Coordinating Multi-Agent Reinforcement Learning with Limited Communication

Chongjie Zhang, and Victor Lesser
University of Massachusetts Amherst
Amherst, MA, US
{chongjie,lesser}@cs.umass.edu

ABSTRACT

Coordinated multi-agent reinforcement learning (MARL) provides a promising approach to scaling learning in large cooperative multi-agent systems. Distributed constraint optimization (DCOP) techniques have been used to coordinate action selection among agents during both the learning phase and the policy execution phase (if learning is off-line) to ensure good overall system performance. However, running DCOP algorithms for each action selection through the whole system results in significant communication among agents, which is not practical for most applications with limited communication bandwidth. In this paper, we develop a learning approach that generalizes previous coordinated MARL approaches that use DCOP algorithms and enables MARL to be conducted over a spectrum from independent learning (without communication) to fully coordinated learning depending on agents' communication bandwidth. Our approach defines an interaction measure that allows agents to dynamically identify their beneficial coordination set (i.e., whom to coordinate with) in different situations and to trade off its performance and communication cost. By limiting their coordination set, agents dynamically decompose the coordination network in a distributed way, resulting in dramatically reduced communication for DCOP algorithms without significantly affecting overall learning performance. Essentially, our learning approach conducts co-adaptation of agents' policy learning and coordination set identification, which outperforms approaches that sequence them.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems, Coherence and coordination*

General Terms

Performance, Design, Algorithms, Experimentation

Keywords

Multiagent learning, coordinated learning, distributed constraint optimization

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013),* Ito, Jonker, Gini, and Shehory (eds.), May 6–10, 2013, Saint Paul, Minnesota, USA.

1. INTRODUCTION

Cooperative multi-agent systems are finding applications in a wide variety of practical domains, including robotics, sensor networks, distributed control, collaborative decision support systems, supply chains, and data mining. A cooperative multi-agent system (MAS) is composed of a set of autonomous agents that interact with one another in a shared environment in order to reach a shared goal or optimize a global performance measure. A central challenge in cooperative MAS research is to design distributed decision policies for agents to coordinate their actions. Multi-agent reinforcement learning (MARL) provides an attractive approach for agents to developing effective coordination policies without explicitly building a complete decision model. MARL allows agents to explore environment through trial and error, adapt their behaviors to the dynamics of the uncertain and evolving environment, and gradually improve their performance through experiences.

One of key research challenges is to scale MARL to large cooperative systems. One promising direction to address this challenge is to coordinate multi-agent learning that exploits both interaction locality and non-local information. Agents learn their policies based on their local observations and interactions, while their learning processes are coordinated to ensure the overall system performance. The Multi-Agent Supervisory Policy Adaptation (MASPA) approach [14] uses an emergent supervisory organization with low overhead that exploits non-local information to dynamically coordinate and shape the learning processes of individual agents while still leaving agents to react autonomously to local feedbacks. The original MASPA paper used some domain knowledge to generate supervisory information to coordinate MARL and did not provide solutions for automating supervision without domain knowledge. Coordinated multi-agent learning approaches [6, 8, 16, 3] exploit distributed constraint optimization (DCOP) techniques to coordinate action selection during the learning, which does not require domain knowledge. However, running DCOP algorithms for each action selection through the whole system results in intensive computation and significant communication among agents, which is not practical for most applications with limited computational resources and communication bandwidth.

In this paper, we develop a learning approach that generalizes previous coordinated MARL techniques [6, 8, 16] and enables MARL to be conducted over a spectrum from independent learning (without communication) to fully coordinated learning depending on agents' communication bandwidth. In many practical applications, although agents may interact with many others over time, they usually only need to coordinate with very few (e.g., one or two) agents that strongly affect their performance at most decision-making points. To exploit this interaction property, we defines an interaction measure that allows agents to dynamically estimate the potential util-

ity loss for lack of coordination with any subgroup of agents. Using this estimation, agents can compute their beneficial coordination set (i.e., containing agents whom to coordinate with) in different situations and make the best use of their limited communication bandwidth to coordinate their learning processes. By limiting the maximum potential loss for lack of coordination, our approach prompts agents to minimize their coordination set. This minimization will dramatically reduce the number of links on the agent coordination network and often decompose it into smaller disjointed sub networks, which results in dramatically reducing communication and computation of DCOP algorithms without significantly affecting overall learning performance. Empirical results confirm that our learning approach can effectively trade off the overall learning performance and communication cost for coordinating agents' learning processes. Essentially, our learning approach co-adapts two interacting activities, i.e., agents' policy learning (or value function learning) and coordination set identification, that is, agents concurrently learn their policies and their coordination sets for different situations. Experiments show that this co-adaptation outperforms approaches that sequence them.

Note that the self-organization approach for coordinating MARL in [15] is also a co-adaptation approach, which allows agents to dynamically evolving supervisory organizations for MASPA [14] to better coordinate their learning processes while they are concurrently learning their operational policies. This self-organization is intended to form a nearly decomposable hierarchical organization structure to improve the performance of coordinating agents' learning processes, while, in contrast, our approach attempts to minimize the communication cost for coordinating agents' learning processes using DCOP algorithms without significantly affecting the overall learning performance. In addition, the work [15] defines an interaction measure that captures how important interactions among a group of agents are to the system performance, but this measure may not help to decide whether to coordinate their learning processes. ~~For example, if this group of agents have already implicitly been optimally coordinated through their local learning~~, then they do not need additional coordination. In contrast, the interaction measure defined in this paper quantifies the potential loss of an agent lack of coordination with any subgroup of agents, which allows agents to identify which subgroups of agents they should coordinate with in order to improve their learning performance.

2. COOPERATIVE MULTI-AGENT DECISION-MAKING MODEL

A cooperative multi agent sequential decision-making problems with N agents can be modeled as a Dec-POMDP ~~[2].~~ In this paper, we consider Dec-POMDPs with factored states and factored rewards, which is a general case over a number of variants of Dec-POMDPs, including Transition-Independent Dec-MDPs [1], Network-Distributed POMDPs (ND-POMDPs) [12], Transition-Decoupled POMDPs [13], and collaborative multi-agent MDPs [7].

Suppose we have n agents, indexed by $\{1, \ldots, n\}$. Each agent i must choose an action a_i from a finite set of actions denoted by A_i. The joint state $s \in S$ can be factored as $s = \langle s_u, s_1, \ldots, s_n \rangle$, where s_u refers to the state of the external environment and s_i is the local state of agent i. The environment is modeled by stationary state transition probabilities $P(s'|s, a)$ with an initial state distribution $P(s_0)$, where $a = \langle a_1, \ldots, a_n \rangle$ is the joint action performed in joint state s resulting in joint state s'. After joint action a are executed in joint state s, each agent i receive observation ω_i governed by observation probability model $O(\omega_1, \ldots, \omega_n|s, a)$. An agent interacts with different subgroups of agents. The joint reward function, $R(s, a) = \sum_i \sum_{g_i} R_{g_i}(s_i, s_{g_i}, s_u, a_i, a_{g_i})$, is decomposable among agents referred by i. The reward of agent i is a sum of rewards it receives from subgroups of agents whom it interacts with, referred by g_i. If $k = |g_i|$ agents $i_1, \ldots, i_k$ are involved in a particular subgroup g_i, then s_{g_i} denotes the state of group g_i, i.e., $s_{g_i} = \langle s_{i1}, \ldots, s_{ik} \rangle$. Similarly, $a_{g_i} = \langle a_{i1}, \ldots, a_{ik} \rangle$.

Figure 1: A 4-chain sensor configuration

Based on the reward function, we can denote the set of subgroups of agents that agent i interacts with by $G_i = \{g_i \in I \mid g_i$ forms the reward component R_{g_i} for agent $i\}$. We denote the set of agents that interact with agent i by $I_i = \cup_{g_i \in G_i} g_i$. Although we define i through the reward structures, in this paper, we assume that I_i specifies all agents interacting with agent i either through its reward function, state transition function, or observation function.

The goal for agents is to compute joint policy π that maximizes the total expected utility of all agents. Without communication, agents can only act based on their local observations. In this case, joint policy π is defined by $\langle \pi_1, \ldots, \pi_n \rangle$, where π_i refers to the individual policy of agent i that maps its history of observations to an action $a_i \in A_i$. If communication is allowed, joint policy π can also be defined by one policy, called the *global policy*, that maps from a history of joint observations to an joint action $a \in A$. This is because agents can exchange their observations and select actions based on joint observations. Obviously, the optimal global policy inherently performs better than the optimal set of individual policies. ~~In this paper, we assume agents can communicate (at least with their neighbors) and their action selections can be coordinated through communication during learning.~~

3. BASIC LEARNING APPROACHES

To learn the joint policy, we need to define a Q-function (or Q-value function). Let Q-function $Q(\vec{h}, a)$ represent the expected reward of doing joint action a with history $\vec{h}$ of joint observations and actions and behaving optimally from then on. The globally joint policy π can be derived from $Q(\vec{h}, a)$ by setting $\pi(\vec{h}) = \text{argmax}_{a \in A} Q(\vec{h}, a)$.

In principle, we can directly estimate $Q(\vec{h}, a)$ by using standard single-agent Q-learning:

$$Q(\vec{h}^t, a^t) = (1 - \alpha) Q(\vec{h}^t, a^t) + \alpha [r^t + \gamma \max_a Q(\vec{h}^{t+1}, a)] \quad (1)$$

where $\alpha \in (0, 1)$ is the learning rate, r^t is the immediate reward of doing a^t for observation history $\vec{h}^t$, $\gamma \in [0, 1]$ is the discount factor, which is usually set to 1 for a finite horizon. We call this approach *globally joint learning*. Although this approach leads to an optimal policy, it is practically intractable, because the policy space is exponential in the number of agents and the agents might not have access to the needed information (i.e., observations, actions, and rewards of all other agents) for learning and selecting actions.

At the other extreme, we have the *independent learning* approach [4] in which agents ignore the actions and rewards of the other agents, and concurrently learn their own action-value functions solely based on their local observations and rewards. To provide local rewards in ND-POMDPs, we can split the reward component R_l evenly among agents in group l. This approach is distributed, re-

sults in big storage and computational savings in the policy space, and does not require communication during learning and execution. However, this approach lacks coordination and might lead to oscillations or converge to local optimal policies. For example, Figure 1 shows a specific target tracking problem consisting of four sensors. Each sensor node can scan in one of four directions: North, South, East or West. To track a target and obtain the associated reward, two sensors with overlapping scanning areas must coordinate by scanning the same area simultaneously. In Figure 1, if location1, location2, and location3 always have targets with sensing reward 40, 60, and 40, respectively, then, by using the independent learning approach, sensor2 and sensor3 will potentially learn to sense location2, which is locally optimal with average expected reward 60. However, the optimal policy is that sensor1 and sensor2 always sense location1 and sensor3 and sensor4 always sense location3, whose global expected reward is 100. Therefore, some form of coordination is needed in order to learn the globally optimal policy.

4. COORDINATED LEARNING WITH CONTROLLED COMMUNICATION

As discussed in the previous section, directly learning the globally joint policy in a centralized way is infeasible from a practical perspective, while independent learning is a distributed, scalable approach, but may yield poor global performance. In many practical applications, an agent has to coordinate its actions with a few agents only, and acts independently with respect to the other agents. ~~As with previous techniques of coordinated multi-agent reinforcement learning [6, 8, 16], our approach presented in this section will take advantage of this structured interactions~~, which achieves scalability by allowing agents to learn based on its local observations and local reward signals and ensure the global learning performance by using DCOP techniques to coordinate distributed learning processes. Our approach generalizes techniques in [6, 8, 16] and allows agents to learn to dynamically identify whom to coordinate with to minimize its communication while concurrently learning their operational policies.

In general, the global expected utility $Q(\vec{h}, a)$ depends on depends on joint observation histories and joint actions. However, in many practical problems, it is possible to approximate the global expected utility Q by the sum of local sub-utilities of agents. In this paper, we use approximation to learn local utilities, such that $\hat{Q}(\vec{h}, a)$ (defined as following) is a good approximation of the optimal total expected utility $Q^*(\vec{h}, a)$.

$$\hat{Q}(\vec{h}, a) = \sum_{i \in I} Q_i(\vec{h_i}, a_i, a_{I_i}), \tag{2}$$

where Q_i is the local utility of agent i, which is defined on its local observation history, local action, and actions of agents I_i that interact with agent i.

The complexity of learning local utility Q_i of agent i is exponential with the number of agents I_i that interact with agent i, which may still be relatively large in large-scale systems. In practical systems, not all agents in the I_i interact with each other. For example, in the target tracking problem shown in Figure 1, both sensor1 and sensor3 interact with sensor2, but sensor1 does not directly interact with sensor3. As described in the model (presented in Section 2), an agent may interact with a number of smaller subgroups of agents. Using the reward decomposition structure, we further assume that local utility Q_i can be approximated by a sum of utilities over its interacting groups G_i. Now, we have the local utility of agent i

$$Q_i(\vec{h_i}, a_i, a_{Ii}) = \sum_{g_i \in G_i} Q_{g_i}(\vec{h_i}, a_i, a_{g_i}), \tag{3}$$

where $Q_{g_i}(\vec{h_i}, a_i, a_{g_i})$ is the local utility of agent i on group g_i, which is defined on its local observation history, local action, and actions of agents in group g_i. Normally, the size of g_i is much less than that of I_i, which makes the learning more tractable.

The total expected utility now becomes

$$\hat{Q}(\vec{h}, a) = \sum_{i \in I} \sum_{g_i \in G_i} Q_{g_i}(\vec{h_i}, a_i, a_{g_i}).$$

We will use Q-learning to learn $\hat{Q}(\vec{h}, a)$. The global Q-learning update rule shown in equation (1) can be rewritten as

$$\begin{aligned} &\textstyle\sum_{i \in I} \sum_{g_i \in G_i} Q_{g_i}(\vec{h_i}^t, a_i^t, a_{g_i}^t) = \\ &\quad (1-\alpha) \textstyle\sum_{i \in I} \sum_{g_i \in G_i} Q_{g_i}(\vec{h_i}^t, a_i^t, a_{g_i}^t) \\ &\quad + \quad \alpha[\textstyle\sum_{i \in I} \sum_{g_i \in G_i} r_{g_i}^t + \gamma \max_a \hat{Q}(\vec{h}^{t+1}, a)], \end{aligned} \tag{4}$$

where r_{g_i} is the reward agent i receives when interacting with agents in subgroup g_i.

Note that the discounted future reward, $\max_a \hat{Q}(\vec{h}^{t+1}, a)$, can not be directly written as the sum of local discounted future rewards, because it depends on the joint action that maximizes the global value. Fortunately, we can accomplish this by defining the joint action $a^* = \text{argmax}_a \hat{Q}(\vec{h}^{t+1}, a)$ and $\max_a \hat{Q}(\vec{h}^{t+1}, a) = \hat{Q}(\vec{h}^{t+1}, a^*) = \sum_{i \in I} \sum_{g_i \in G_i} Q_{g_i}(\vec{h_i}^{t+1}, a_i^*, a_{g_i}^*)$. We are now able to decompose all terms in (4) and write the update rule for each agent i interacting with agent group g_i:

$$\begin{aligned} Q_{g_i}(\vec{h_i}^t, a_i^t, a_{g_i}^t) &= (1-\alpha) Q_{g_i}(\vec{h_i}^t, a_i^t, a_{g_i}^t) \\ &+ \quad \alpha[r_{g_i}^t + \gamma Q_{g_i}(\vec{h_i}^{t+1}, a_i^*, a_{g_i}^*)] \end{aligned} \tag{5}$$

Similar to those in [8, 16], the update rule in (5) is now based on local observations, local reward, and local Q-function, except for a_i^* and $a_{g_i}^*$. This update rule is quite different from coordinated reinforcement learning approach in [6], where local Q-function update depends ~~on the global reward signal and the global Q-value~~, which are not usually specifically tailored to local behaviors, thus resulting in slower learning convergence. ~~Note that the local contribution $Q_{g_i}(\vec{h_i}^{t+1}, a_i^*, a_{g_i}^*)$ of group g_i to the global action value might be lower than $\max_{a_i, a_{g_i}} Q_{g_i}(\vec{h_i}^{t+1}, a_i, a_{g_i})$, the maximizing value of its local Q-function, because it is unaware of the dependencies among groups.~~

~~Computing the joint action a^* that maximizes $\hat{Q} = \sum_i \sum_{g_i} Q_{g_i}$~~ seems intractable a prior, as it would require the enumeration of the joint action space of all agents. Fortunately, by exploiting the *interaction locality* shown in Q_{g_i}-functions, computing the optimal action can be potentially efficient through message-passing DCOP algorithms, as proposed in [6, 8, 16]. However, as we need to compute the coordinated joint action at every Q-function update during the learning and, if using offline learning, at every action selection during the policy execution after learning, running DCOP algorithms through the whole systems can be inefficient or even infeasible when communication is very limited and real-time response is critical. In practical applications, an agent can interact with a lot of other agents over time, but it may only interact with very few (e..g, one or two) agents at a particular situation and with different agents at different situations. In the following sections, we will present an approach that allows agents to learn to dynamically identify who to coordinate at different situations and to trade off

between the quality of coordinated learning and communication. We will first describe the coordinated learning process of agents, then discuss how an agent identifies who to coordinate with given a learning quality control parameter, and finally show how to compute the coordinated action selection.

4.1 Learning Processes with Emergent Coordination

Using update rule in (5), our approach distributes the learning of the global function $\hat{Q}$ among agents. Each agent i currently learns and maintains local utility functions Q_{g_i} for all $g_i \in G_i$. Algorithm 1 shows the coordinated learning process of agent i. The loss rate threshold $\xi \in [0, 1]$ is used to compute the coordination set, which contains agents who agent i will coordinate with for action selection. It is the key parameter to minimize the communication while ensuring the quality of coordinated action selection. Each agent i need to incrementally estimate conditional probability distribution $P_{g_i}(a_{g_i}|\vec{h}_i, a_i)$ for all $g_i \in G_i$ based on its execution history and observation of other agents' actions. We assume that every agent i can observe actions executed by agents in i_i (i.e., agents interacting with agent i). These conditional probability distributions will be used to fully or partially marginalize the expected reward $Q_{g_i}(\vec{h}_i, a_i, a_{g_i})$ through actions of agents that agent i does not coordinate with. For example, if agent i does not coordinate with any other agents, that is, $CS = \emptyset$, it needs to compute its expected reward $Q_i(\vec{h}_i, a_i)$ defined its local observations and actions as following:

$$Q_i(\vec{h}_i, a_i) = \sum_{g_i \in G_i} Q_{g_i}(\vec{h}_i, a_i, a_{g_i}) P(a_{g_i}|\vec{h}_i, a_i),$$

so that it can locally select action $a_i^* = \text{argmax}_{a_i} Q_i(\vec{h}_i, a_i)$ based on the current observation $\vec{h}_i$. In summary, these estimated joint probability distributions will be used in both computing the coordination set and selecting the coordinated action, which will be discussed in detail in the following sections.

Note that the coordination set may change during the process of coordinating action selection. For example, the initial computed coordination set of agent i does not contain agent j, but agent j's coordination set contains agent i. During the action selection process by running a DCOP algorithm, agent i will add agent j to its coordination set by receiving DCOP messages from agent j.

The coordinated learning process is essentially a co-evolution of an agent's policy (i.e., the expected utility function) and its coordination set at different situations. This is because the computation of the coordination set of an agent relies on its local utility functions $Q_{g_i \in G_i}$, while changing the coordination set will change the agent's learning environment, which will affect the learning of its local utility functions. From the system-level view, our coordinated multi-agent reinforcement learning co-adapts agents' policies and their coordination network.

After being learned, the global expected reward function $\hat{Q}$ is distributedly represented by local utility functions, Q_{g_i}. As a result, during the execution phase (if agents learn offline), agents' action selections are computed online in a distributed manner by a DCOP algorithm from local Q-functions, which is similar to the process described in Algorithm1. Note that local Q-function Q_{g_i} is defined on the observation history of agent i, which scales exponentially with the horizon. To deal with a large or infinite horizon, one approach is to use a fixed-size window of observations, as we did in our experiments. Other more sophisticated approaches (i.e., utile suffix memory [10]) for dealing with partial observability can also be used with our approach.

Algorithm 1: The coordinated learning process of agent i

```
Let α be the learning rates, γ be the discount factor, ξ be the loss rate threshold;
h_i ← ∅ ;
Initialize utility functions Q_{g_i} for all g_i ∈ G_i ;
Initialize estimated conditional probability distribution P_{g_i}(a_{g_i}|h_i, a_i) for all g_i ∈ G_i ;
CS ← computeCoordinationSet(i, Q_{g_i∈G_i}, P_{g_i∈G_i}, h_i, ξ) ;
a_i^* ← select the coordinated action by running DCOP algorithm through agents in CS ;
repeat
    Execute a_i^* or an action selected under suitable exploration on observation history h_i ;
    Receive observation ω_i and rewards {r_{g_i}|g_i ∈ G_i} for its interaction subgroups ;
    Observe actions of other agents that interact with agent i ;
    Update observation history h_i with ω_i ;
    CS ← computeCoordinationSet(i, Q_{g_i∈G_i}, P_{g_i∈G_i}, h_i, ξ) ;
    a_i^* ← select the coordinated action by running DCOP algorithm through agents in CS ;
    foreach subgroup g_i ∈ G_i do
        Update Q_{g_i} using rule (5) ;
        Update estimated conditional probability distribution P_{g_i}(a_{g_i}|h_i, a_i);
    end
until the process is terminated;
```

4.2 Identifying Coordination Set

To identify the coordination set for each agent, we define an agent interaction measure, called *potential loss in lack of coordination*, to quantity how much utility an agent will potentially lose if it does not coordinate with a subgroup of agents. This interaction measure is built upon a measure, called *potential expected utility*, which identifies the maximum expected utility of an agent can potentially receive when it solely coordinates with a subgroup of agents.

DEFINITION 1. *The* **potential expected utility** $PV_i(\vec{h}_i, a_i, C)$ *of agent i exclusively coordinating with a subgroup of agents $C \subseteq I_i$ and selecting action a_i on observation history $\vec{h}$ is defined as*

$$PV_i(\vec{h}_i, a_i, C) = \sum_{g_i \in G_i} \max_{a_C} Q_{g_i}(\vec{h}_i, a_i, a_C), \qquad (6)$$

where

$$Q_{g_i}(\vec{h}_i, a_i, a_C) = \sum_{a_{g_i \setminus C}} Q_{g_i}(\vec{h}_i, a_i, a_{g_i}) P_{g_i}(a_{g_i \setminus C}|\vec{h}_i, a_i, a_{g_i \cap C}).$$

Here I_i is the set of all agents that interacts with agent i. Note that $P_{g_i}(a_{g_i \setminus C}|\vec{h}_i, a_i, a_{g_i \cap C})$ can be computed from our estimated conditional probability distribution $P_{g_i}(a_{g_i}|\vec{h}_i, a_i)$. According to the definition, we can see that the potential expected utility PV_i optimistically assume the behaviors of subgroup of agents C, which unconditionally cooperate with agent i's action a_i. Therefore, in general, this measure is an overestimate of the expected utility that an agent can get if it coordinates with a subgroup of agents. The potential expected utility measure has the following property, that is, coordinating with additional agents will not decrease the potential expected utility.

Algorithm 2: computeCoordinationSet$(i, Q_{g_i \in G_i}, P_{g_i \in G_i}, \vec{h}_i, \xi)$

1 $maxLoss = \xi * \max\{|\max_{a_i} PV_i(\vec{h}_i, a_i, I_i)|,$
$|\max_{a_i} PV_i(\vec{h}_i, a_i, I_i)|, PLILOC_i(\vec{h}_i, \emptyset)\}$;

2 Find $NC \subset I_i$, such that

1. $PLILOC_i(\vec{h}_i, NC) \leq maxLoss$,
2. $PLILOC_i(\vec{h}_i, NC) \leq PLILOC_i(\vec{h}_i, D)$, for all $D \subset I_i$ and $|D| = |NC|$;
3. $PLILOC_i(\vec{h}_i, D) > maxLoss$, for all $D \subset I_i$ and $|D| > |NC|$;

Return $I_i \setminus NC$;

PROPOSITION 1. *If $D \subseteq C \subseteq I_i$, then $PV_i(\vec{h}_i, a_i, D) \leq PV_i(\vec{h}_i, a_i, C)$ for all $\vec{h}_i$ and a_i.*

PROOF.

$$\begin{aligned}
&PV_i(\vec{h}_i, a_i, C) = \textstyle\sum_{g_i \in G_i} \max_{a_C} Q_{g_i}(\vec{h}_i, a_i, a_C) \\
&= \textstyle\sum_{g_i \in G_i} \max_{a_D} [\max_{a_{C \setminus D}} Q_{g_i}(\vec{h}_i, a_i, a_D, a_{C \setminus D})] \\
&\geq \textstyle\sum_{g_i \in G_i} \max_{a_D} [\sum_{a_{C \setminus D}} Q_{g_i}(\vec{h}_i, a_i, a_D, a_{C \setminus D}) \\
&\qquad\qquad P_{g_i}(a_{C \setminus D} | \vec{h}_i, a_i, a_D)] \\
&= \textstyle\sum_{g_i \in G_i} \max_{a_D} Q_{g_i}(\vec{h}_i, a_i, a_D \\
&= PV_i(\vec{h}_i, a_i, D)
\end{aligned}$$

□

Based on the definition of the potential expected utility, we now can define the potential utility loss if an agent does not coordinate with some subgroup of agents when selecting its action.

DEFINITION 2. *The* **potential loss in lack of coordination** *$PLILOC_i(\vec{h}_i, NC)$ of agent i is the difference of the potential expected utility of agent i when it coordinates with all agents in I_i from that of agent i when it only coordinates with agents C, that is,*

$$PLILOC_i(\vec{h}_i, NC) = \max_{a_i} PV_i(\vec{h}_i, a_i, I_i) - \max_{a_i} PV_i(\vec{h}_i, a_i, NC). \tag{7}$$

The potential loss $PLILOC$ has following properties.

COROLLARY 1. *If $ND \subseteq NC \subseteq I_i$, then $PLILOC_i(\vec{h}_i, ND) \geq PLILOC_i(\vec{h}_i, NC)$ for all $\vec{h}_i$.*

COROLLARY 2. *For all $NC \subseteq I_i, 0 \leq PLILOC_i(\vec{h}_i, NC) \leq PLILOC_i(\vec{h}_i, \emptyset)$ for all $\vec{h}_i$.*

Borth Corollary 1 and 2 follow directly from Proposition 1. Corollary 1 indicates that the potential loss $PLILOC_i(\vec{h}_i, NC)$ monotonically decreases when agent i coordinates with additional agents in group NC at any observation history $\vec{h}$. Corollary 2 shows that $PLILOC(\vec{h}_i, NC)$ is always positive and bounded by the potential loss when agent i does not coordinate with any other agents.

Agents will use *potential loss in lack of coordination* $PLILOC$ to identify whom they should solely coordinate with in order to minimize its communication for coordinating its action selection with other agents without significantly affecting its learning performance.

Algorithm 2 shows how an agent searches for the best coordination set to trade off the quality of coordination and communication. This algorithm first tries to find a subgroup of agents NC with the maximum size such that the lack of coordination between agent i and agents NC will not cause a potential utility loss greater than $maxLoss$ and will result in the minimum loss among all subgroups with the same size. It then returns the difference of the set of all agents interacting with agent i from NC. As a result, this algorithm will return a coordination set that minimizes the communication while ensuring the quality of coordinated action selection. Note that the algorithm always can find such subgroup NC because $PLILOC_i(\vec{h}_i, \emptyset) == 0$. ~~Note that agents' local utilities Q_{g_i} can be negative, this is why that we use the formula shown in Line 1 to compute the max loss threshold.~~ By using this computed $maxLoss$, we can ensure when $\xi = 0$, every agent i will coordinate with all agents I_i, and when $\xi = 1$, every agent will not coordinate with any other agents. As the measure of *potential expected value* is usually overestimate of real expected value, the maximum loss $maxLoss$ that the agent calculates is also overestimated. As a result, the real rate of the utility loss using partial coordination will usually greater than threshold ξ.

4.3 Coordinating Action Selection

Our learning approach requires computing the coordinated action selection for updating local Q-functions or for acting during execution. When communication permits and agents do not need to trade off the learning quality and communication, our approach will compute the joint action that maximizes the total utility $\hat{Q} = \sum_{i \in I} \sum_{g_i \in G_i} Q_{g_i}$. However, as we discussed, in many practical systems, communication is limited and computing the action selection needs to be relatively fast. In those cases, our approach allows agents to find a minimum sub set of agents to coordinate their action selection while ensuring their learning performance. Now agents computing the joint action try to maximize the marginalized total utility

$$f(\vec{h}, a_1, ..., a_n) = \sum_i \sum_{g_i} Q_{g_i}(\vec{h}, a_i, a_{g_i \cap C_i}), \tag{8}$$

where $Q_{g_i}(\vec{h_i}, a_i, a_{a_{g_i \cap C_i}}) = \sum_{a_{g_i \setminus C_i}} Q_{g_i}(\vec{h_i}, a_i, a_{g_i})$ $P_{g_i}(a_{g_i \setminus C_i} | \vec{h_i}, a_i, a_{g_i \cap C_i})$ and C_i is the coordination set of agent i on observation history $\vec{h}$. When, for all agent i, $C_i = I_i$, then f is exactly the same as the total utility $\hat{Q}$.

We can formulate this joint action selection problem as a DCOP, which is defined by a set of discrete variables $a = \{a_1, \ldots, a_n\}$, where $a_i \in A_i$ is controlled by agent i and represents its action choice, and a set of functions $Q = \bigcup i\{Q_{g_i} | g_i \in G_i\}$. Note that history $\vec{h}$ is fixed for every computation, so we will ignore it in the following discussion and denote $Q_{g_i}(\vec{h}, a_i, a_{g_i})$ by $Q_l(a_i, a_{g_i})$. The goal is to find the coordinated joint action a^*, such that marginalized total utility f function is maximized.

We can represent this DCOP as a factor graph by creating a node for each variable and for each function and connecting a function node to a variable node if the corresponding function is dependent upon that variable. The resulting graph is bipartite. When agent i has a coordination set $C_i = \emptyset$, the variable node i in the factor graph will solely connect to its local utility functions $Q_{g_i}(a_i)$, which is also solely connect to the variable node i, which means agent i and its local functions are separated from factors and variables. When coordination sets are relatively small for many agents, it is highly likely that the original whole interaction network is decomposed into a set of disjointed sub networks, which will dramatically (may even exponentially) reduce communication when running DCOP algorithms.

A variable elimination algorithm [5] can be used to compute an optimal solution for this DCOP, but it requires extensive commu-

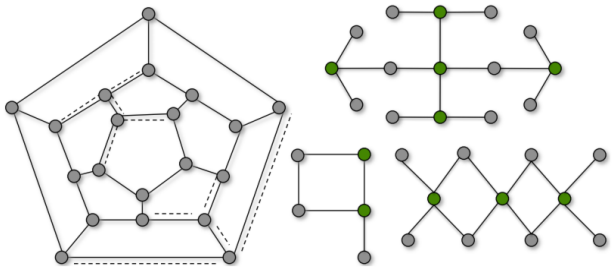

Figure 2: Sensor network configurations

nication and computation (scaling exponentially with the induced width of the agent interaction graph). As with [16], in this paper, we use the max-sum algorithm [11] for computing an approximate solution, which requires much less communication and computation and can be readily implemented as an anytime algorithm to trade off the quality and efficiency of computing joint actions.

The max-sum algorithm operates directly on the factor graph, and does so by specifying the messages that should be passed from variable to function nodes, and from function nodes to variable nodes, which are defined as follows:

- **Message from variable node i to function node g:**

$$q_{i\to g}(a_i) = \sum_{h\in\mathcal{F}_i\setminus g} r_{h\to i}(a_i) + c_{ig}$$

where $\mathcal{F}_i$ is a vector of function indexes, indicating which function nodes are connected to variable node i, and c_{ig} is a normalizing constant to prevent the messages from increasing endlessly in cyclic graphs.

- **Message from function node g to variable node i:**

$$r_{g\to i}(a_i) = \max_{\mathbf{a}_g\setminus a_i}[Q_l(\mathbf{a}_g) + \sum_{h\in\mathcal{V}_g\setminus i} q_{h\to g}(a_h)]$$

where $\mathcal{V}_g$ is a vector of variable indexes, indicating which variable nodes are connected to function node g and $\mathbf{a}_g\setminus a_i = \{a_h : h \in \mathcal{V}_g\setminus i\}$.

Here variable node i is agent i who needs to select its action and function node g hosts the local Q-value function Q_g. If the factor graph is cycle-free, the algorithm is guaranteed to converge to the optimal global solution such that each agent i can find its optimal action a_i^* by locally calculating $a_i^* = \text{argmax}_{a_i} z_i(a_i)$, where $z_i(a_i) = \sum_{h\in\mathcal{F}_i} r_{h\to i}(a_i)$. Otherwise, there is no guarantee of convergence. However, extensive empirical results show that, even in this case, the algorithm frequently provides good solutions. Before convergence, the value $z_i(a_i)$ of agent i calculated from incoming messages is actually an approximation of the exact value of action a_i given that other agents act optimally. The max-sum algorithm is essentially distributed. Its messages are small (linearly scaling with the maximum number of actions of agents), the number of messages typically varies linearly with the number of agents and hyperlinks, and its computational complexity scales exponentially with the maximum size of hyperlinks (which typically is much less than the total number of agents).

5. EXPERIMENTS

To evaluate our learning approach with emergent coordination, we experimented on a target tracking application in sensor networks modeled as an ND-POMDP [12]. Figure 2 shows four sensor topologies: 5P, 11H and 15-3D and 20D from [9]. Each node in these graphs is a sensor agent and edges are locations where targets can move. Tracking a target requires simultaneous scan of the edge by two adjacent sensors, producing a joint reward (+80), otherwise a penalty (-1) is given per scanning agent. The internal state of a sensor indicates its battery level (4 possible states). Each scan action depletes the battery and an empty battery renders the sensor unusable. Sensors can conserve power with the off action or recharge the battery at some cost (-1). Each sensor has three observations: target present, target absent and idle. The first two observations can be false positive/negative with accuracy=0.8. The 5P domain has 2 targets, 11H has 3 targets, 15-3D has 5 targets, and 20D has 6 targets.

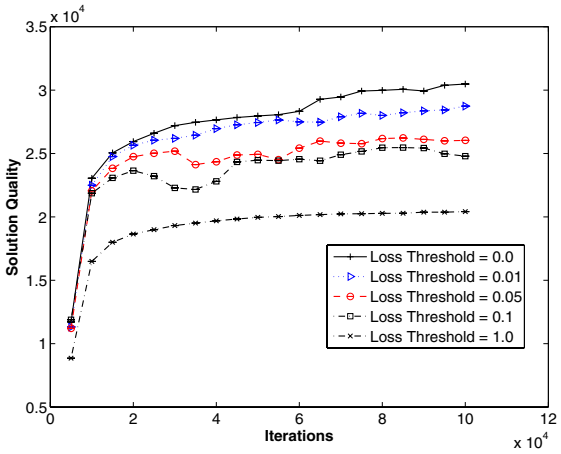

Figure 3: The performance of learning policies during the learning process in the 20D network

Since our coordinated learning approach is model-free, we developed a simulator for this target tracking application to learn and evaluate policies. Our learning approach learned policies that map fixed-windows of observations (with size ≤ 3) to an action even for scenarios with horizon greater than 4. Agents learn policies by repeatedly interacting with the simulator. To evaluate the learned policies, we run policies in the simulator until horizon=200. The solution quality of learned policies is computed by averaging over 10000 runs the total reward received from the simulator over horizon=200. The learning rate α for each truncated observation history $\vec{h}$ is set to $\min(0.001, visits(\vec{h}))$, where $visits(\vec{h})$ is the number of visits on $\vec{h}$ and discount factor $\gamma = 0.99$. We used the max-sum algorithm to compute coordinated actions and set the maximum round of message passing to the diameter of the connected factored graph.

Figures 3 and 4 show the trend of rewards of tracking targets and the number of messages used by the max-sum algorithm to coordinate action selection, respectively, as agents learn with different loss thresholds for computing coordination sets in the 20D sensor network. When the loss threshold = 0, agents' learning processes are fully coordinated, that is, they coordinate all other agents that interact with them, and when the loss threshold = 1, agents learn independently and have no coordination messages. As seen from Figures 3 and 4, as expected, with a smaller loss threshold, agents potentially coordinate with more other agents and learn better policies, but they also requires more communication messages. As the learning goes on, the performance of learned poli-

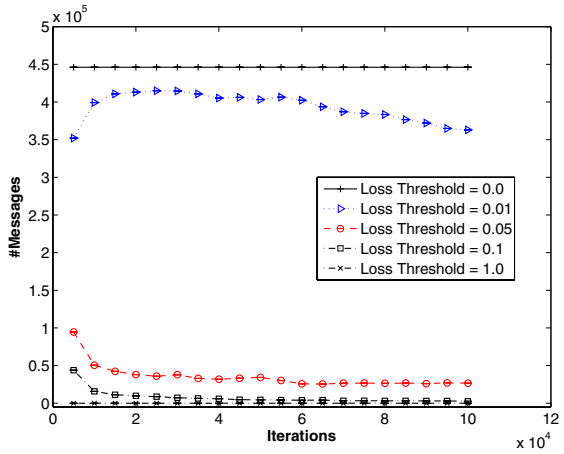

Figure 4: Communication for coordinating learning during the learning process in the 20D network

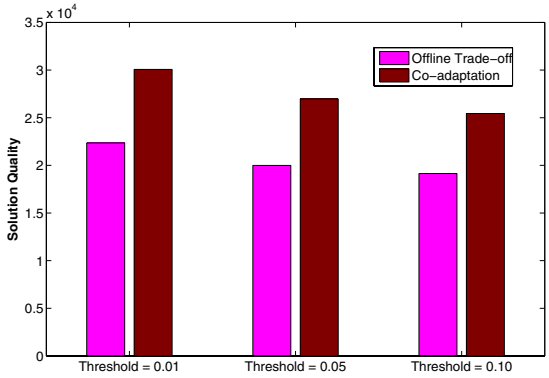

Figure 5: The performance of learned policies in the 20D network

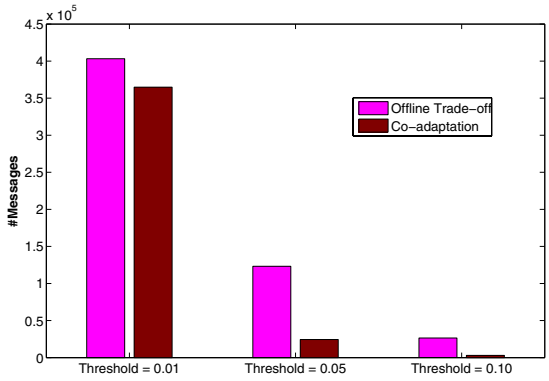

Figure 6: Communication for coordinating joint action selection in the 20D network

cies generally gradually improves. However, it seems not to be the case for threshold = 0.05 and 0.1, where the learning performance curve goes down at some points. This is because of co-adaptation of learning policies and dynamically computing coordination sets for agents. When an agent's coordination set changes, its learning environment can dramatically change, which causes the performance of learned policies decreases. As the time goes, agents will adapt their policies (i.e., utility functions) to this new environment and their performance improves. This process may iterate multiple times, as shown by the cases of threshold = 0.05 and 0.1 in Figure 3.

From Figure 4, we can also see that the number of messages for coordinating learning processes decreases as agents learn, which indicates agents are simultaneously learning better coordination sets while learning their policies. In the case for threshold=0.01, the communication increases at the early learning stage. Another interesting observation is that there is a big gap (with more than one order of magnitude difference at the late learning stage) in communication between cases of threshold=0.01 and threshold=0.05, while their learning performance does not differ as much, around 5%. We also observe this kind of phenomena in other network scenarios. For applications with very limited communication bandwidth or high communication cost, identifying such gaps may be crucial to find the best trade-off of communication and performance.

In order to illustrate the benefit of our approach's co-adaption of learning policies and dynamically computing coordination sets, we can compare it to another approach, called *offline trade-off*, where agents first learns policies with full coordination (i.e., threshold = 0), and then stop learning and, when executing learned policies, dynamically coordinate their action selection by computing coordinate set using Algorithm 2. Both approaches use the same number of learning cycles and then are evaluated by executing their learned policies without further learning. Figures 5 and 6 show the evaluation results of their performance and communication with different loss thresholds in the 20D sensor network, respectively. It is clear that, in all cases, our co-adaptation approach outperforms the offline trade-off approach with more than 30% in term of solution quality. It also uses significantly less communication for computing coordinated joint action when executing learned policies, especially, saving more than 80% in cases where loss threshold $= 0.05$ and 0.1.

Figures 7 and 8 show the performance and communication cost of executing policies learned by our co-adaptation approach using different loss thresholds in different sensor networks. We can see that, in all cases, the performance and communication generally increases as the loss threshold decreases. In all three cases, as with the 20D network configuration, it seems that setting the loss threshold ξ between 0.01 and 0.05 will significantly save communication (e.g., more than 80%) without degrading the performance much (e.g., less than 20% or even 5% for some cases). Another thing to note is that, as we discussed in Section 4.2, the loss threshold ξ may not be a good indicator to predict the whole system performance, because, when an agent computes the coordination set, it overestimates the maximum loss it can tolerate (i.e., larger than real expected utility discounted by ξ). As a result, we often observe that the actual system performance decreases more than ξ. For example, in the 15-3D network case, when agents use threshold $\xi = 0.05$, the actual performance decreases around 20%, 3 times larger than ξ. It is our future work to lower the estimate of the maximum loss when computing coordination sets and make ξ be a more predictable indicator for the system performance.

6. SUMMARY

We have introduced a coordinated multi-agent reinforcement learning (MARL) approach that generalizes previous approaches and allows agents to learn efficiently over a spectrum of applications with different communication bandwidth. Our approach provides

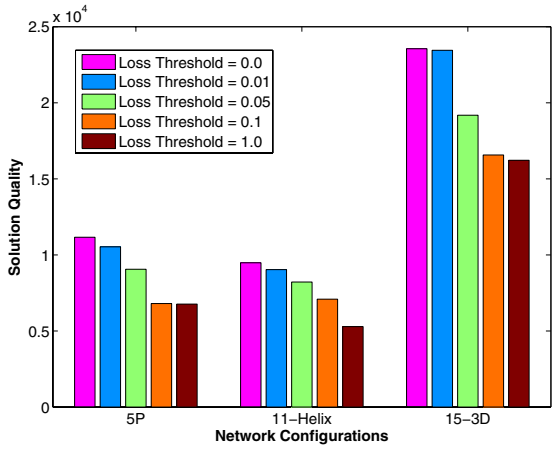

Figure 7: The performance of learned policies in different network configurations

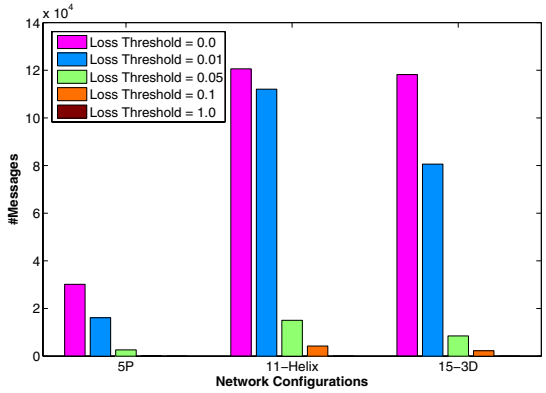

Figure 8: Communication for coordinating joint action in different network configurations

an interaction measure that enables learning agents to dynamically identify their coordination set in different situations to improve local and overall learning performance. By limiting the utility loss for lack of coordination, our approach minimizes agents' coordination sets while ensuring its learning performance. This minimization on agents' coordination sets will dramatically reduce links in the agent coordination network and often decomposes it into smaller connected sub networks, which allows DCOP algorithms to efficiently compute joint action and coordinate agents' learning processes. Our approach can scale MARL to large cooperative systems and improves its applicability over varied domains.

Two observations are worth noting. Proper co-adaptation of learning operational policies and identifying coordination set can improve the overall performance and reduce the communication cost for coordinating concurrent learning. Another observation is that many practical systems may have a crucial point that offers the best trade-off of performance and communication. Using the loss threshold, our learning approach will be useful to find such points.

7. ACKNOWLEDGMENT

This work is supported partially by the National Science Foundation (NSF) under Agreement IIS-1116078 . Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the National Science Foundation.

8. REFERENCES

[1] R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman. Transition-Independent Decentralized Markov Decision Processes. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 41–48, Melbourne, Australia, 2003. ACM Press.

[2] D. S. 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.

[3] S. Cheng. *Coordinating Decentralized Learning and Conflict Resolution Across agent Boundaries*. PhD thesis, University of North Carolina at Charlotte, 2012.

[4] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *AAAI'98*, pages 746–752. AAAI Press, 1998.

[5] C. Guestrin, D. Koller, and R. Parr. Multiagent planning with factored mdps. In *NIPS-14*, pages 1523–1530, 2001.

[6] C. Guestrin, M. G. Lagoudakis, and R. Parr. Coordinated reinforcement learning. In *ICML '02: Proceedings of the Nineteenth International Conference on Machine Learning*, pages 227–234, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[7] C. E. Guestrin. *Planning under uncertainty in complex structured environments*. PhD thesis, Stanford University, Stanford, CA, USA, 2003.

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

[9] A. Kumar, S. Zilberstein, and M. Toussaint. Scalable multiagent planning using probabilistic inference. In T. Walsh, editor, *IJCAI*, pages 2140–2146. IJCAI/AAAI, 2011.

[10] R. A. Mccallum. Instance-based utile distinctions for reinforcement learning with hidden state. In *In Proceedings of the Twelfth International Conference on Machine Learning*, pages 387–395. Morgan Kaufmann, 1995.

[11] R. Stranders, A. Farinelli, A. Rogers, and N. R. Jennings. Decentralised coordination of mobile sensors using the max-sum algorithm. In *IJCAI*, pages 299–304, 2009.

[12] P. Varakantham, M. Tambe, and M. Yokoo. Networked distributed pomdps: A synthesis of distributed constraint optimization and pomdps. In *AAAI*, pages 133–139, 2005.

[13] S. J. Witwicki and E. H. Durfee. Influence-based policy abstraction for weakly-coupled dec-pomdps. In R. I. Brafman, H. Geffner, J. Hoffmann, and H. A. Kautz, editors, *ICAPS*, pages 185–192. AAAI, 2010.

[14] C. Zhang, S. Abdallah, and V. Lesser. Integrating organizational control into multi-agent learning. In *AAMAS'09*, 2009.

[15] C. Zhang, V. Lesser, and S. Abdallah. Self-organization for coordinating decentralized reinforcement learning. In *AAMAS'10*, 2010.

[16] C. Zhang and V. R. Lesser. Coordinated multi-agent reinforcement learning in networked distributed pomdps. In W. Burgard and D. Roth, editors, *AAAI*. AAAI Press, 2011.