Decentralized Multi-Agent Reinforcement Learning in Average-Reward Dynamic DCOPs

Duc Thien Nguyen[†], William Yeoh[‡], Hoong Chuin Lau[†], Shlomo Zilberstein[⋆], and Chongjie Zhang[∗]

[†]School of Information Systems, Singapore Management University, Singapore 178902
{dtnguyen.2011,hclau}@smu.edu.sg

[‡]Department of Computer Science, New Mexico State University, Las Cruces, NM 88003, USA
wyeoh@cs.nmsu.edu

[⋆]School of Computer Science, University of Massachusetts, Amherst, MA 01003, USA
shlomo@cs.umass.edu

[∗]CSAIL, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
chongjie@csail.mit.edu

Abstract. Researchers have introduced the *Dynamic Distributed Constraint Optimization Problem* (Dynamic DCOP) formulation to model dynamically changing multi-agent coordination problems, where a dynamic DCOP is a sequence of (static canonical) DCOPs, each partially different from the DCOP preceding it. Existing work typically assumes that the problem in each time step is decoupled from the problems in other time steps, which might not hold in some applications. Therefore, in this paper, we make the following contributions: (*i*) We introduce a new model, called *Markovian Dynamic DCOPs* (MD-DCOPs), where the DCOP in the next time step is a function of the value assignments in the current time step; (*ii*) We introduce two distributed reinforcement learning algorithms, the Distributed RVI Q-learning algorithm and the Distributed R-learning algorithm, that balance exploration and exploitation to solve MD-DCOPs in an online manner; and (*iii*) We empirically evaluate them against an existing multi-arm bandit DCOP algorithm on dynamic DCOPs.

1 Introduction

Distributed Constraint Optimization Problems (DCOPs) are problems where agents need to coordinate their value assignments to maximize the sum of the resulting constraint utilities [21, 24, 33]. They are well-suited for modeling multi-agent coordination problems where the primary interactions are between local subsets of agents, such as the distributed scheduling of meetings [20], the distributed allocation of targets to sensors in a network [9], the distributed allocation of resources in disaster evacuation scenarios [15], and the distributed coordination of logistics operations [17].

Unfortunately, DCOPs only model static problems or, in other words, problems that do not change over time. In the above-mentioned coordination problems, various events that change the problem can occur. As a result, researchers have extended DCOPs to *Dynamic DCOPs*, where the problem can change over time [25, 26, 16, 31, 37]. Researchers have thus far taken an *online* approach by modeling it as a sequence of (canonical) DCOPs, each partially different from the DCOP preceding it, and solving it by searching for a new solution each time the problem changes. However, existing work typically assumes that the problem in each time step is decoupled from the problems in other time steps, which might not hold in some applications.

Therefore, in this paper, we introduce a new model, called *Markovian Dynamic DCOPs* (MD-DCOPs), where the DCOP in the next time step is a function of the value assignments in the current time step. Similar to existing work on dynamic DCOPs, we assume that the agents in MD-DCOPs are not aware of the underlying transition functions and, thus, need to solve the problem in an online manner. Specifically, we introduce two reinforcement learning algorithms, the distributed RVI Q-learning algorithm and the distributed R-learning algorithm, that use a multi-arm bandit strategy to balance exploration (learning the underlying transition functions) and exploitation (taking the currently believed optimal joint action). We empirically evaluate them against an existing multi-arm bandit DCOP algorithm on dynamic DCOPs.

2 Background: DCOPs

A *distributed constraint optimization problem* (DCOP) [21, 24, 33] is defined by $\langle \mathcal{X}, \mathcal{D}, \mathcal{F}, \mathcal{A}, \alpha \rangle$, where $\mathcal{X} = \{x_1, \ldots, x_n\}$ is a set of *variables*; $\mathcal{D} = \{D_1, \ldots, D_n\}$ is a set of finite *domains*, where D_i is the domain of variable x_i; $\mathcal{F} = \{f_1, \ldots, f_m\}$ is a set of binary *reward functions* (also called *constraints*), where each reward function $f_i : D_{i_1} \times D_{i_2} \mapsto \mathbb{N} \cup \{-\infty, 0\}$ specifies the reward of each combination of values of variables x_{i_1} and x_{i_2} that are in the function's scope; $\mathcal{A} = \{a_1, \ldots, a_p\}$ is a set of *agents* and $\alpha : \mathcal{X} \rightarrow \mathcal{A}$ maps each variable to one agent. Although the general DCOP definition allows one agent to own multiple variables as well as the existence of k-ary constraints, we restrict our definition here for in the interest of clarity. One can transform a general DCOP to our DCOP using pre-processing techniques [34, 8, 3]. We will thus use the terms "agent" and "variable" interchangeably. A solution is a value assignment for a subset of variables. Its reward is the evaluation of all reward functions on that solution. A solution is *complete* iff it is a value assignment for all variables. The goal is to find a reward-maximal complete solution.

3 Motivating Domain

We motivate our work by a sensor network application, where a group of sensors need to coordinate to track targets in a grid. Figure 1 illustrates this problem, where there are three sensors a_1, a_2, and a_3, and two targets t_1 and t_2. The possible locations of target t_1 are denoted by "X's" and the possible locations of target t_2 are denoted by filled circles. The possible areas that sensor a_1 can scan are denoted by dotted circles in Figure 1(a), and the possible areas that sensors a_2 and a_3 can scan are denoted by dotted circles in Figure 1(b). Therefore, sensor a_2 can only track target t_1, sensor a_3 can only track target t_2, and sensor a_1 can track both targets.

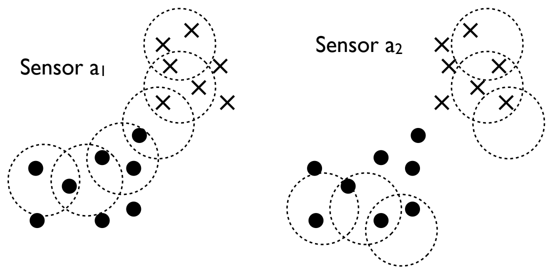

Fig. 1. Sensor Network Example

Each target has an associated reward if it is found by any one sensor and a larger reward if it is found by two sensors. The reward is also location dependent – it is larger in locations of higher importance. The sensors need to coordinate with one another to decide which areas they would each like to scan at each time step for an infinite horizon. Additionally, the targets can observe the areas scanned by the sensors in the current time step, which will influence its choice of location to move to in the next time step. The sensors do not know how the targets decide to move.

If the sensors need to coordinate for a single time step only and there is no uncertainty in the target movements, then one can represent this problem as a (canonical) DCOP, where the sensors are agents, the areas that a sensor can scan are its domain values, and the coordination between sensors to track each target is represented as a constraint.

4 Markovian Dynamic DCOPs

We now introduce the Markovian Dynamic DCOP (MD-DCOP) model. At a high level, an MD-DCOP can be visualized as a sequence of (canonical) DCOPs with one (canonical) DCOP at each time step. The variables $\mathcal{X}$, domains $\mathcal{D}$, agents $\mathcal{A}$, and ownership mapping α of the initial DCOP remains unchanged across all time steps, but the reward functions $\mathcal{F}$ *can* change and is a function of the global joint state $\mathbf{s}$ at the current time step. The problem transitions from one global joint state to another in the next time step according to a pre-defined joint transition probability function, which is a function of the values of the agents in the current time step. In this paper, we assume that the agents do not know this underlying joint transition probability function.

In more detail, an MD-DCOP is defined by a tuple $\langle \mathbf{S}, \mathbf{D}, \mathbf{P}, \mathbf{F} \rangle$, where

S is the finite set of global joint states. $\mathbf{S} = \times_{1 \leq i \leq m} \mathbf{S}_i$, where $\mathbf{S}_i$ is the set of local states of reward function $f_i \in \mathcal{F}$. Each global joint state $\mathbf{s} \in \mathbf{S}$ is defined by $\langle \mathbf{s}_1, \ldots, \mathbf{s}_m \rangle$, where $\mathbf{s}_i \in \mathbf{S}_i$.

D is the finite set of global joint values. $\mathbf{D} = \times_{1 \leq i \leq n} D_i$, where $D_i \in \mathcal{D}$ is the set of local values of variable x_i. Each global joint value $\mathbf{d} \in \mathbf{D}$ is defined by $\langle d_1, \ldots, d_n \rangle$, where $d_i \in D_i$. We also use the notations $\mathbf{D}_i = \langle d_{i_1}, d_{i_2} \rangle$, where $d_{i_1} \in D_{i_1}$ and $d_{i_2} \in D_{i_2}$, to denote the set of local joint values of variables x_{i_1} and x_{i_2} that are in the scope of reward function $f_i \in \mathcal{F}$ and $\mathbf{d}_i \in \mathbf{D}_i$ to denote an element of this set.

P is the finite set of joint transition probability functions that assume conditional transition independence. $\mathbf{P} = \times_{\mathbf{s},\mathbf{s}' \in \mathbf{S},\mathbf{d} \in \mathbf{D}} P(\mathbf{s}' \mid \mathbf{s}, \mathbf{d})$, where $P(\mathbf{s}' \mid \mathbf{s}, \mathbf{d}) = \Pi_{1 \leq i \leq m} P_i(\mathbf{s}'_i \mid \mathbf{s}_i, \mathbf{d}_i)$ is the probability of transitioning to joint state $\mathbf{s}'$ after taking joint value $\mathbf{d}_i$ in joint state $\mathbf{s}$. In this paper, we assume that the underlying joint transition probably functions are *not known* to the agents a priori.

F is the finite set of joint reward functions. $\mathbf{F}(\mathbf{s}, \mathbf{d}) = \sum_{1 \leq i \leq m} f_i(\mathbf{s}_i, \mathbf{d}_i)$, where $f_i(\mathbf{s}_i, \mathbf{d}_i)$ is the reward of taking joint value $\mathbf{d}_i$ in joint state $\mathbf{s}_i$.

This model bears a lot of similarities with factored Markov decision processes (MDPs) [7, 11], multi-agent MDPs [6] and decentralized MDPs [5], which we discuss in detail in Section 7.

In our sensor network example, if constraint f_i represents the coordination between sensors s_1 and s_2 to track target t_1, then each state $\mathbf{s}_i$ represents the possible locations of target t_1; each value $\mathbf{d}_i$ represents a pair of areas that the agents s_1 and s_2 can scan; the probability function $P_i(\mathbf{s}'_i \mid \mathbf{s}_i, \mathbf{d}_i)$ represents the movement strategy of target t_1; and the reward function $f_i(\mathbf{s}_i, \mathbf{d}_i)$ denotes the joint reward of sensors in those zones if they took joint movement $\mathbf{d}_i$ in state $\mathbf{s}_i$.

A solution to an MD-DCOP is a global joint policy $\Pi : \mathbf{S} \mapsto \mathbf{D}$ that maps each global joint state $\mathbf{s} \in \mathbf{S}$ to a global joint value $\mathbf{d} \in \mathbf{D}$. The objective of an MD-DCOP is for the agents to assign a sequence of values to their variables (to learn the underlying transition probability function and explore the state space) and converge on a global joint policy that together maximizes the expected average reward:

The objective of an MD-DCOP is find a global joint policy that maximizes the expected average reward, which is defined as follows:

Definition 1. *The expected average reward of an MD-DCOP policy Π found using exploration strategy Φ is*

$$\lim_{T \to \infty} \frac{1}{T} \mathbb{E} \left[\sum_{t=0}^{T-1} \sum_{i=1}^{m} f_i(\mathbf{s}_i, \mathbf{d}_i^t) \right] \tag{1}$$

where $\mathbf{d}_i^t$ is the local joint value for local state $\mathbf{s}_i$ at time step t. Let t^* be the time step in which all the agents converged on the global joint policy Π. Then, $\mathbf{d}_i^t$ is the local joint value defined by the exploration strategy Φ for $0 \leq t < t^*$, and it is the the local joint value given by the global joint policy Π for the global joint state $\mathbf{s}$ for $t^* \leq t$.

5 Distributed RVI Q-Learning

Since the underlying transition probability functions of MD-DCOPs are not known to the agents a priori, there is a clear need for algorithms that perform exploration vs. exploitation tradeoffs. In this paper, we explore reinforcement learning methods to solve this problem. Specifically, we extend the (centralized) RVI Q-learning and (centralized) R-learning algorithms to solve MD-DCOPs in a distributed way.

5.1 (Centralized) RVI Q-Learning

The Relative Value Iteration (RVI) Q-learning algorithm [1] was developed to solve average-reward single-agent Markov decision processes (MDPs), but it can be tailored to solve MD-DCOPs in a centralized fashion. It is one of the more popular learning algorithms as it does not make any assumptions on the underlying model of the problem and its convergence is guaranteed for any arbitrary exploration strategy [1]. We now describe its operations using the notations defined above.

Step 1: Let time step $t = 0$. Initialize (say to 0) the Q values $Q^t(\hat{\mathbf{s}}, \hat{\mathbf{d}})$ for all global joint states $\hat{\mathbf{s}} \in \mathbf{S}$ and global joint values $\hat{\mathbf{d}} \in \mathbf{D}$.

Step 2: Let the current global joint state be $\mathbf{s}$. Choose the global joint value $\mathbf{d}$ based on some exploration strategy or converged policy. Let the immediate joint reward of choosing this value in this state be $\mathbf{F}(\mathbf{s}, \mathbf{d})$, and let the next resulting global joint state be $\mathbf{s}'$.

Step 3: Update the Q values using the following rule:

$$Q^{t+1}(\mathbf{s}, \mathbf{d}) = Q^t(\mathbf{s}, \mathbf{d}) + \gamma(t)\big[\mathbf{F}(\mathbf{s}, \mathbf{d}) + \max_{\mathbf{d}' \in \mathbf{D}} Q^t(\mathbf{s}', \mathbf{d}) - Q^t(\mathbf{s}, \mathbf{d}) - Q^t(\mathbf{s}^0, \mathbf{d}^0)\big] \tag{2}$$

where $\gamma(t)$ is a diminishing step-size schedule of stochastic approximation satisfying

$$\sum_{t=1}^{\infty} \gamma(t) = \infty \quad \text{and} \quad \sum_{t=1}^{\infty} \gamma(t)^2 < \infty \tag{3}$$

and $(\mathbf{s}^0, \mathbf{d}^0)$ is the initial state and value pair.

Step 4: Repeat Steps 2 and 3 until convergence.

5.2 Algorithm Description

While the (centralized) RVI Q-learning algorithm has been shown to be a general and powerful technique to solve average-reward (centralized) Markov decision processes (MDPs), there is no work done on how to adapt it to solve a decentralized constraint-based version of infinite-horizon average-reward MDPs, otherwise known as MD-DCOPs, in a *distributed* manner. We now describe how to do so.

We first show how the update rules in Step 3 of the (centralized) Q-learning algorithm can be done in a distributed manner. Observe that each global Q value can be decomposed into local Q values:

$$Q^t(\mathbf{s}, \mathbf{d}) \leftarrow \sum_{i=1}^{m} Q_i^t(\mathbf{s}, \mathbf{d}_i) \tag{4}$$

where each local Q value $Q_i^t(\mathbf{s}, \mathbf{d}_i)$ is associated with reward function f_i. Then, Equation 2 can be decomposed into the following for each reward function f_i:[1]

$$\begin{aligned} Q_i^{t+1}(\mathbf{s}, \mathbf{d}_i) = Q_i^t(\mathbf{s}, \mathbf{d}_i) + \gamma(t)\big[&f_i(\mathbf{s}_i, \mathbf{d}_i) \\ &+ Q_i^t(\mathbf{s}', \mathbf{d}_i' \mid \mathbf{d}_i' \in \operatorname*{argmax}_{\mathbf{d}' \in \mathbf{D}} Q^t(\mathbf{s}', \mathbf{d}')) \\ &- Q_i^t(\mathbf{s}, \mathbf{d}_i) - Q_i^t(\mathbf{s}^0, \mathbf{d}_i^0)\big] \end{aligned} \tag{5}$$

[1] We slightly abuse notations and use the symbol "∈" to mean an element of a vector instead of an element of a set.

The diminishing schedule $\gamma(t)$ is the same as the schedule in the centralized version.

There are two obstacles that prevent this equation from being updated in a completely distributed fashion. The first obstacle is that the local Q-value functions are functions of the global joint states, and each agent is only aware of the local state of each of its constraints. In our algorithm, before updating the local Q values, we require each agent to broadcast the local state of each of its constraints to all other agents. As a result, each agent can construct the global joint state.

The second obstacle is that there is one term for which a global optimization method is required. It is the term for which an argmax over the entire set of global joint values $\mathbf{D}$ is required:

$$\mathbf{d}'_i \in \underset{\mathbf{d}' \in \mathbf{D}}{\text{argmax}}\, Q^t(\mathbf{s}', \mathbf{d}') \tag{6}$$

To solve this term in a distributed manner, we map the problem into a canonical DCOP, where the utilities in the DCOP equal the local R values, and solve it using any off-the-shelf DCOP solvers. For example, in order to identify the local joint values $\mathbf{d}'_i$ in Equation 6, we set the reward $f_i(\mathbf{d}_i) = Q^t_i(\mathbf{s}', \mathbf{d}_i)$ of the canonical DCOP to equal the local Q value. Then, once the agents solve this canonical DCOP, the agents can update their Q values according to Equation 5 in a distributed fashion.

Since each Q value $Q^t_i(\mathbf{s}, \mathbf{d}_i)$ is associated with two agents, specifically the agents that are in the scope of the reward function f_i, there needs to be an agreement on which agent should maintain and update these values. We let the lower priority agent (the child or the pseudo-child in the constraint tree) maintain these values.

While this distributed RVI Q-learning algorithm can converge with any arbitrary exploration strategy, we describe a multi-arm bandit strategy. For each time step t, the agents choose their values that together maximizes the sum of the utilities over all constraints, whereby the reward of a local joint value $\mathbf{d}'_i \in \mathbf{D}_i$ for the current joint state $\mathbf{s}$ for reward function $f_i \in \mathcal{F}$ is defined by:

$$f_i(\mathbf{d}'_i) = R^t_i(\mathbf{s}, \mathbf{d}'_i) + \sqrt{2 \ln t \frac{|\mathbf{D}_i|}{n^t(\mathbf{d}'_i)}} \tag{7}$$

where $n^t(\mathbf{d}'_i)$ is the number of times the joint value $\mathbf{d}'_i$ has been chosen in the previous time steps. The problem then corresponds to finding the new value $\mathbf{d}_i$ according to:

$$\mathbf{d}_i \in \underset{\mathbf{d}' \in \mathbf{D}}{\text{argmax}} \sum_{i=1}^{m} f_i(\mathbf{d}'_i \mid \mathbf{d}'_i \in \mathbf{d}') \tag{8}$$

In order to solve this problem, we can again map this problem into a canonical DCOP, where the utilities in the DCOP are defined in Equation 7, and solve it in a distributed way. The agents continue to use this exploration strategy until convergence, after which, they use the converged strategy.

In summary, the operation of the distributed RVI Q-learning algorithm is as follows:

Step 1: For each reward function $f_i \in \mathcal{F}$, the lower priority agent in the scope of the reward function will initialize (say to 0) the current time step t, local Q values $Q^t_i(\hat{\mathbf{s}}, \hat{\mathbf{d}}_\mathbf{i})$ for all global joint states $\hat{\mathbf{s}} \in \mathbf{S}$ and local joint values $\hat{\mathbf{d}}_i \in \mathbf{D}_i$.

Step 2: Let the current local state be $\mathbf{s}_i$ and the global joint state be $\mathbf{s}$. Choose the local joint value $\mathbf{d}_i$ based on the multi-arm bandit exploration strategy described above. Then, the immediate joint reward of choosing this value in this state is $f_i(\mathbf{s}_i, \mathbf{d}_i)$, and let the next resulting local state be $\mathbf{s}'_i$.

Step 3: Broadcast the new local state $\mathbf{s}'_i$, and let the new global joint state be $\mathbf{s}'$.

Step 4: Solve the DCOP corresponding to Equation 6, and then update the local Q values according to Equation 5.

Step 5: Repeat Steps 2 through 4 until convergence.

5.3 Theoretical Properties

In this paper, we consider MDPs where a joint state can transition to any other joint state with non-zero probability, that is, the MDP is *unichain*. We are going to show the decomposability of the value function of a unichain MDP, thereby leading us to the property that the Distributed RVI Q-learning algorithm converges to an optimal solution. The detailed proofs for the lemma and theorems below are available in a technical report [23].

It is known that there always exists an optimal solution for a given unichain MDP and this solution can be characterized by the $V^*(\mathbf{s})$ value:

Theorem 1. *[28] There exists an optimal Q-value $Q^*(\mathbf{s}, \mathbf{d})$ for each joint state $\mathbf{s}$ and joint action $\mathbf{a}$ in an average-reward unichain MDP with bounded reward function satisfying:*

$$Q^*(\mathbf{s}, \mathbf{d}) + \rho^* = \mathbf{F}(\mathbf{s}, \mathbf{d}) + \sum_{\mathbf{s}'} P(\mathbf{s}', \mathbf{s}, \mathbf{d}) \max_{\mathbf{d}' \in \mathbf{D}} Q^*(\mathbf{s}', \mathbf{d}') \tag{9}$$

Additionally, there exists a unique V-value $V^*(\mathbf{s}) = \max_{\mathbf{d} \in \mathbf{D}} Q^*(\mathbf{s}, \mathbf{d})$ for each joint state $\mathbf{s}$ such that

$$V^*(\mathbf{s}) + \rho^* = \max_{\mathbf{d} \in \mathbf{D}} [\mathbf{F}(\mathbf{s}, \mathbf{d}) + \sum_{\mathbf{s}'} P(\mathbf{s}', \mathbf{s}, \mathbf{d}) V^*(\mathbf{s}')] \tag{10}$$

with $V^*(\mathbf{s}^0) = 0$ for any initial state $\mathbf{s}^0$.

To help us to prove the decomposability of the value function, we first show the decomposibility of average reward ρ^* of a given optimal policy:

Lemma 1. *For a given unichain MDP, the optimal average reward $\rho^* = \sum_{i=1}^{m} \rho_i^*$ can be decomposed into a sum of local average rewards ρ_i^* for each reward function $f_i \in \mathcal{F}$.*

From the decomposability of average reward given by Lemma 1 and the characteristic of V^* value given in Theorem 1, we now prove the decomposability of V^* as follows:

Theorem 2. *There exists $V_i^*(\mathbf{s}) = Q_i^*\big(\mathbf{s}, \mathbf{d}_i \mid \mathbf{d}_i \in \operatorname{argmax}_{\mathbf{d} \in \mathbf{D}} Q^*(\mathbf{s}, \mathbf{d})\big)$ and ρ_i for each reward function $f_i \in \mathcal{F}$ under an optimal policy $\Phi(\mathbf{s}) = \mathbf{d}_i \in \operatorname{argmax}_{\mathbf{d} \in \mathbf{D}} Q^*(\mathbf{s}, \mathbf{d})$ such that*

$$V_i^*(\mathbf{s}) + \rho_i^* = f_i(\mathbf{s}_i, \mathbf{d}_i \mid \mathbf{s}_i \in \mathbf{s}, \mathbf{d}_i \in \operatorname*{argmax}_{\mathbf{d} \in \mathbf{D}} Q^*(\mathbf{s}, \mathbf{d}))$$

$$+\sum_{\mathbf{s}'} \bar{P}_i(\mathbf{s}', \mathbf{s}, \mathbf{d}_i \mid \mathbf{d}_i \in \operatorname*{argmax}_{\mathbf{d}\in\mathbf{D}} Q^*(\mathbf{s}, \mathbf{d}))\, V_i^*(\mathbf{s}') \tag{11}$$

and $V^*(\mathbf{s}) = \sum_i V_i^*(\mathbf{s})$.

As a result of the existence of the local value $V_i^*(\mathbf{s})$, we can derive the convergence proof of our distributed RVI Q-learning algorithm:

Theorem 3. *The Distributed RVI Q-learning algorithm converges to an optimal solution.*

5.4 Complexity Analysis

Each lower priority agent that is in the scope of a reward function f_i needs to store the local Q values $Q_i^t(\mathbf{s}, \mathbf{d}_i)$ for all global joint states $\mathbf{s}$ and local joint values $\mathbf{d}_i$ for the current and next time steps t and $t+1$, respectively. Thus, the memory requirement of the agent to maintain these values is $O(|\mathbf{S}| \times |\mathbf{D}_i|)$. Therefore, the overall memory requirement per agent is $O(|\mathcal{F}| \times \max_i |\mathbf{S} \times \max_i |\mathbf{D}_i|)$.

Note that this computation does not take into account the memory required to solve the (canonical) DCOPs that correspond to solving Equations 6 and 8.

6 Distributed R-Learning

We now describe how to extend the (centralized) R-learning algorithm to solve MD-DCOPs in a distributed way.

6.1 (Centralized) R-Learning

One of the limitations of the (centralized) RVI Q-learning algorithm is that it rate of convergence can be slow in practice. As a result, researchers developed the R-learning algorithm [29, 19], which can be seen as a variant of the Q-learning algorithm. Unfortunately, the convergence of this algorithm has not been proven to the best of our knowledge. Nonetheless, it has been shown to work well in practice.

The operations of the R-learning algorithm are very similar to that of the RVI Q-learning algorithm, except that it maintains R values $R^t(\mathbf{s}, \mathbf{d})$ and average utilities ρ^t instead of Q values $Q^t(\mathbf{s}, \mathbf{d})$ for each time step t. These values are updated using the following rules:

$$\begin{aligned} R^{t+1}(\mathbf{s}, \mathbf{d}) \leftarrow R^t(\mathbf{s}, \mathbf{d})(1-\beta) + \beta\big[&\mathbf{F}(\mathbf{s}, \mathbf{d}) - \rho^t \\ &+ \max_{\mathbf{d}'\in\mathbf{D}} R^t(\mathbf{s}', \mathbf{d}')\big] \end{aligned} \tag{12}$$

$$\begin{aligned} \rho^{t+1} \leftarrow \rho^t(1-\alpha) + \alpha\big[&\mathbf{F}(\mathbf{s}, \mathbf{d}) \\ &+ \max_{\mathbf{d}'\in\mathbf{D}} R^t(\mathbf{s}', \mathbf{d}') - \max_{\mathbf{d}'\in\mathbf{D}} R^t(\mathbf{s}, \mathbf{d}')\big] \end{aligned} \tag{13}$$

where $0 \le \beta \le 1$ is the learning rate for updating the R values and $0 \le \alpha \le 1$ is the learning rate for updating the average reward ρ.

6.2 Algorithm Description

Like the global Q values, each global R value and average reward can be decomposed into local R values and local utilities:

$$R^t(\mathbf{s}, \mathbf{d}) \leftarrow \sum_{i=1}^{m} R_i^t(\mathbf{s}_i, \mathbf{d}_i) \qquad\qquad \rho^t \leftarrow \sum_{i=1}^{m} \rho_i^t \tag{14}$$

where local R value $R_i^t(\mathbf{s}_i, \mathbf{d}_i)$ and local average reward ρ_t^i are associated with reward function f_i. Then, Equations 12 and 13 can be decomposed into the following for each reward function f_i:

$$\begin{aligned} R_i^{t+1}(\mathbf{s}_i, \mathbf{d}_i) \leftarrow & R_i^t(\mathbf{s}_i, \mathbf{d}_i)(1-\beta) + \beta\big[f_i(\mathbf{s}_i, \mathbf{d}_i) - \rho_i^t \\ & + R_i^t(\mathbf{s}_i', \mathbf{d}_i' \mid \mathbf{d}_i' \in \operatorname*{argmax}_{\mathbf{d}' \in \mathbf{D}} R^t(\mathbf{s}', \mathbf{d}'))\big] \end{aligned} \tag{15}$$

$$\begin{aligned} \rho_i^{t+1} \leftarrow & \rho_i^t(1-\alpha) + \alpha\big[f_i(\mathbf{s}_i, \mathbf{d}_i) \\ & + R_i^t(\mathbf{s}_i', \mathbf{d}_i' \mid \mathbf{d}_i' \in \operatorname*{argmax}_{\mathbf{d}' \in \mathbf{D}} R^t(\mathbf{s}', \mathbf{d}')) \\ & - R_i^t(\mathbf{s}_i, \mathbf{d}_i' \mid \mathbf{d}_i' \in \operatorname*{argmax}_{\mathbf{d}' \in \mathbf{D}} R^t(\mathbf{s}, \mathbf{d}'))\big] \end{aligned} \tag{16}$$

The learning rates $0 \leq \alpha \leq 1$ and $0 \leq \beta \leq 1$ are the same as the learning rates in the centralized version.

One of the differences here is that, unlike local Q value functions, the local R value and average reward functions are functions of *local* states $\mathbf{s}_i$ instead of *global* joint states. Therefore, there is no need for agents to broadcast the local states of their constraints to the other agents in the distributed R-learning algorithm. However, there are now two terms for which a global optimization method is required. They are:

$$\mathbf{d}_i' \in \operatorname*{argmax}_{\mathbf{d}' \in \mathbf{D}} R^t(\mathbf{s}, \mathbf{d}') \qquad \mathbf{d}_i' \in \operatorname*{argmax}_{\mathbf{d}' \in \mathbf{D}} R^t(\mathbf{s}', \mathbf{d}') \tag{17}$$

Similar to distributed Q-learning, we can solve for each of these terms in a distributed manner by mapping the problems into canonical DCOPs. Once the agents solve the two canonical DCOPs that correspond to solving Equation 17, they can update their R values and average utilities according to Equations 15 and 16 in a distributed fashion. Finally, we let the agents in this algorithm use the same multi-arm bandit exploration strategy as that in distributed Q-learning.

In summary, the operation of the distributed R-learning algorithm is as follows:

Step 1: For each reward function $f_i \in \mathcal{F}$, the lower priority agent in the scope of the reward function will initialize (say to 0) the current time step t, local R values $R_i^t(\hat{\mathbf{s}}_\mathbf{i}, \hat{\mathbf{d}}_\mathbf{i})$ and local average reward ρ_i^t for all local states $\hat{\mathbf{s}}_i \in \mathbf{S}_i$ and local joint values $\hat{\mathbf{d}}_i \in \mathbf{D}_i$.

Step 2: Let the current local state be $\mathbf{s}_i$. Choose the local joint value $\mathbf{d}_i$ based on the multi-arm bandit exploration strategy described above. Then, the immediate joint reward of choosing this value in this state is $f_i(\mathbf{s}_i, \mathbf{d}_i)$, and let the next resulting local state be $\mathbf{s}_i'$.

Step 3: Solve the DCOPs corresponding to Equation 17, and then update the local R values and local average utilities according to Equations 15 and 16.

Step 4: Repeat Steps 2 and 3 until convergence.

6.3 Complexity Analysis

Using the same reasoning as that in Section 5.4, the memory requirement per agent per constraint is $O(|\mathbf{S}_i| \times |\mathbf{D}_i|)$, and the overall memory requirement per agent is $O(|\mathcal{F}| \times \max_i |\mathbf{S}_i| \times \max_i |\mathbf{D}_i|)$. Note that this computation does not take into account the memory required to solve the (canonical) DCOPs that correspond to solving Equations 17 and 8.

7 Related Work

One can view the MD-DCOP model as lying at the intersection of Stochastic DCOPs, Uncertain DCOPs, and Decentralized MDPs. We thus give a brief treatment of related work in each of these areas.

Related DCOP Models: In a *Stochastic DCOP* (S-DCOP), each constraint $f_i : D_{i_1} \times D_{i_2} \rightarrow P(x_{i_1}, x_{i_2})$ now specifies a probability distribution function of the reward as a function of the values of the variables $x_{i_1}, x_{i_2} \in X$ in the scope of the function [2, 22]. Another close extension is the DCOP under Stochastic Uncertainty model [18], where there are random variables that take on values according to known probability distribution functions. In these problems, the goal is to find a complete solution that maximizes the expected sum of all utilities. MD-DCOPs are similar to S-DCOPs in that the reward of each value combination in a constraint is stochastic. However, the stochasticity is due to the change in the underlying joint state of the problem instead of having the reward function explicitly represented as a probability distribution function.

In an *Uncertain DCOP* (U-DCOP), the probability distribution reward functions are not known a priori to the agents. Thus, the agents need to strike a balance between exploration (learning the underlying probability distribution functions) and exploitation (taking the currently believed optimal joint action). A simpler version of this problem is one where the reward of each value combination is a number instead of a distribution function [13, 32].[2] In these problems, the goal is to maximize the expected cumulative reward over the time horizon. Researchers have introduced a regret minimizing algorithm, called HEIST, that uses a multi-arm bandit strategy to solve this problem [30]. MD-DCOPs are similar to U-DCOPs in that learning is required and the algorithms need to balance exploration and exploitation. However, in MD-DCOPs, the agents need to learn the state transition functions but, in U-DCOPs, the agents need to learn the reward function.

Finally, a related dynamic DCOP model is the DCOP_MST model, where the constraint graph can change as a function of the value assignments in the previous time step [37]. In contrast, MD-DCOP constraint graph remains unchanged across all iterations. Additionally, they use local search techniques to solve their problem, while we introduce reinforcement learning techniques to solve our problem.

Related MDP Models: *Markov Decision Processes* (MDPs) have been shown to be effective models for planning under uncertainty involving a single decision maker.

[2] Researchers have called the former model MAB-DCOP and the latter model DCEE. We use the term U-DCOPs as a generalization of both models.

Researchers have thus extended these models to treat multi-agent coordination problems. One such model is the *Multi-agent MDP* [6], where the action space is factored into actions for each agent. Another model is the *Factored MDP* [7, 11], where in addition to the factored action space, the overall reward function is also factored into rewards among subgroups of agents. There is also the *Decentralized MDP* (Dec-MDP) [5], where the global state is also factored into local states for each agent. Lastly, a *Transition-Independent Dec-MDP* (TI-Dec-MDP) [4] is a subclass of Dec-MDPs, where the transition from a local state to the next local state is independent of the local states and actions of all other agents.

The MD-DCOP model can be viewed as part of this family of algorithms. Like all the above algorithms, it too assumes a factored action space. Like factored MDPs, it too assumes that rewards are factored into agent subgroups, where each agent subgroup is the scope of agents in a constraint. Like Dec-MDPs, it too assumes that the global state is factored, but the state is factored into local states for each constraint instead of each agent. Like TI-Dec-MDPs, it too assumes that the transition from a local state to the next is independent of the local states. However, unlike TI-Dec-MDPs, MD-DCOPs assume a different general form of transition independence – the transition of a local state of a constraint to the next local state is independent of the local states of other constraints but is dependent on the joint actions of the agents in the scope of the constraint associated with that local state. We call this assumption *constraint-based transition independence*.

While there is a large body of work on applying coordinated reinforcement learning on Factored MDPs [10, 12, 14] and Dec-MDPs [35, 36], they typically optimize the expected cumulative reward over a finite horizon or the expected cumulative *discounted* reward over an infinite horizon. In contrast, we optimize the *average* expected reward over an infinite horizon. We believe that for infinite horizon problems, optimizing the average expected reward fits more naturally with many multi-agent applications, where achieving a goal in a future time step is as important as achieving it in the current time step. For example, in our sensor network application, detecting an intruder next week should not be any less valuable than detecting an intruder today. In some cases, optimizing the average expected reward is harder and the motivation for using discounted utilities is mostly a convenience, not a better match with the real objective. It is well known that discount factors are often selected arbitrarily despite the fact that they could affect the final policy. These considerations underscore the importance of developing methods for optimizing the average expected reward.

To the best of our knowledge, the only work that optimizes the average expected reward in this family of problems is [27]. The main differences are that the assumption of transition independence is different for the two models, as described above; they use a centralized approach (a mixed integer linear program) while we use a distributed approach; and they assume that the underlying joint distribution is known while we do not.

8 Experimental Results

In addition to the distributed Q- and R-learning algorithms described above, we also implemented another version of the distributed Q-learning algorithm, where we maintain a Q-value $Q_i(\mathbf{s}_i, \mathbf{d_i})$ for each local state $\mathbf{s}_i \in \mathbf{S}_i$ instead of each global state $\mathbf{s} \in \mathbf{S}$.

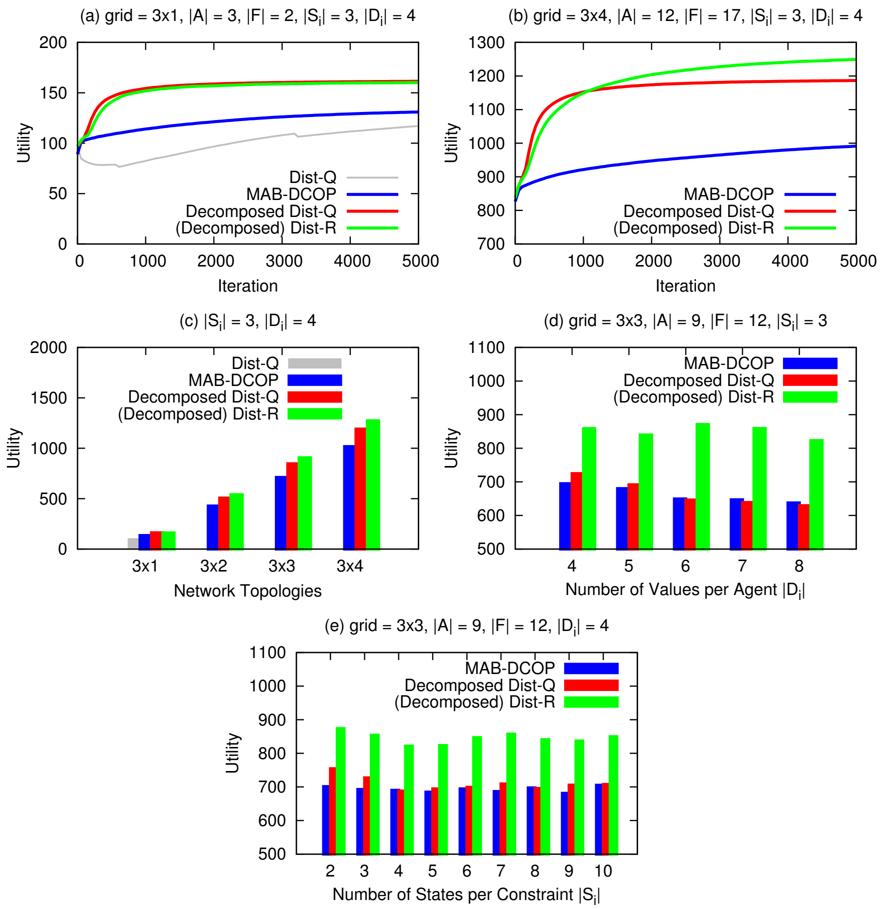

Fig. 2. Experimental Results

The update rule is thus:

$$
\begin{aligned}
Q_i^{t+1}(\mathbf{s}_i, \mathbf{d}_i) = Q_i^t(\mathbf{s}_i, \mathbf{d}_i) + \gamma(t)\big[& f_i(\mathbf{s}_i, \mathbf{d}_i) \\
&+ Q_i^t(\mathbf{s}_i', \mathbf{d}_i' \mid \mathbf{d}_i' \in \operatorname*{argmax}_{\mathbf{d}' \in \mathbf{D}} Q^t(\mathbf{s}_i', \mathbf{d}')) \\
&- Q_i^t(\mathbf{s}, \mathbf{d}_i) - Q_i^t(\mathbf{s}^0, \mathbf{d}_i^0)\big] \qquad (18)
\end{aligned}
$$

We call this version the Decomposed Distributed Q-learning algorithm. Like the distributed R-learning algorithm, this algorithm does not have any convergence guarantees.

We compare the above three algorithms to Multi-Arm Bandit DCOP (MAB-DCOP) [30], a regret-minimizing algorithm that seeks to maximize the expected cumulative reward over the time horizon in a DCOP with reward uncertainty. We run our experiments on a 64 core Linux machine with 2GB of memory and evaluate the algorithms on our motivating sensor network domain.

We vary the number of agents $|\mathcal{A}|$ and the number of constraints $|\mathcal{F}|$ by varying the topology of the sensor network. We used a 4-connected grid topology, where each sensor has a constraint with each sensor in its four cardinal directions. We fixed the number of rows in the grid to 3 and varied the number of columns from 1 to 4. We also varied the number of values per agent $|D_i|$ of each agent a_i from 4 to 8 and the number of local states per constraint $|\mathbf{S}_i|$ from 2 to 10.

We make the following observations about the results shown in Figure 2:

- Figures 2(a–b) show the convergence rate of the four algorithms. In the bigger problem, the Distributed Q algorithm ran out of memory due to the large number of global joint states. In the smaller problem, it finds worse solutions than the other algorithms, but that is likely due to its slow convergence rate as it is guaranteed to converge to the global optimum. The results also show that both decomposed versions of the Q- and R-learning algorithms performed better than MAB-DPOP, which is not surprising as MAB-DPOP was not designed to solve MD-DCOPs and thus does not exploit the assumption that the underlying transitions are Markovian.
- Figures 2(c–e) show the final solution quality of the algorithms.We omitted Distributed Q in the latter two figures as it ran out of memory. In general, MAB-DPOP and the Decomposed Distributed Q-learning algorithm performed equally well and the Distributed R-learning algorithm performed the best. These results thus affirm that the conclusions found for the centralized versions of the reinforcement learning algorithms, where the centralized R-learning algorithm was found to perform better than the centralized Q-learning algorithm [19], also hold for the distributed case.

9 Conclusions

The Dynamic DCOP formulation is attractive as it is able to model various multi-agent coordination problems that dynamically change over time. Unfortunately, existing work typically assumes that the problem at each time step is independent of the problems in the other time steps. This assumption does not hold in problems like our motivating sensor network domain, where the choice of value assignments (i.e., actions by the agents) in one time step can affect the problem in the next time step. In this paper, we take the first step towards capturing this inter-dependence, by introducing the Markovian Dynamic DCOP (MD-DCOP) model, where the underlying transition functions are Markovian. We also introduce several distributed reinforcement algorithms to solve this problem, and show that they outperform MAB-DPOP, a regret-minimizing multi-arm bandit algorithm, for a range of sensor network problems.

Acknowledgment

This research is partially supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office and by the National Science Foundation under grant number IIS-1116917.

Bibliography

[1] J. Abounadi, D. Bertsekas, and V. Borkar. Learning algorithms for markov decision processes with average cost. *SIAM Journal on Control and Optimization*, 40(3):681–698, 2001.

[2] J. Atlas and K. Decker. Coordination for uncertain outcomes using distributed neighbor exchange. In *Proceedings of AAMAS*, pages 1047–1054, 2010.

[3] F. Bacchus, X. Chen, P. van Beek, and T. Walsh. Binary vs. non-binary constraints. *Artificial Intelligence*, 140(1–2):1–37, 2002.

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

[5] D. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.

[6] C. Boutilier. Sequential optimality and coordination in multiagent systems. In *Proceedings of IJCAI*, pages 478–485, 1999.

[7] C. Boutilier, R. Dearden, and M. Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121(1–2):49–107, 2000.

[8] D. Burke and K. Brown. Efficiently handling complex local problems in distributed constraint optimisation. In *Proceedings of ECAI*, pages 701–702, 2006.

[9] A. Farinelli, A. Rogers, A. Petcu, and N. Jennings. Decentralised coordination of low-power embedded devices using the Max-Sum algorithm. In *Proceedings of AAMAS*, pages 639–646, 2008.

[10] C. Guestrin, D. Koller, and R. Parr. Multiagent planning with factored MDPs. In *Proceedings of NIPS*, pages 1523–1530, 2001.

[11] C. Guestrin, D. Koller, R. Parr, and S. Venkataraman. Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research*, 19:399–468, 2003.

[12] C. Guestrin, M. Lagoudakis, and R. Parr. Coordinated reinforcement learning. In *Proceedings of ICML*, pages 227–234, 2002.

[13] M. Jain, M. Taylor, M. Tambe, and M. Yokoo. DCOPs meet the real world: Exploring unknown reward matrices with applications to mobile sensor networks. In *Proceedings of IJCAI*, pages 181–186, 2009.

[14] J. Kok and N. Vlassis. Collaborative multiagent reinforcement learning by payoff propagation. *Journal of Machine Learning Research*, 7:1789–1828, 2006.

[15] R. Lass, J. Kopena, E. Sultanik, D. Nguyen, C. Dugan, P. Modi, and W. Regli. Coordination of first responders under communication and resource constraints (Short Paper). In *Proceedings of AAMAS*, pages 1409–1413, 2008.

[16] R. Lass, E. Sultanik, and W. Regli. Dynamic distributed constraint reasoning. In *Proceedings of AAAI*, pages 1466–1469, 2008.

[17] T. Léauté and B. Faltings. Coordinating logistics operations with privacy guarantees. In *Proceedings of IJCAI*, pages 2482–2487, 2011.

[18] T. Léauté and B. Faltings. Distributed constraint optimization under stochastic uncertainty. In *Proceedings of AAAI*, pages 68–73, 2011.

[19] S. Mahadevan. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine Learning*, 22(1–3):159–195, 1996.

[20] R. Maheswaran, M. Tambe, E. Bowring, J. Pearce, and P. Varakantham. Taking DCOP to the real world: Efficient complete solutions for distributed event scheduling. In *Proceedings of AAMAS*, pages 310–317, 2004.

[21] P. Modi, W.-M. Shen, M. Tambe, and M. Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1–2):149–180, 2005.

[22] D. T. Nguyen, W. Yeoh, and H. C. Lau. Stochastic dominance in stochastic DCOPs for risk-sensitive applications. In *Proceedings of AAMAS*, pages 257–264, 2012.

[23] D. T. Nguyen, W. Yeoh, H. C. Lau, S. Zilberstein, and C. Zhang. Decentralized multi-agent reinforcement learning in average-reward dynamic DCOPs (theoretical proofs). Technical Report TR-CS-NMSU-2014-2-10, Department of Computer Science, New Mexico State University, 2014.

[24] A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *Proceedings of IJCAI*, pages 1413–1420, 2005.

[25] A. Petcu and B. Faltings. Superstabilizing, fault-containing multiagent combinatorial optimization. In *Proceedings of AAAI*, pages 449–454, 2005.

[26] A. Petcu and B. Faltings. Optimal solution stability in dynamic, distributed constraint optimization. In *Proceedings of IAT*, pages 321–327, 2007.

[27] M. Petrik and S. Zilberstein. Average-reward decentralized Markov decision processes. In *Proceedings of IJCAI*, pages 1997–2002, 2007.

[28] M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 2005.

[29] A. Schwartz. A reinforcement learning method for maximizing undiscounted rewards. In *Proceedings of ICML*, pages 298–305, 1993.

[30] R. Stranders, F. Delle Fave, A. Rogers, and N. Jennings. DCOPs and bandits: Exploration and exploitation in decentralised coordination. In *Proceedings of AAMAS*, pages 289–297, 2012.

[31] E. Sultanik, R. Lass, and W. Regli. Dynamic configuration of agent organizations. In *Proceedings of IJCAI*, pages 305–311, 2009.

[32] M. Taylor, M. Jain, Y. Jin, M. Yokoo, and M. Tambe. When should there be a "me" in "team"?: Distributed multi-agent optimization under uncertainty. In *Proceedings of AAMAS*, pages 109–116, 2010.

[33] W. Yeoh and M. Yokoo. Distributed problem solving. *AI Magazine*, 33(3):53–65, 2012.

[34] M. Yokoo, editor. *Distributed Constraint Satisfaction: Foundation of Cooperation in Multi-agent Systems*. Springer, 2001.

[35] C. Zhang and V. Lesser. Coordinated multi-agent reinforcement learning in networked distributed POMDPs. In *Proceedings of AAAI*, pages 764–770, 2011.

[36] C. Zhang and V. Lesser. Coordinating multi-agent reinforcement learning with limited communication. In *Proceedings of AAMAS*, pages 1101–1108, 2013.

[37] R. Zivan, R. Glinton, and K. Sycara. Distributed constraint optimization for large teams of mobile sensing agents. In *Proceedings of IAT*, pages 347–354, 2009.