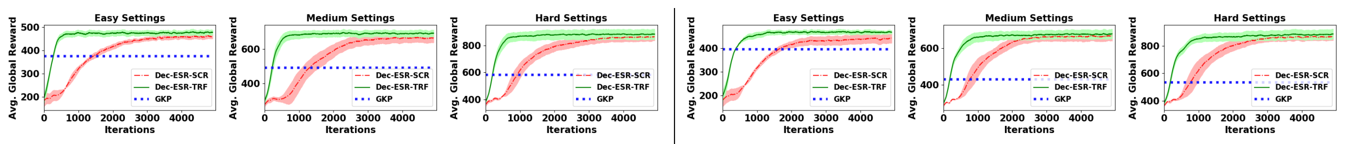
where the expectation $\mathbb{E}$ is over the samples $s_{2:T}^i,a_{2:T}^i|s_1^i,a_1^i$. A detailed proof is available in paper's longer version. We have therefore shown how to compute the gradient of joint value function using samples.

5 Transfer Learning

Taylor and Stone (2009) defines *transfer learning* as improving learning performance in a related, but different, *target task* based on experience gained in learning to perform the *source task*. In our case, we focus on *target tasks* such that the dynamics of the environment does not change, however different rewards (joint or local rewards) may change. In the multi-agent coverage domain, this is motivated by the fact that the rewards for inspection of a station may change on different occasions (e.g., sports stadium station will need to be inspected with higher priority during sport events than on non-event days). Despite the change in rewards, the transition function of agents may not change as the MRT network is fixed.

Next, we discuss what and how to transfer. We examine the sample efficiency of adapting a trained SR model on multiple novel tasks where the reward signal changes. Using SFs, the local Q function is decomposed into two components: a reward predictor w and a successor map $\psi(s_t,a_t)$. The successor map acts as a local critic for MDP and represents the expected future occupancy. The event-based critic $\Psi_E(s_t,a_t)$ represents the probability of event E when action a_t is taken in state s_t. Since our target tasks differ from the source task in terms of the reward signal, we can transfer the **learned** critic functions and therefore, we can quickly re-optimize the new policy with *better* gradient updates in (11) and (13) based on more informed critic signal, than starting learning from scratch. For transferring this knowledge, we start the learning for new policy with learned parameters $(\alpha,\tilde{\alpha},\beta,\gamma,w)$.

Taylor and Stone defined many metrics to measure the benefits of transfer as follows: (1) *Asymptotic Performance*: The final average global reward accumulated by the new joint policy is improved via transfer against learning from scratch; (2) *Total Reward*: The total average global reward accumulated by the joint policy (i.e., the area under the learning curve) may be improved if it uses learned parameters from old policy, compared to learning without transfer; (3) *Time to Threshold*: The learning time needed by the new joint policy to achieve a pre-specified performance level may be reduced via knowledge transfer. In our case, we use the approach proposed by Gupta, Kumar, and Paruchuri as a baseline performance level.



(a) Only Joint-Reward Changes

(b) Both Joint and Local Reward Changes

Figure 3: **Transfer Learning**: Convergence and Quality comparisons between using transfer (Dec-ESR-TRF) against learning from scratch (Dec-ESR-SCR) in $\mathrm{Dec\text{-}ESR}$ approach. GKP is used as a baseline algorithm. Shaded regions are one standard deviation over 5 runs.

6 Experiments

For testing the scalability of our $\mathrm{Dec\text{-}ESR}$ approach, we experimented with the multi-agent coverage problem introduced by Gupta, Kumar, and Paruchuri (2018). We refer to their approach as GKP. Domain details and other experimental settings are provided in the longer version of the paper.

Levels of difficulty: The state space of multi-agent coverage problem is exponential in the number of locations and therefore, we evaluate all the models with three levels of task difficulty, i.e. $\langle \mathrm{Easy}, \mathrm{Medium}, \mathrm{Hard}\rangle$. The categories $\langle \mathrm{Easy}, \mathrm{Medium}, \mathrm{Hard}\rangle$ include $\langle 3, 4, 5\rangle$ MRT lines, $\langle 50, 90, 140\rangle$ stations, and $\langle 5, 10, 15\rangle$ stations being shared as an *interchange* station by $\langle 3, 4, 5\rangle$ lines respectively. Increasing number of shared locations is the key to increasing the complexity of the multi-agent interactions via their joint-actions of visiting these locations. We tested 6 instances in each category with rewards sampled from same distribution for each category. Each line has a single agent able to move among locations on the line.

Figure Description: The x axis of each subfigure in Figure 2 and Figure 3 shows the number of iterations used for training. The y axis of each figure shows solution quality for the $\langle \mathrm{Easy}, \mathrm{Medium}, \mathrm{Hard}\rangle$ categories in terms of average global reward accumulated by all the agents. The results are averaged over all 6 instances of each category. The results for each individual instance in each category can be seen in the longer version of the paper.

Comparison against previous approach: We tested our critic based DSR approach against the deep RL based GKP approach. Fig. 2 shows that our actor-critic based approach produces much higher solution quality than GKP and the GainA i.e. the % improvement by $\mathrm{Dec\text{-}ESR}$ over GKP computed after 5000 iterations is 28% for Easy settings, 42.8% for Medium settings and close to 50% for Hard settings.

Ablations: Since we use both *local* and *event-based critic* functions, we perform ablation experiments to validate the importance of *event-based critic* in our DSR approach. For this purpose, we ran GKP using actor-critic for local MDPs of agents (using Q function) but without any event-based critic. We denote this approach by *No Event Critic* (NEC) as shown in Fig. 2. The figure shows that even though NEC produces slightly higher solution quality than GKP, the critic based DSR approach still gives $\langle 18.9\%, 28.7\%, 32.5\%\rangle$ gain over NEC for $\langle \mathrm{Easy}, \mathrm{Medium}, \mathrm{Hard}\rangle$ categories respectively. This highlights the importance of an event-based true bootstrapping critic in our approach.

Transfer Learning: In this section we use experiments to assess whether the proposed approach can indeed promote transfer on large scale domains. For this purpose, we perform two sets of experiments: (1) *Change in Joint Reward (Figure 3a)*: In this experiment, we changed the joint rewards $c_k\, \forall k \in \rho$ for all shared locations on the MRT map for all the instances; (2) *Change in both Joint and Local Reward (Figure 3b)*: In this experiment, we changed both the joint reward for all shared locations as well as the local rewards $r(s, a, s')$ for all locations on the MRT map for all agents for all the instances. For both sets of experiments, we trained the policy for changed reward signal from scratch without any transfer of information and also trained using transfer of learned parameters $(\alpha, \tilde{\alpha}, \beta, \gamma, w)$. As a baseline, we used GKP approach to evaluate *Time to Threshold* metric as discussed before.

Figure 3 shows the results for all the metrics discussed in Section 5: (1) *Asymptotic Performance*: The final average global reward accumulated by the new joint Dec-ESR-TRF policy when learned using transfer was around 5% higher than learning from scratch; (2) *Time to Threshold*: The time taken for the Dec-ESR-TRF policy (policy learned with transfer) to converge (number of iterations after which the given policy stabilizes i.e. less than epsilon change in average global reward in successive 100 iterations) is lesser than 700 iterations when just the joint reward is changed and around 1000 iterations when both local and joint rewards are changed against a convergence time of 3000 iterations for Dec-ESR-SCR policy when learning from scratch. Also, the time taken for the $\mathrm{Dec\text{-}ESR}$-TRF policy to reach the GKP threshold is faster when trained using transfer against learning from scratch. This highlights the importance of using successor features in our $\mathrm{Dec\text{-}ESR}$ approach as the new policy for changed reward signal can be quickly reoptimized with significantly fewer observations than learning the policy from scratch.

Synthetic Map: We evaluate the scalability of our $\mathrm{Dec\text{-}ESR}$ approach by introducing more agents. To address this, we created a synthetic MRT map having $\langle 10, 20, 30\rangle$ lines with $\langle 10, 20, 30\rangle$ agents in $\langle \mathrm{Easy}, \mathrm{Medium}, \mathrm{Hard}\rangle$ category. The number of shared locations are set as $\langle 5, 10, 15\rangle$ respectively. Each shared locations can be visited by any agent in the synthetic map which increases the complexity of the multi-agent interactions significantly. Every other experimental setting is same as in our current experiments. The results in Table 1 show that our end-to-end approach can scale to a large state-space of individual agents as well as to a large number of agents. The results are averaged over 3 instances in each category.

	Easy	Medium	Hard
Gain A	37.95%	40.93%	52%
Gain B	22.86%	29.56%	28.77%

Table 1: Solution quality on large synthetic MRT map. The $\langle \text{Gain A}, \text{Gain B} \rangle$ refers to % improvement in solution quality of Dec-ESR approach over GKP and No Event Critic (NEC) approaches respectively.

7 Conclusion

We developed a new actor-critic method for event-based Dec-MDPs based on successor features (SFs). The approach used an event-based critic to bootstrap the learning and produces higher solution quality than the previous best approach. Thanks to SFs which decouple environment dynamics from rewards, we are able to transfer knowledge from source to target tasks where only the reward signal in target task changes. This approach achieves significantly faster convergence on target tasks than learning from scratch.

Acknowledgements

We thank anonymous reviewers for their helpful feedback. This research was supported by the Singapore Ministry of Education (MOE) Academic Research Fund (AcRF) Tier 1.

References

Amato, C.; Konidaris, G.; Anders, A.; Cruz, G.; How, J. P.; and Kaelbling, L. P. 2016. Policy search for multi-robot coordination under uncertainty. *International Journal of Robotics Research* 35(14):1760–1778.

Asadi, K.; Allen, C.; Roderick, M.; Mohamed, A.-r.; Konidaris, G.; and Littman, M. 2017. Mean actor critic. *arXiv preprint arXiv:1709.00503*.

Barreto, A.; Dabney, W.; Munos, R.; Hunt, J. J.; Schaul, T.; van Hasselt, H. P.; and Silver, D. 2017. Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems*, 4055–4065.

Becker, R.; Zilberstein, S.; Lesser, V.; and Goldman, C. V. 2004. Solving transition independent decentralized Markov decision processes. *Journal of Artificial Intelligence Research* 22:423–455.

Becker, R.; Zilberstein, S.; and Lesser, V. 2004. Decentralized Markov decision processes with event-driven interactions. In *Proceedings of the 3rd International Conference on Autonomous Agents and Multiagent Systems*, 302–309.

Bernstein, D. S.; Givan, R.; Immerman, N.; and Zilberstein, S. 2002. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research* 27:819–840.

Dayan, P. 1993. Improving generalization for temporal difference learning: The successor representation. *Neural Computation* 5(4):613–624.

Dibangoye, J. S., and Buffet, O. 2018. Learning to act in decentralized partially observable MDPs. In *International Conference on Machine Learning*, 1241–1250.

Dibangoye, J. S.; Amato, C.; Doniec, A.; and Charpillet, F. 2013. Producing efficient error-bounded solutions for transition independent decentralized MDPs. In *International conference on Autonomous Agents and Multi-Agent Systems*, 539–546.

Durfee, E., and Zilberstein, S. 2013. Multiagent planning, control, and execution. In Weiss, G., ed., *Multiagent Systems*. Cambridge, MA, USA: MIT Press. chapter 11, 485–546.

Foerster, J.; Farquhar, G.; Afouras, T.; Nardelli, N.; and Whiteson, S. 2017. Counterfactual multi-agent policy gradients. In *Arxiv*.

Galceran, E., and Carreras, M. 2013. A survey on coverage path planning for robotics. *Robotics and Autonomous Systems* 61(12):1258–1276.

Goldman, C. V., and Zilberstein, S. 2004. Decentralized control of cooperative systems: Categorization and complexity analysis. *J. Artif. Intell. Res.* 22:143–174.

Gupta, T.; Kumar, A.; and Paruchuri, P. 2018. Planning and learning for decentralized mdps with event driven rewards. In *AAAI Conference on Artificial Intelligence*, 6186–6194.

Konda, V. R., and Tsitsiklis, J. N. 2003. On actor-critic algorithms. *SIAM Journal on Control and Optimization* 42(4):1143–1166.

Kraemer, L., and Banerjee, B. 2016. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing* 190:82–94.

Kulkarni, T. D.; Saeedi, A.; Gautam, S.; and Gershman, S. J. 2016. Deep successor reinforcement learning. *arXiv preprint arXiv:1606.02396*.

Kumar, A.; Zilberstein, S.; and Toussaint, M. 2011. Scalable multiagent planning using probabilistic inference. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, 2140–2146.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529.

Nguyen, D. T.; Kumar, A.; and Lau, H. C. 2017a. Collective multiagent sequential decision making under uncertainty. In *AAAI Conference on Artificial Intelligence*, 3036–3043.

Nguyen, D. T.; Kumar, A.; and Lau, H. C. 2017b. Policy gradient with value function approximation for collective multiagent planning. In *Neural Information Processing Systems*.

Oliehoek, F. A.; Spaan, M. T.; and Vlassis, N. 2008. Optimal and approximate q-value functions for decentralized pomdps. *Journal of Artificial Intelligence Research* 32:289–353.

Petrik, M., and Zilberstein, S. 2011. Robust approximate bilinear programming for value function approximation. *Journal of Machine Learning Research* 12:3027–3063.

Spaan, M. T. J., and Melo, F. S. 2008. Interaction-driven Markov games for decentralized multiagent planning under uncertainty. In *International Conference on Autonomous Agents and Multi Agent Systems*, 525–532.

Sutton, R. S.; McAllester, D.; Singh, S.; and Mansour, Y. 1999. Policy gradient methods for reinforcement learning with function approximation. In *International Conference on Neural Information Processing Systems*, 1057–1063.

Taylor, M. E., and Stone, P. 2009. Transfer learning for reinforcement learning domains: A survey. *JMLR* 10:1633–1685.

Varakantham, P.; Adulyasak, Y.; and Jaillet, P. 2014. Decentralized stochastic planning with anonymity in interactions. In *AAAI Conference on Artificial Intelligence*, 2505–2512.

Witwicki, S. J., and Durfee, E. H. 2010. Influence-based policy abstraction for weakly-coupled Dec-POMDPs. In *International Conference on Automated Planning and Scheduling*, 185–192.

Yehoshua, R., and Agmon, N. 2016. Multi-robot adversarial coverage. In *European Conference on Artificial Intelligence*, 1493–1501.