The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Planning and Learning for Decentralized MDPs with Event Driven Rewards

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

Abstract

Decentralized (PO)MDPs provide a rigorous framework for sequential multiagent decision making under uncertainty. However, their high computational complexity limits the practical impact. To address scalability and real-world impact, we focus on settings where a large number of agents primarily interact through *complex* joint-rewards that depend on their entire histories of states and actions. Such history-based rewards encapsulate the notion of events or tasks such that the team reward is given only when the joint-task is completed. Algorithmically, we contribute — 1) A nonlinear programming (NLP) formulation for such event-based planning model; 2) A probabilistic inference based approach that scales much better than NLP solvers for a large number of agents; 3) A policy gradient based multiagent reinforcement learning approach that scales well even for exponential state-spaces. Our inference and RL-based advances enable us to solve a large real-world multiagent coverage problem modeling schedule coordination of agents in a real urban subway network where other approaches fail to scale.

1 Introduction

Decentralized MDPs and POMDPs (Dec-(PO)MDPs) provide a rigorous framework for collaborative multiagent sequential decision making under uncertainty and partial observability (Bernstein et al. 2002). They model settings where agents act based on different partial observations about the environment and about each other to maximize a global objective. Applications of Dec-POMDPs include coordinating planetary rovers (Becker et al. 2004), multi-robot coordination (Amato et al. 2015) and throughput optimization in a wireless network (Pajarinen, Hottinen, and Peltonen 2014). Solving Dec-POMDPs is computationally challenging with NEXP-Hard complexity (Bernstein et al. 2002).

For scalability and practical application, several variants of Dec-POMDPs have been developed including state transition and observation independence (Becker et al. 2004; Nair et al. 2005; Kumar, Zilberstein, and Toussaint 2011; Dibangoye et al. 2013), weak coupling among agents (Witwicki and Durfee 2010) and collective interactions (Sonu, Chen, and Doshi 2015; Robbel, Oliehoek, and Kochenderfer 2016; Nguyen, Kumar, and Lau 2017a; 2017b). Our focus is on agent interactions with complex *event-based* rewards which depend on entire state-action histories of multiple agents. We use the transition independent Dec-MDP model (TIDec-MDP) (Becker et al. 2004) where agents have their independent local MDPs. Agents are coupled through event-based global rewards dependent on their execution histories. Intuitively, events capture if agents accomplished some high-level task. Becker et al. show such event-driven rewards are highly expressive being able to model temporal relations among agents (e.g., agent A's activity *facilitates* or *hinders* agent B's activity rewards). Joint-rewards are given when *at least* x or *at most* x or *exactly* x out of k events occur. Such expressiveness is useful to model several practical problems such as multiagent coverage (Yehoshua and Agmon 2016). Several approaches have been developed to solve TIDec-MDPs and related models. (Dibangoye et al. 2013) use occupancy measures over the joint state-space of agents to compute agent policies. Their focus is on standard reward setting, and do not address event-based rewards. Bilinear programming based approach of (Petrik and Zilberstein 2011) is limited to two agents. (Scharpff et al. 2016) solve transition independent *multiagent* MDPs where an agent's policy can depend on the global state while the global state is unobservable in TIDec-MDPs. Thus, our work addresses this gap by developing scalable approaches for large multiagent settings ($\gg 2$ agents) and event-based rewards.

Our contributions are: 1) we present a nonlinear programming (NLP) formulation for TIDec-MDPs with event-based rewards. The NLP formulation is more scalable than the approach for 2-agent TIDec-MDPs in (Becker et al. 2004). Becker et al.'s approach models some types of joint-rewards, e.g., *at least* or *at most* x events, by adding additional bits to states leading to exponential state-space increase. Our NLP formulation does not require any modifications to the state-space and is highly compact; 2) we present an inference-based approach that translates the NLP to that of inference in a graphical model by extending the planning-as-inference strategy (Toussaint, Harmeling, and Storkey 2006; Kumar, Zilberstein, and Toussaint 2015). A key difference is that we *directly* solve the underlying nonconvex, nonlinear program without requiring expensive forward-backward message-passing. Unlike the NLP, our inference approach always involves solving one small *convex* program per agent

[†]Equal advising.

(regardless of the total number of agents/events) thus highly scalable, and has good anytime performance; 3) we develop a policy gradient based multiagent RL approach that exploits advances in deep RL to represent and optimize agent policies as deep neural networks (NN) using stochastic gradient ascent. We show how to backpropagate gradients through event-based rewards, which are not present in standard RL settings. The RL approach is particularly suitable for large problems with (possibly) exponential state-spaces.

We experiment on: (a) Mars-rover problem from (Becker et al. 2004), and (b) Multiagent coverage under uncertainty and partial observability. Coverage problems have received a lot of attention in multiagent and robotics literature with several applications such as vacuum cleaning robots, search-and-rescue, intrusion detection, mine clearing among others (Yehoshua and Agmon 2016; Galceran and Carreras 2013). We focus on multiagent coverage by patrol units across a mass rapid transit (MRT) system. In contrast to previous multiagent coverage settings (Yehoshua and Agmon 2016) which require full communication during execution time, our approach assumes partial observability where agents observe only their local state and cannot observe the status of other agents. This is also closer to reality in underground MRT systems where full communication is infeasible. Each agent is responsible for some *private* locations. Inspecting a private location *at least once every k* time steps give a local reward to the agent. Thus, the agent is incentivized to spread its inspections to different locations across time. There are *shared* locations also which denote interchange stations where multiple agents can inspect. Thus, agents should coordinate their inspections to shared locations given that any one agent-inspection is enough to claim the reward. We test our multiagent RL approach on the real MRT map of Singapore. The state-space is exponential for this setting — given n locations, state space is $O(2^n)$ per agent. We show that our multiagent RL scales well for this problem whereas EM and NLP fails, and also provides much better solution quality than independently optimizing agent policies, confirming the effectiveness of incorporating joint-events for computing gradients. Thus, our work significantly advances the scalability of multiagent planning for real-world problems.

2 Model Definition

We define an n-agent transition independent Dec-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 $\bar{s}$ has probability $P(\bar{s}|s,a) = \times_{i=1}^{n} P^i(\bar{s}^i|s^i,a^i)$, 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 agents' transition, observation, and the reward functions are all independent. We next describe how *joint-rewards* are defined that depend on actions of multiple agents. Such event-based joint-rewards are the key to defining a rich class of non-linear interaction among agents that can model several practical scenarios. Solving Dec-MDPs with such rich joint-reward structure is the main focus of our work.

Events: Joint-rewards are defined based on the high-level notion of *events*. We first review their definitions with detailed treatment in (Becker et al. 2004).

Definition 1. *A **primitive event** for an agent i, $e = (\hat{s}^i, a^i, \hat{s}^{i\prime})$, is a tuple including agent's local state, an action, and an outcome state. An **event** $E = \{e_1, \ldots, e_h\}$ is a set of primitive events.*

Let Φ^i denote a valid local state-action execution sequence $[s_1^i, a_1^i, s_2^i, a_2^i, \ldots]$ for an agent i (subscripts denote time).

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

Intuitively, events model the accomplishment of some high-level task by the agent. An event E is composed of multiple primitive events to account for the uncertainty in the domain and multiple ways of accomplishing the same task. E.g., consider the multiagent MRT patrolling domain. We define the high-level task for the agent is to visit a particular station once between time steps t and $t+k$. Primitive events would correspond to visiting the station at time t, $(t+1)$ through $(t+k)$. As long as any *one* of these primitive events occurs in a history, it implies the accomplishment of the task.

Definition 3. *A primitive event e is **proper** if it can occur at most once in each possible history of a given MDP:*

$$\forall \Phi^i = \Phi_1^i e \Phi_2^i \quad : \neg(\Phi_1^i \models e) \wedge \neg(\Phi_2^i \models e)$$

Definition 4. *An event E is **proper** if it consists of mutually exclusive proper primitive events w.r.t. a given MDP.*

Mutually exclusive condition implies that any two primitive events $e, e' \in E$ cannot occur together within any possible history Φ^i or we shall never observe $\Phi^i \models e \wedge \Phi^i \models e'$ for any Φ^i. Becker et al. show how non-proper events can be cast as proper events using techniques such as making time as part of the state or including additional bits in the state to memorize the occurrence of some primitive events. For algorithmic development, we focus on *proper* events.

Joint-Reward: A joint-reward is described as a *constraint* among agents that specifies how the interaction among

Variables: $x_t^i(s^i,a^i)\ \forall t, \forall s^i, a^i \forall i$;

Variables: $x(E_k^j)\ \forall j \in G_k, \forall k \in \rho$ (1)

$$\max_{\boldsymbol{x}} \sum_{i=1}^{n}\sum_{t=1}^{H}\sum_{s^i,a^i} x_t^i(s^i,a^i) r^i(s^i,a^i) + \sum_{k\in\rho} c_k \prod_{j\in G_k} x(E_k^j) \quad (2)$$

$$\sum_{a^i} x_{t+1}^i(s^{i\prime},a^i) - \sum_{s^i,a^i} x_t^i(s^i,a^i) P(s^{i\prime}|s^i,a^i) = 0\ \forall t, \forall s^{i\prime}, \forall i \quad (3)$$

$$\sum_{a^i} x_1^i(s^i,a^i) = b_1^i(s^i) \qquad \forall s^i, \forall i \quad (4)$$

$$x(E_k^j) = \sum_{e\in E_k^j}\sum_{t=1}^{H} x_t^j(s_e,a_e) P^j(s_e'|s_e,a_e)\ \forall j \in G_k, \forall k \in \rho \quad (5)$$

Table 1: Nonlinear program (NLP) for event-based TIDec-MDP

agents affects the global value of the system. A constraint k exists among a subset of agents G_k ($|G_k| \geq 2$). It is defined as a tuple $\langle\langle E_k^j\ \forall j \in G_k\rangle, c_k\rangle$. Semantically, the constraint k specifies that if each involved agent in G_k satisfies its part of the constraint, then the global reward c_k is given. Formally, let Φ^1 through Φ^n denote histories for all the agents. Constraint $\langle\langle E_k^j\ \forall j \in G_k\rangle, c_k\rangle$ specifies that the reward c_k is added to the global value iff $\Phi^j \models E_k^j$ for each agent $j \in G_k$. Let ρ be the set of all constraints; same logic is followed for each constraint $k \in \rho$.

Expressiveness: As per the above constraint semantics, global reward c_k is given if *all* events in a constraint occur. Several other types of constraint structures are possible. E.g., the global reward can be given at the occurrence of *at most x events*, *exactly x events* or *at least x events*. Becker et al. show how such variants can be reformulated using *all event occurring* semantics. However, in their approach, translating different constraint types to the standard *all event* syntax requires adding additional bits to states, which increases the state-space exponentially. In contrast, our work addresses different constraint types directly without the need to expand the state-space. Thus, our modeling and algorithmic techniques are significantly more effective and scalable.

Policy and joint-value function: 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:

$$P(e;\pi^i) = \sum_{t=1}^{H} P(s_t^i = \hat{s}^i; \pi^i)\pi^i(a^i|\hat{s}^i)P^i(\hat{s}^{i\prime}|\hat{s}^i, a^i) \quad (6)$$

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:

$$GV(\boldsymbol{s}_1;\boldsymbol{\pi}) = \sum_{i=1}^{n} V^i(s_1^i;\pi^i) + JV(\rho;\boldsymbol{\pi}) \quad (7)$$

$$JV(\rho;\boldsymbol{\pi}) = \sum_{k\in\rho} c_k \prod_{j\in G_k} P(E_k^j;\pi^j) \quad (8)$$

where V^i is the value function of agent i's local MDP. Our goal is to compute the joint-policy $\boldsymbol{\pi}$ that optimizes the global value function (7).

3 Optimization Based Formulation

An optimal algorithm is presented in (Becker et al. 2004) to solve TIDec-MDPs with event-based rewards. Given the NP-Hardness of the problem, the optimal approach is not scalable to a large number of several agents. We therefore first present a nonlinear math programming (NLP) formulation for TIDec-MDPs. The NLP formulation has been successfully used for approximately solving POMDPs (Amato, Bernstein, and Zilberstein 2007b; Charlin, Poupart, and Shioda 2007) and Dec-POMDPs (Amato, Bernstein, and Zilberstein 2007a; 2010) using off-the-shelf solvers such as SNOPT (Gill, Murray, and Saunders 2005).

Table 1 shows the NLP formulation for TIDec-MDPs. This program is similar to the standard MDP dual LP formulation using occupancy variables for state-action pairs (Puterman 1994). Variables $x_t^i(s^i,a^i)$ denote the probability of being in state s^i and taking action a^i at time t for agent i. The first part of the objective computes the total local reward of all agents, and the second summation computes the total joint-value based on (8). Constraint (3) is the standard flow conservation constraint for each state of each agent; (4) connects the initial belief b_1 at time step 1 with x variables; (5) computes the probability of events using (6). The stochastic policy $\pi(a|s)$ can be computed as $x(s,a)/\sum_a x(s,a)$. The multilinear term modeling event-based rewards in the objective make this program nonlinear and nonconvex.

We can modify the objective (2) to handle other global reward types such as *at least 1 event*. Consider the setting $\langle\langle E_k^j\ \forall j \in G_k\rangle, c_k\rangle$ where the reward c_k is given if at least one event out of $|G_k|$ happens. The probability of at least one event happening is one minus the probability that none of the events happen. It is modeled as:

$$\sum_{k\in\rho} c_k\Big(1 - \prod_{j\in G_k}\big(1 - x(E_k^j)\big)\Big) \quad (9)$$

We can similarly handle other joint-reward types such as *exactly x events* or *at most x events* by deriving the contribution of such joint-rewards to the objective analogous to (9). In addition, we can also have multiple types of joint-rewards for our model by simply adding their contribution to the objective (2) of the math program. Notice that our approach of directly modifying nonlinear objective terms does not add additional bits to states, thus do not result in exponential state-space increase as in (Becker et al. 2004).

Program in Table 1 can in principle be solved using standard NLP solvers. However, we observed empirically that

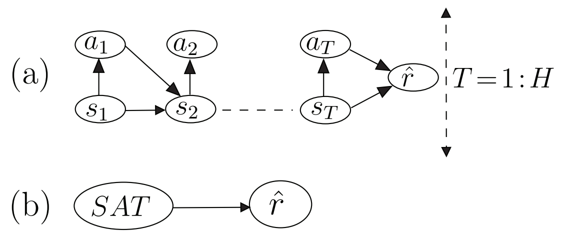


Figure 1: (a) Standard mixture model for MDP; (b) our *single* Bayesian network for policy optimization of an MDP

with the increasing number of agents, NLP solvers were unable to scale, slow and often ran out of memory. To address this, we next present a probabilistic inference based reformulation of the program in Table 1 that results in a much more scalable approach for large multiagent systems.

4 Inference for TIDec-MDPs

Planning-as-inference is a recent paradigm that translates the planning problem to that of probabilistic inference. Policy optimization in MDPs and POMDPs can be recast to that of maximum likelihood estimation (MLE) problem in a graphical model (Toussaint, Harmeling, and Storkey 2006), and have also been applied to multiagent planning (Pajarinen and Peltonen 2011; Wu, Zilberstein, and Jennings 2013; Kumar, Zilberstein, and Toussaint 2015). Recently, the problem of marginal MAP, which calculates the mode of the marginal posterior distribution of a subset of variables with the remaining variables marginalized is shown to be equivalent to decision making further establishing close connections between planning and inference (Liu and Ihler 2012; 2013). Expectation-maximization (EM) (Dempster, Laird, and Rubin 1977) is a commonly used technique to solve such planning-as-inference formulations.

We also develop here an inference based approach to solve TI-Dec-MDPs. We present a graphical model such that MLE in this model is equivalent to solving the program in Table 1. We use the EM algorithm for MLE in this graphical model. Our work differs from previous applications of EM to planning — 1) Previous works model the *sequential* aspect of decision making by using dynamic Bayes nets (DBNs) of varying length for reformulation as an inference problem. E.g., Figure 1(a) shows a T-length DBN for an MDP. In contrast, we aim to solve the math program encoding the policy optimization problem directly. E.g., Figure 1(b) shows a much simpler Bayesian net (BN) that encodes MDP policy optimization; 2) As we directly solve the math program, there is no need for the expensive forward-backward message-passing in large DBNs which can become inaccurate for infinite-horizon problems; 3) Previous DBN based methods can address immediate rewards only but unable to model event-based complex global rewards which depend on entire execution histories of agents. Our approach addresses this drawback by directly translating the program in Table 1 as an inference problem. To the best of our knowledge, our approach is one of the first to solve large nonlinear, nonconvex math programs using the probabilistic inference machinery.

4.1 Solving Dual LP for an MDP Using Inference

We first show how to formulate the dual LP for an MDP as an inference problem, which will be a sub-step in inference model for TIDec-MDPs. Given an MDP with transition function $P(s'|s,a)$, reward function $r(s,a)$ and initial state distribution b_1, our goal is to compute the policy optimizing total reward over a finite-horizon H. The dual LP for this MDP is the single-agent analogue of the program in Table 1 without nonlinear terms modeling event-based reward in objective (2) and without constraint (5). Previous works model this problem using MLE in a mixture of DBNs of varying length T (shown in Figure 1(a)). Our reformulation instead is a simple BN with two random variables SAT and $\hat{\mathrm{r}}$ as shown in Figure 1(b). The variable $\hat{\mathrm{r}}$ is a binary random variable. The variable SAT has domain $\{\langle s,a,t\rangle \forall s\in S, \forall a\in A, \forall t=1:H\}$.
We set the parameters of this BN as follows:

$$P(\hat{\mathrm{r}}=1|s,a,t)=\hat{\mathrm{r}}_{sa}=\frac{r(s,a)-\mathrm{r}_{\min}}{\mathrm{r}_{\max}-\mathrm{r}_{\min}} \quad (10)$$

$$P(s,a,t)=x_t(s,a)/H \quad (11)$$

Intuitively, the conditional probabilities $P(\hat{\mathrm{r}}=1|\cdot)$ model the scaled reward of the original MDP, and the probabilities $P(s,a,t)$ model the occupancy measures or x_t variables of the dual LP. The following result shows the equivalence of MLE and the dual LP for MDP.

Theorem 1. *Let the parameters of the BN be set as per* (10),(11). *Maximizing the likelihood* $P(\hat{\mathrm{r}}=1;\boldsymbol{x})$ *of observing* $\hat{\mathrm{r}}=1$ *in the BN subject to constraints* (3),(4) *over parameters* $\boldsymbol{x}$ *optimally solves the dual LP for the MDP.*

Proof. The dual LP objective is given as $\mathrm{dualObj} = \sum_{s,a,t} x_t(s,a)r(s,a)$. We have the probability in the BN of Figure 1(b) as:

$$P(\hat{\mathrm{r}}=1;\boldsymbol{x})=\sum_{s,a,t}P(s,a,t)P(\hat{\mathrm{r}}=1|s,a,t)$$

$$=\frac{1}{H}\sum_{s,a,t}x_t(s,a)\,\hat{\mathrm{r}}_{sa}=\frac{1}{H}\sum_{s,a,t}x_t(s,a)r(s,a)-\frac{\mathrm{r}_{\min}}{\mathrm{r}_{\max}-\mathrm{r}_{\min}}$$

The last expression implies $P(\hat{\mathrm{r}}=1;\boldsymbol{x})\propto \mathrm{dualObj}$. Thus, maximizing likelihood would also solve the dual LP for MDP. Notice that we enforce constraints (3),(4) over parameters $\boldsymbol{x}$ while maximizing the likelihood making sure that our solution is always feasible. □

Theorem 1 provides the standard setting for the EM algorithm application. The observed data is $\hat{\mathrm{r}}=1$, variable SAT is hidden, and parameters to optimize are $\boldsymbol{x}$. EM is an iterative approach that starts with random initial parameters $\boldsymbol{x}$ and maximizes the following *expected* log-likelihood to obtain a better estimate $\boldsymbol{x}^\star$.

$$Q(\boldsymbol{x},\boldsymbol{x}^\star)\propto\sum_{s,a,t}P(\hat{\mathrm{r}}=1,s,a,t;\boldsymbol{x})\log P(\hat{\mathrm{r}}=1,s,a,t;\boldsymbol{x}^\star)$$

$$\propto\sum_{s,a,t}\hat{\mathrm{r}}_{sa}\,x_t(s,a)\log x_t^\star(s,a)$$

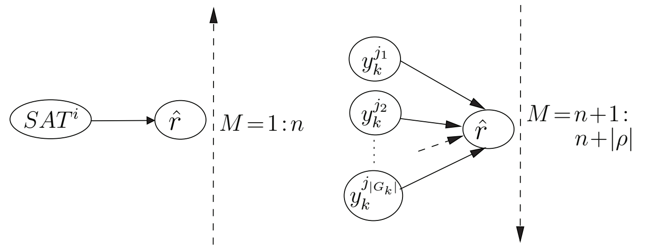


Figure 2: Mixture model for TIDec-MDP; M is mixture variable with discrete domain from 1 through $n+|\rho|$; there is one BN (left) for each agent $i=1:n$; one BN (right) for each joint-reward $k\in\rho$

where we have omitted terms independent of $\boldsymbol{x}^\star$. The convex optimization problem EM solves in each iteration is:

$$\max_{\boldsymbol{x}^\star}\sum_{s,a,t}\hat{\mathrm{r}}_{sa}\,x_t(s,a)\log x_t^\star(s,a) \tag{12}$$
$$\sum_a x_{t+1}^\star(s',a)-\sum_{s,a}x_t^\star(s,a)P(s'|s,a)=0\ \forall s'\in S, t=1:H$$
$$\sum_a x_1^\star(s,a)=b_1(s)\ \forall s\in S$$

The above program directly operates with occupancy measures x, and does not require any forward-backward message-passing like other applications of EM to planning (Toussaint, Harmeling, and Storkey 2006). Even though EM algorithm is not guaranteed to converge to a global optima, we show that it is globally optimal for MDPs:

Proposition 1. *EM algorithm converges to the global optimum of the log-likelihood for the MDP inference model in Figure 1(b).*

Proof is in the longer version of the paper. We next show how to solve the NLP in Table 1 for TIDec-MDPs analogous to the inference model and reasoning developed in this section.

4.2 Inference Model for TIDec-MDP

We now present a mixture of BNs such that MLE in the mixture is equivalent to solving the NLP in Table 1. In EM, optimizing the expected log-likelihood (or the M-step) becomes decoupled resulting in a separate optimization problem for each agent regardless of the number of joint rewards or the number of agents in a joint-reward. This is a significant scalability boost as NLP solvers directly optimize the monolithic program in Table 1 which quickly becomes unscalable due to a large number of variables/nonlinear terms, whereas EM solves an independent *convex* program for each agent.

Figure 2 shows the mixture of BNs for TIDec-MDPs. We create one BN for each agent i (left Figure 2) and one BN for each global reward $k\in\rho$. The left BN simulates the total local reward from agent i's MDP, similar to the BN for an MDP. The structure and interpretation of this BN is same as in previous section. The mixture variable M can take any integer value in range $[1,n+|\rho|]$ to index each BN, and has a fixed uniform distribution (=$1/(n+|\rho|)$).

To simulate event-based rewards, the right BN (in Figure 2) is created for each global reward $k\in\rho$. The variable $\hat{\mathrm{r}}$ is binary as before. In addition, we create one *binary* variable y_k^j for each agent $j\in G_k$ involved in the global reward $\langle\langle E_k^j\ \forall j\in G_k\rangle, c_k\rangle$. Intuitively, $y_k^j=1$ implies agent j's event E_k^j occur; vice versa for $y_k^j=0$. W.l.o.g. we assume all rewards c_k are positive (otherwise we can subtract $\mathrm{r}_{\min}$ from them), and θ is a positive constant. Let $\boldsymbol{y}_k$ denote the random vector $(y_k^j\forall j\in G_k)$. Probabilities for this BN are as:

$$P(y_k^j=1)=x(E_k^j);\ P(y_k^j=0)=1-x(E_k^j)\ \forall j\in G_k \tag{13}$$
$$P(\hat{\mathrm{r}}=1|\boldsymbol{y}_k)=\begin{cases}(\theta+c_k)/\,\mathrm{r}_{\max} & \text{iff } y_k^j=1\forall j\in G_k\\ \theta/\,\mathrm{r}_{\max} & \text{otherwise}\end{cases} \tag{14}$$

where positive constant $\mathrm{r}_{\max}$ is chosen such that $(\theta+c_k)/\,\mathrm{r}_{\max}$ is less than one (i.e. valid probabilities). Intuitively, (13) connect probabilities of variables y_k^j with parameters $x(E_k^j)$ used in the NLP of Table 1, (14) model the condition that global reward c_k is given only when all events E_k^j happen, otherwise a default reward is awarded.

Theorem 2. *Let the parameters of the BNs in the mixture model (Figure 2) be set as per* (10),(11),(13),(14). *Maximizing the likelihood $P(\hat{\mathrm{r}}=1;\boldsymbol{x})$ of observing $\hat{\mathrm{r}}=1$ in the BN subject to constraints* (3),(4),(5) *over parameters $\boldsymbol{x}$ optimally solves the NLP for the TIDec-MDP.*

Proof is provided in the longer version of the paper.

M-step Given theorem 2, we can again use the EM algorithm similar to section 4.1 to iteratively optimize the expected log-likelihood. We directly show the M-step optimization problem (analogous to problem (12)) for TIDec-MDPs. It involves solving one *independent* convex optimization problem for each agent i.

$$\max_{\boldsymbol{x}^{i\star}}\sum_{s^i,a^i,t}\hat{\mathrm{r}}^i_{s^ia^i}\,x_t^i(s^i,a^i)\log x_t^{i\star}(s^i,a^i)+$$
$$\sum_{k\in\rho(i)}\hat{c}_k\Big(\prod_{j\in G_k}x(E_k^j)\Big)\log x^\star(E_k^i)-\hat{\theta}\sum_{k\in\rho(i)}H_k(x^i,x^{i\star}) \tag{15}$$
$$\sum_{a^i}x_{t+1}^{i\star}(s^{i\prime},a^i)-\sum_{s^i,a^i}x_t^{i\star}(s^i,a^i)P(s^{i\prime}|s^i,a^i)=0\forall t,\forall s^{i\prime} \tag{16}$$
$$\sum_{a^i}x_1^{i\star}(s^i,a^i)=b_1^i(s^i)\qquad\forall s^i,\forall i \tag{17}$$
$$x^\star(E_k^i)=\sum_{e\in E_k^i}\sum_{t=1}^{H}x_t^{i\star}(s_e,a_e)P^j(s_e'|s_e,a_e)\ \forall k\in\rho(i) \tag{18}$$

where $\hat{\mathrm{r}}^i_{s^ia^i}$ is the normalized local reward for the local MDP of agent i; $\hat{c}_k$ is the normalized global reward ($c_k/\,\mathrm{r}_{\max}$); $\hat{\theta}$ is $\theta/\,\mathrm{r}_{\max}$; $\rho(i)$ denotes the set of joint-rewards in which agent i participates. For a global reward k, H_k denote the cross entropy between previous iteration's parameters x^i and current parameter $x^{i\star}$:$-[x^i(E_k^i)\log x^{i\star}(E_k^i)+x^i(\tilde{E}_k^i)\log x^{i\star}(\tilde{E}_k^i)]$, where $x^i(\tilde{E}_k^i)=1-x^i(E_k^i)$.

The above M-step optimization highlights the scalability of the EM algorithm which solves one *separate* optimization problem for each agent i. Note that even though the M-step solves each agent's sub-problem separately, these sub-problems are still correlated as each sub-problem involves information from other agents and events from previous iteration's solution. Since previous iteration's solution is fixed, different sub-problems can still be solved separately. Each sub-problem is convex and much smaller than the large NLP in Table 1. Thus, the EM algorithm significantly boosts the scalability for a large number of agents.

Other joint-reward types: The EM approach is also extendible to other joint-reward types. In Figure 2, we construct one separate Bayesian network (BN) for each joint-reward $k \in \rho$. Based on the type of the joint-reward k (*at most x events* or *at least x events*), we can set the parameters of its corresponding BN appropriately by modifying (14).

5 RL for Event-Based Rewards

The previous section presented a scalable EM algorithm for TIDec-MDPs. However, the scalability still suffers when the state-space of each agent i is exponential, which is often the case for several patrolling and coverage problems (Yehoshua, Agmon, and Kaminka 2015). To address such settings, we develop a reinforcement learning (RL) approach that uses function approximators such as deep neural nets (NN) to represent agent policies and optimizes them using the policy gradient approach (Williams 1992; Sutton et al. 1999). Policy gradient is a natural counterpart to the EM algorithm— Schulman et al. (2015) show that EM optimizes a *surrogate loss* function by minimizing an upper bound on this function, whereas policy gradient uses gradient descent to minimize the same loss function. Empirically, we observed that for small and medium-sized problems, EM is faster (RL is slower but comparable in solution quality). The main advantage of RL lies for large problems where EM cannot scale well due to its tabular policies.

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 represent NN parameters), our goal is to compute the gradient of global-value function (7):

$$\nabla_{\theta^i} GV(\boldsymbol{s}_1;\boldsymbol{\pi}) = \nabla_{\theta^i} V^i(s_1^i;\pi^i) + \nabla_{\theta^i} \sum_{k\in\rho(i)} c_k \prod_{j\in G_k} P(E_k^j) \quad (19)$$

$$= \nabla_{\theta^i} V^i(s_1^i;\pi^i) + \sum_{k\in\rho(i)} c_k \nabla_{\theta^i} P(E_k^i) \prod_{j\in G_k\setminus\{i\}} P(E_k^j) \quad (20)$$

where we have used the fact that agent i's policy does not affect agent j's local value function V^j and event probabilities $P(E^j)$ to simplify the expression; $\rho(i)$ denotes the set of joint-rewards in which agent i participates. The gradient of local MDP value function $\nabla_{\theta^i} V^i(s_1^i;\pi^i)$ can be computed using the technique in (Sutton et al. 1999). We next focus on how the gradient backpropagates from event-based rewards. Given that we have $P(E;\pi^i) = \sum_{e\in E} P(e;\pi^i)$, gradient is:

$$\nabla_{\theta^i} P(E;\pi^i) = \sum_{e\in E} \nabla_{\theta^i} P(e;\pi^i) \quad (21)$$

We next compute the gradient of a proper, primitive event or $\nabla_{\theta^i} P(e;\pi^i)$. Let $e = \{\hat{s}^i, \hat{a}^i, \hat{s}^{i\prime}\}$ be a primitive event for agent i. The probability of this event is given as:

$$P(e) = \sum_{t=1}^{H} P^\pi(\hat{s}^i|t)\pi^i(\hat{a}^i|\hat{s}^i)P(\hat{s}^{i\prime}|\hat{s},\hat{a}^i)$$
$$= \sum_{t=1}^{H} \sum_{s^i_{1:t+1}, a^i_{1:t}} P^\pi(s^i_{1:t+1}, a^i_{1:t}) \mathbb{I}(\langle s^i_t, a^i_t, s^i_{t+1}\rangle = \langle \hat{s}^i, \hat{a}^i, \hat{s}^{i\prime}\rangle)$$

where $\mathbb{I}$ is the indicator function returning one when input logical condition is true, zero otherwise. The gradient is:

$$\nabla_{\theta^i} P(e) = \sum_{t=1}^{H} \sum_{s^i_{1:t+1}, a^i_{1:t}} P^\pi(s^i_{1:t+1}, a^i_{1:t}) \Big[\sum_{t'=1}^{t} \nabla_{\theta^i} \log \pi^i(a^i_{t'}|s^i_{t'})\Big] \times$$
$$\mathbb{I}(\langle s^i_t, a^i_t, s^i_{t+1}\rangle = \langle \hat{s}^i, \hat{a}^i, \hat{s}^{i\prime}\rangle) \quad (22)$$

where we used the log derivative trick (Schulman et al. 2015) that allows sampling-based evaluation of $\nabla_{\theta^i} P(e)$. Consider a complete state-action trajectory $\Phi^i = (s^i_{1:H+1}, a^i_H)$. For any Φ^i, the proper, primitive event e can only occur *at most once* (see definition 3). Based on this fact, we derive the stochastic gradient estimate as (complete derivation in paper's extended version):

$$\nabla_{\theta^i} P(e) = \frac{1}{|\xi|} \sum_{\Phi^i\in\xi:\Phi^i\models e} \Bigg(\sum_{t'=1}^{t(e,\Phi^i)} \nabla_{\theta^i} \log \pi^i(a_{t'}|s_{t'}) \Bigg) \quad (23)$$

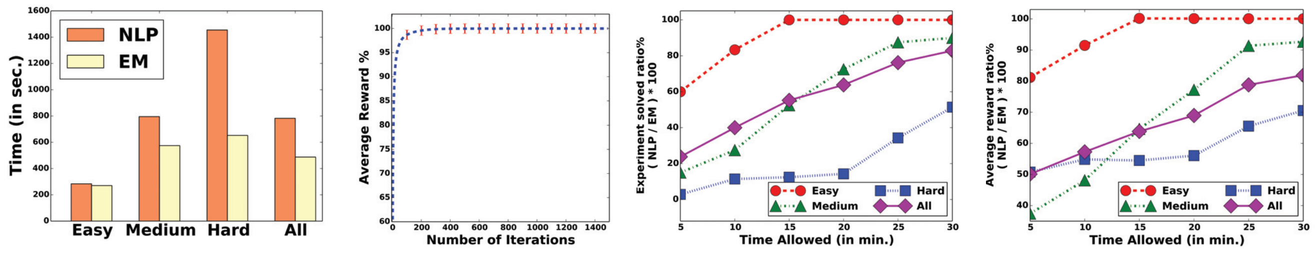
ξ is the set of complete state-action samples from $P^\pi(s^i_{1:H+1}, a^i_{1:H})$; we sum over those samples Φ^i such that event e occurs in Φ^i at some point in time (denoted using $t(e,\Phi^i)$). We can use the above gradient within (21) to compute the gradient of an event, and use it in (20) to compute the required gradient. Given sample set ξ for each agent, we can also empirically compute the probability estimates of primitive events e, and use it to compute empirical estimate of events $P(E^j_k)$, which can be used in (20). Thus, we have shown how the policy gradient $\nabla_{\theta^i} GV$ for TIDec-MDPs can be computed using sampling.

Other joint-reward types: We can also derive policy gradient for other types of events mentioned in section 2. E.g., we show the gradient of $JV(\rho;\boldsymbol{\pi})$ (in eq (8)) when constraints $k \in \rho$ are of the type *at least one event*:

$$\nabla_{\theta^i} JV = \sum_{k\in\rho(i)} c_k \nabla_{\theta^i} P(E^i_k) \prod_{j\in G_k\setminus\{i\}} \big(1 - P(E^j_k)\big) \quad (24)$$

One can compute $\nabla_{\theta^i} P(E^i_k)$ as earlier.

Multiagent credit assignment: A key problem while learning from global rewards in multiagent setting is that the gradient computed for an agent i does not explicitly reason about the contribution of that agent to the global team reward. As a result, the gradient becomes noisy given that other agents are also exploring, leading to poor quality solutions (Foerster et al. 2017; Bagnell and Ng 2005). Fortunately, creating a separation among local MDPs of agents and joint event-based rewards automatically addresses this problem of noisy gradient in TIDec-MDPs. E.g., consider the gradient (24) for *at least one event* w.r.t. agent i's parameters. Intuitively, if another agent j has a high probability of finishing event E^j_k(or $P(E^j_k) \approx 1$), then the term $(1 - P(E^j_k))$ in $\nabla_{\theta^i} JV$ would discourage agent i to increase its probability $P(E^i_k)$. Similarly, if all other agents j have a very low probability of performing E^j_k then $(1 - P(E^j_k))$ would be high encouraging agent i to perform its event E^i_k for higher joint-value. Thus, event-based rewards help make policy gradients precise and accurate in TIDec-MDPs.



(a) Runtime Analysis (b) EM: Anytime Nature (c) Experiments Solved ratio (d) Average Reward ratio

Figure 3: Runtime and Solution Quality Results for Mars rover domain

6 Experiments

We tested on two domains—Mars rover and the Multiagent coverage problem. Domain details and other experimental settings are provided in the extended version of the paper.

Mars rover domain: This domain is adapted from (Becker et al. 2004). Agents have private and shared locations from which they can collect data. Joint-rewards are given when all the involved agents perform their corresponding events on the shared location.

Runtime: Figure 3a shows the runtime results for NLP and EM for the four categories, $\langle \mathrm{Easy}, \mathrm{Medium}, \mathrm{Hard}, \mathrm{All} \rangle$, of problems on the x-axis. The results are averaged over $\langle 30, 40, 35, 105 \rangle$ instances corresponding to the four categories ('All' includes all the instances). The categories $\langle \mathrm{Easy}, \mathrm{Medium}, \mathrm{Hard} \rangle$ include $\langle 10, 30, 40 \rangle$ agents, $\langle 4, 5, 7 \rangle$ locations per agent and $\langle 10, 18, 25 \rangle$ agents sharing a common site respectively.

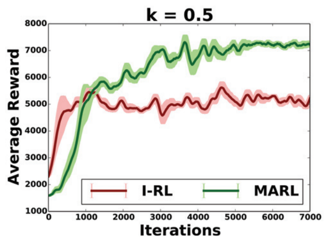
Figure 3a shows that EM has a much lower runtime on an average. Only those problems that got solved by NLP within the cutoff time of 30 minutes were included in the figure (EM being an anytime algorithm, always returns a solution). For the hard instances, EM was much faster than the NLP solver showing that EM's strategy of solving independent program per agent results in significant speedups over NLP. Figure 3b highlights the anytime nature of EM by showing the normalized solution quality of EM for each iteration averaged over 35 Hard instances. The solution quality increases monotonically with iterations till convergence and reaches to $\langle 98.7\%, 99.6\% \rangle$ of final converged quality within the first $\langle 100, 200 \rangle$ iterations. Similar trends hold for Easy and Medium problems.

Solution Quality: Figures 3c and 3d present solution quality results for NLP and EM. In particular, the x-axis of both the figures show the cutoff time in minutes, i.e., maximum allowable time for the algorithms to return a solution. The y-axis of Figure 3c shows the ratio (($\#$ of instances solved by NLP/$\#$ of instances solved by EM) (in $\%$) for the time limit indicated on the x-axis. Figure 3c shows that the ratio increases monotonically as the time limit increases for all categories. For Hard instances, NLP could provide a solution for only 40% of problems, and did not terminate for the rest of the problems. In contrast, EM always converged within 30 min for all the instances, confirming its better scalability.

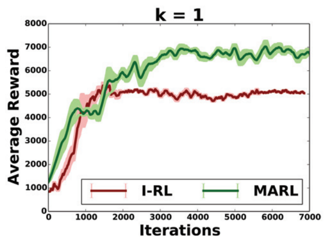
The y-axis of Figure 3d shows the ratio (in $\%$) of total average rewards obtained by NLP w.r.t. the EM within the cutoff time on the x-axis. This figure captures the reward for all the experiments performed. That is, if NLP does not return a solution within the cutoff time, a valid random policy was assigned for evaluation purposes. Trends remain similar as in Figure 3c. For Easy category, the ratio becomes 100% within 15 minutes while it remains below 71% for Hard cases even after 30 minutes (and does not reach 100% for any other category). To summarize, the anytime property of EM provides a significant runtime as well as solution quality advantage (with performance gap widening significantly as we move from Easy to the Hard category of problems).

Multiagent coverage problem: For testing the scalability of our multiagent RL (MARL), we experimented with the multiagent coverage problem introduced in section 1 (detailed domain settings are in the extended version of the paper). The problem involves inspecting different locations within a mass rapid transit (MRT) network. Reward is given when a location is inspected at least once within a fixed timeframe, such as every 1 hour or half hour. These problems are challenging for EM/NLP as the state-space is exponential in the number of locations in the MRT system. We first tested MARL on relatively tractable instances that were solvable by EM. These instances involved up to 2 lines in the MRT map, with a maximum of 3 private locations per line, and a maximum of 3 shared locations. Figure 5a shows that MARL achieves similar solution quality as EM for all the instances (within 97% of EM's quality on average). However, the average runtime of MARL (5 hours) was significantly higher than EM's runtime (10 minutes). Thus, for relatively smaller instances both EM and MARL provide similar quality, but EM is preferable because of its lower runtime.

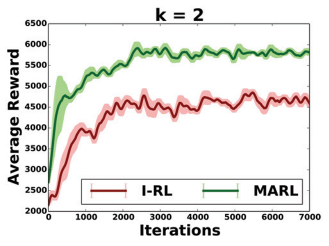
For hard settings, we tested on the Singapore MRT map with 5 lines, each having 20 private locations per line, and 20 shared locations. Each line has a single agent able to move among locations on the line. Shared locations correspond to interchange stations where multiple lines meet. Thus, shared locations can be inspected by multiple agents. The time horizon was 1024 minutes (17 hours). The joint reward to inspect any shared location was much higher than private locations given that shared locations are heavily crowded and thus



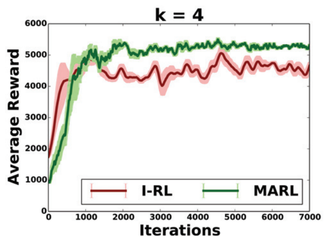
(a) Gain = 36.1%



(b) Gain = 29.6%



(c) Gain = 26.0%

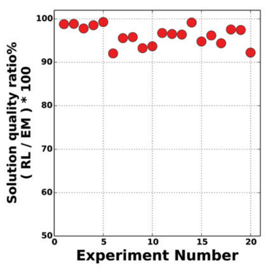


(d) Gain = 15.2%

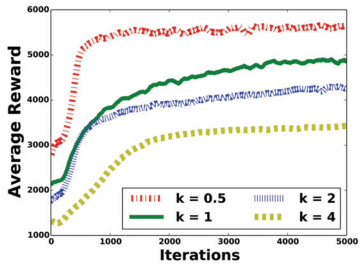
Figure 4: Quality comparisons between coordinated RL (MARL) and independent policy optimization (I-RL) for varying reset time k (in hours). Gain is % quality improvement by MARL over I-RL upon convergence

more important. To claim the reward, agents must successfully inspect a location once every k time steps (also called *reset* time), and k (in hours) varied in the range $\{0.5, 1, 2, 4\}$. The inspect action consumes 15 minutes, and moving to the next location on the line takes 3 minutes. Inspecting a location multiple times within the reset time window does not fetch an additional reward. Thus, agents are required to increase their coverage to gain additional rewards. Similarly, for shared locations, inspection by any single agent is sufficient to get the reward. Thus, agents are also incentivized to coordinate with each other to avoid multiple agents inspecting the same shared location within the reset time window. We model such global rewards using the *at least one event* semantics.

For these problems, EM and NLP were unable to scale due to the large state-space of the problem. However, MARL scaled well for these problems due to its neural network based parametric policies and policy gradient based optimization (settings for neural network and gradient ascent are provided in the extended version of the paper). For figures 4 and 5b, y-axis shows average reward value at each iteration which is computed as a moving average of last 100 iteration values (i.e., value at that iteration and previous 99 values). Figure 5b shows the average reward quality achieved by MARL for different settings of the reset time k. A smaller value of k implies agents can claim rewards frequently. Therefore, we can observe that the reward is higher for lower k values. For all the settings, despite the large problem size with a long horizon, convergence was achieved in about 5000 iterations.



(a) Easy



(b) Hard

Figure 5: Solution Quality - Multiagent RL

Coordinated vs. Uncoordinated RL: We tested the MARL approach against a baseline method that independently optimized the policy of each agent using policy gradient based RL ('uncoordinated' RL). We also observed that when shared locations are fewer, agents need to coordinate tightly to gain the reward from inspecting these shared locations. To test this hypothesis, we reduced the number of shared locations down to 10, from 20 in the real map. Figure 4 shows quality improvements by MARL over independent policy optimization (I-RL). The figure shows four plots for values of k (in hours) varied in the range $\{0.5, 1, 2, 4\}$. The uncertainty in the figure represents the standard deviation in a moving interval of 100 iterations. From the plots we can see that the benefit of MARL is higher for lower reset time (k) as it provides more opportunities to inspect shared locations and claim higher reward from coordinated actions. These results clearly illustrate that our MARL approach provides significant benefits when agents learn in a coordinated fashion versus learning independently.

7 Conclusion

We addressed multiagent decision making in settings where joint-rewards may depend upon entire state-action histories of agents. Such history-dependent rewards can capture the notion of events and tasks in multiagent planning. We developed a scalable approach for this setting by translating the problem to that of inference in a graphical model. The resulting EM algorithm was shown to be more scalable than the standard nonlinear program. We also developed a multiagent RL approach that optimized agent policies represented as neural networks using stochastic policy gradients. Our inference and RL-based advances enabled us to solve large synthetic problems and a realistic multiagent coverage problem for schedule coordination of agents in an MRT network where other approaches failed to scale.

Acknowledgements

This research was supported by the Singapore Ministry of Education (MOE) Academic Research Fund (AcRF) Tier 1 grant. We also thank Kohli Center on Intelligent Systems, International Institute of Information Technology, Hyderabad for the generous support.

References

Amato, C.; Bernstein, D. S.; and Zilberstein, S. 2007a. Optimizing memory-bounded controllers for decentralized POMDPs. In *International Conference on Uncertainty in Artificial Intelligence*, 1–8.

Amato, C.; Bernstein, D. S.; and Zilberstein, S. 2007b. Solving POMDPs using quadratically constrained linear programs. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 2418–2424.

Amato, C.; Bernstein, D. S.; and Zilberstein, S. 2010. Optimizing fixed-size stochastic controllers for POMDPs and decentralized POMDPs. *Autonomous Agents and Multi-Agent Systems* 21(3):293–320.

Amato, C.; Konidaris, G.; Cruz, G.; Maynor, C. A.; How, J. P.; and Kaelbling, L. P. 2015. Planning for decentralized control of multiple robots under uncertainty. In *IEEE International Conference on Robotics and Automation, ICRA*, 1241–1248.

Bagnell, J. A., and Ng, A. Y. 2005. On local rewards and scaling distributed reinforcement learning. In *International Conference on Neural Information Processing Systems*, 91–98.

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.

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.

Charlin, L.; Poupart, P.; and Shioda, R. 2007. Automated hierarchy discovery for planning in partially observable environments. In *Advances in Neural Information processing Systems*, 225–232.

Dempster, A. P.; Laird, N. M.; and Rubin, D. B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical society, Series B* 39(1):1–38.

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.

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.

Gill, P. E.; Murray, W.; and Saunders, M. A. 2005. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Rev.* 47:99–131.

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

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.

Kumar, A.; Zilberstein, S.; and Toussaint, M. 2015. Probabilistic inference techniques for scalable multiagent decision making. *Journal of Artificial Intelligence Research* 53(1):223–270.

Liu, Q., and Ihler, A. T. 2012. Belief propagation for structured decision making. In *International Conference on Uncertainty in Artificial Intelligence*, 523–532.

Liu, Q., and Ihler, A. T. 2013. Variational algorithms for marginal MAP. *Journal of Machine Learning Research* 14(1):3165–3200.

Nair, R.; Varakantham, P.; Tambe, M.; and Yokoo, M. 2005. Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. In *AAAI Conference on Artificial Intelligence*, 133–139.

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*.

Pajarinen, J., and Peltonen, J. 2011. Efficient planning for factored infinite-horizon dec-pomdps. In *International Joint Conference on Artificial Intelligence*, 325–331.

Pajarinen, J.; Hottinen, A.; and Peltonen, J. 2014. Optimizing spatial and temporal reuse in wireless networks by decentralized partially observable Markov decision processes. *IEEE Trans. on Mobile Computing* 13(4):866–879.

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

Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY, USA: John Wiley & Sons, Inc., 1st edition.

Robbel, P.; Oliehoek, F. A.; and Kochenderfer, M. J. 2016. Exploiting anonymity in approximate linear programming: Scaling to large multiagent MDPs. In *AAAI Conference on Artificial Intelligence*, 2537–2543.

Scharpff, J.; Roijers, D. M.; Oliehoek, F. A.; Spaan, M. T. J.; and de Weerdt, M. M. 2016. Solving transition-independent multiagent mdps with sparse interactions. In *AAAI Conference on Artificial Intelligence*, 3174–3180.

Schulman, J.; Heess, N.; Weber, T.; and Abbeel, P. 2015. Gradient estimation using stochastic computation graphs. In *International Conference on Neural Information Processing Systems*, 3528–3536.

Sonu, E.; Chen, Y.; and Doshi, P. 2015. Individual planning in agent populations: Exploiting anonymity and frame-action hypergraphs. In *International Conference on Automated Planning and Scheduling*, 202–210.

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.

Toussaint, M.; Harmeling, S.; and Storkey, A. 2006. Probabilistic inference for solving (PO)MDPs. Technical report, University of Edinburgh, Edinburgh, UK.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8(3):229–256.

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.

Wu, F.; Zilberstein, S.; and Jennings, N. R. 2013. Monte-carlo expectation maximization for decentralized POMDPs. In *International Joint Conference on Artificial Intelligence*, 397–403.

Yehoshua, R.; Agmon, N.; and Kaminka, G. A. 2015. Frontier-based RTDP: A new approach to solving the robotic adversarial coverage problem. In *International Conference on Autonomous Agents and Multiagent Systems*, 861–869.

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