Self-Organization for Coordinating Decentralized Reinforcement Learning

Chongjie Zhang
Computer Science Dept.
University of Massachusetts
Amherst, MA 01002, US
chongjie@cs.umass.edu

Victor Lesser
Computer Science Dept.
University of Massachusetts
Amherst, MA 01002, US
lesser@cs.umass.edu

Sherief Abdallah
Institute of Informatics
British University in Dubai
Dubai, United Arab Emirates
sherief.abdallah@buid.ac.ae

ABSTRACT

Decentralized reinforcement learning (DRL) has been applied to a number of distributed applications. However, one of the main challenges faced by DRL is its convergence. Previous work has shown that hierarchically organizational control is an effective way of coordinating DRL to improve its speed, quality, and likelihood of convergence. In this paper, we develop a distributed, negotiation-based approach to dynamically forming such hierarchical organizations. To reduce the complexity of coordinating DRL, our self-organization approach groups strongly-interacting learning agents together, whose exploration strategies are coordinated by one supervisor. We formalize this idea by characterizing interactions among agents in a decentralized Markov Decision Process model and defining and analyzing a measure that explicitly captures the strength of such interactions. Experimental results show that our dynamically evolving organizations outperform predefined organizations for coordinating DRL.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence

General Terms

Algorithms, Measurement, Experimentation

Keywords

Multiagent Learning, Self-Organization, Coordination

1. INTRODUCTION

A collaborative multiagent system (MAS) consists of a group of agents that interact with each other in order to optimize a global performance measure. Theoretically, the underlying decision-making problem can be modeled as a decentralized Markov Decision Process (DEC-MDP) [1]. However, because of its complexity or the lack of access to the transition or reward model, it is infeasible to generate an optimal solution offline, except for the simplest cases. Distributed online learning provides an attractive, scalable, and approximate alternative, where each agent learns its policy based on its local observations and rewards. Example applications include packet routing [2, 3], sensor networks [4], distributed resource/task allocation [5], peer-to-peer information retrieval [6], and elevator scheduling [7].

Cite as: Self-Organization for Coordinating Decentralized Reinforcement Learning, Chongjie Zhang, Victor Lesser and Sherief Abdallah, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. XXX-XXX.

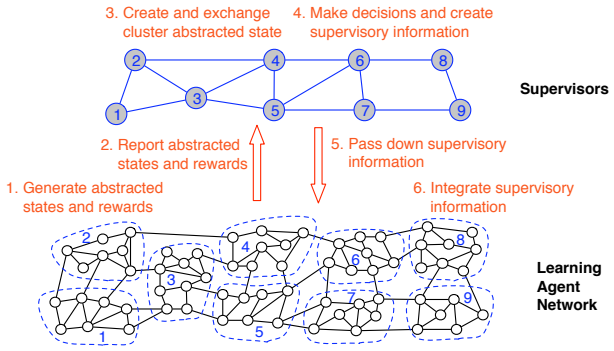

Figure 1: A supervision process of the organization-based control framework

However, due to factors including non-stationary learning environment, partial observability, a large number of agents and communication delay between agents, decentralized reinforcement learning (DRL) may converge slowly, converge to inferior equilibria, or even diverge in realistic settings. To deal with issues of DRL convergence, our previous work [8] proposed a supervision framework that employed periodic organizational control to coordinate and guide agents' learning exploration. The framework defined a multi-level organizational structure and a communication protocol for exchanging information between lower-level agents (or subordinates) and higher-level supervising agents (or supervisors) within an organization. As shown in Figure 1, subordinates reported their abstract states and rewards to their supervisors, which in turn generated and passed down supervisory information. The supervision framework also specified a supervisory policy adaptation that integrated supervisory information into the learning process, guiding subordinates' exploration of their state-action space. Empirical results demonstrated that hierarchically organizational control is an effective way of coordinating distributed learning to improve its speed, quality, and likelihood of convergence [8].

The supervision framework proposed in [8] , however, suffered from a serious limitation. The hierarchical organization, which formed the heart of the framework, was assumed to be given and fixed. Addressing this limitation involves answering the following questions: can supervisory

organizations automatically form while agents are concurrently learning their decision policies? do such dynamically evolving organizations perform better than static supervisory organizations? This paper makes a twofold contribution. First, we formalize joint-event-driven interactions among agents using a DEC-MDP model and define a measure for capturing the strength of such interactions. Second, we develop a distributed self-organization approach, based on the interaction measure, that dynamically adapts supervision organizations for coordinating DRL during the learning process. Unlike the work in [9], our self-organization process does not change the connectivity of the original agent network, but form a hierarchical supervisory organization on top of it. The key problem of the organization adaptation is to decide which agents need to be clustered together so that their exploration strategies can be coordinated. Our approach to this problem is inspired by the concept of *nearly decomposable systems* [10], where interactions between subsystems are generally weaker than interactions within subsystems. In order to improve the quality and reduce the complexity of coordinating DRL, our approach attempts to group agents together that strongly interact with each other. Unlike most of the previous work on self-organization (e.g., [11, 12]), our approach uses dynamic, rather than static, information about agents' behaviors based on their current state of learning. In our approach, the organization adaptation and individual agents' learning concurrently progress and interact with each other. Experimental results show that our dynamically evolving organizations outperform predefined organizations for coordinating DRL.

The rest of the paper is organized as follows. Section 2 reviews some background knowledge. Section 3 develops a distributed self-organization approach for dynamically evolving supervisory organizations to better coordinate DRL, and extends the supervision framework [8] to integrate our approach. Section 4 empirically evaluates our approach. Finally, Section 5 summarizes the contribution of this work.

2. BACKGROUND

This section reviews a DEC-MDP model for representing collaborative MAS, DRL for learning efficient approximate policies for agents in collaborative MAS, and the supervision framework for improving the performance of DRL.

2.1 Average-Reward, Factored DEC-MDP

We use factored DEC-MDP [13] to model the multiagent sequential decision-making problem in a collaborative MAS.

DEFINITION 1. *An n-agent factored DEC-MDP is defined by a tuple $\langle S, A, T, \mathcal{R}\rangle$, where*

- $S = S_1 \times \cdots \times S_n$ *is a finite set of world states, where S_i is the state space of agent i*
- $A = A_1 \times \cdots \times A_n$ *is a finite set of joint actions, where A_i is the action set for agent i*
- $T : S \times A \times S \to \Re$ *is the transition function. $T(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ is the probability of transiting to the next state $\mathbf{s}'$ after a joint action $\mathbf{a} \in A$ is taken by agents in state $\mathbf{s}$*
- $\mathcal{R} = \{R_1, R_2, \ldots, R_n\}$ *is a set of reward functions. $R_i : S \times A \to \Re$ provides agent i with an individual reward $\mathbf{r_i} \in R_i(\mathbf{s}, \mathbf{a})$ for taking action $\mathbf{a}$ in state $\mathbf{s}$. The global reward is a weighted sum of all local rewards: $R(\mathbf{s}, \mathbf{a}) = \sum_{i=1}^{n} w_i R_i(\mathbf{s}, \mathbf{a})$, where w_i is a weight.*

A policy $\pi : S \times A \to \Re$ is a function which returns the probability of taking action $\mathbf{a} \in A$ for any given state $\mathbf{s} \in S$. Similar to [14], the value function for a policy π is defined relative to the average expected reward per time step under the policy:

$$\rho(\pi) = \lim_{N\to\infty} \frac{1}{N}\mathbf{E}[\sum_{t=0}^{N-1} R(\mathbf{s}^t, \mathbf{a}^t)|\pi] \quad (1)$$

where the expectation operator $\mathbf{E}(\cdot)$ averages over stochastic transitions and $\mathbf{s^t}$ and $\mathbf{a}^t$ are the global state and the action taken at time t, respectively. The optimal policy is a policy that yields the maximum value $\rho(\pi)$.

Assume that the Markov chain of states under policy π is ergodic. The expected reward $\rho(\pi)$ then does not depend on the starting state. Let $p(\mathbf{s}|\pi)$ be the probability of being in state $\mathbf{s}$ under the policy π, which can be calculated as the average probability of being in state $\mathbf{s}$ at each time step over the infinite execution sequence:

$$p(\mathbf{s}|\pi) = \lim_{N\to\infty} \frac{1}{N} \sum_{t=0}^{N-1} P(\mathbf{s}^t = \mathbf{s}) \quad (2)$$

LEMMA 1. *Suppose $R(\mathbf{s})$ is the global reward function. Then the value of policy π is*

$$\rho(\pi) = \sum_{\mathbf{s}\in S} p(\mathbf{s}|\pi) \sum_{\mathbf{a}\in A} \pi(\mathbf{s}, \mathbf{a}) R(\mathbf{s}, \mathbf{a}) \quad (3)$$

The lemma follows immediately from Equation 2 and the definition of the policy value in Equation 1 based on the assumption that the state process is ergodic.

2.2 Decentralized Reinforcement Learning

DRL is used by agents to learn efficient approximate policies in a factored DEC-MDP environment, especially when the transition and reward function is unknown. Each agent learns its local policy based on its local observation and reward in presence of other agents, who are also learning a policy under the same conditions. The local policy $\pi_i : S_i \times A_i \to \Re$ for agent i returns the probability of taking action $a_i \in A_i$ in local state $s_i \in S_i$. As each agent only observes local reward signals, the value function of a local policy π_i of agent i is defined as:

$$\rho_i(\pi_i) = \lim_{N\to\infty} \frac{1}{N}\mathbf{E}[\sum_{t=0}^{N-1} r_i^t|\pi_i] \quad (4)$$

where the expectation operator $\mathbf{E}(\cdot)$ averages over both stochastic transitions and nondeterministic rewards and r_i^t is the local reward received at time t. The local reward $r_i^t = R_i(\mathbf{s^t})$ depends on the global state $\mathbf{s^t}$ and appears nondeterministic from the local perspective. The objective of agent i is to learn an optimal policy π_i^* to maximize $\rho_i(\pi_i)$.

Similar to Lemma 1, we can also reformulate the value function of the local policy.

LEMMA 2. *Suppose $\mathbf{E}[r_i(s_i)|\pi]$ is the expected local reward of taking action a_i in state s_i given a joint policy π.*

$$\rho_i(\pi_i|\pi_{-i}) = \sum_{s_i\in S_i} p(s_i|\pi) \sum_{a_i\in A_i} \pi_i(s_i, a_i)\mathbf{E}[r_i(s_i, a_i)|\pi], \quad (5)$$

where $p(s_i|\pi)$ as the probability of being in local state s_i under the joint policy π and π_{-i} is the set of policies of all agents except agent i.

Due to factored reward, we have the following lemma that can directly be proved from the definitions of factored DEC-MDP and value functions of both joint and local policies.

LEMMA 3. *The value of a joint policy is a weighted sum of the values of local policies, that is,*

$$\rho(\pi) = \sum_i w_i \rho_i(\pi_i|\pi_{-i}), \quad (6)$$

where the joint policy $\pi = (\pi_1, \ldots, \pi_n)$ *and* π_{-i} *is the set of policies of all agents except agent* i.

2.3 Organization-Based Control Framework For Supervising DRL

A supervision framework was proposed in [8] is intended to improve the speed, quality, and likelihood of DRL convergence. This framework employed low-overhead, periodic organizational control to coordinate and guide agents' exploration during the learning process. The supervision process described in Figure 1 contains two iterative activities: *information gathering* and *supervisory control*. During the information gathering phase, each learning agent records its execution sequence and associated rewards and does not communicate with its supervisor. After a period of time, agents move to the supervisory control phase. During this phase, each agent generates an abstracted state projected from its execution sequence over the last period of time and then reports it with an average reward to its cluster supervisors. After receiving abstracted states of its subordinate agents, a supervisor generates and sends an abstracted state of its cluster to neighboring supervisors. Based on abstracted states of its local cluster and neighboring clusters, each supervisor generates and passes down supervisory information, which is incorporated into the learning of subordinates and guides them to collectively learn their policies until new supervisory information arrives. After integrating supervisory information, agents move back to the information gathering phase and the process repeats.

To limit communication overhead, learning agents report their activities through their abstracted states. The abstract state of a learning agent captures its slow dynamics. It can be defined by features that are projected from fast-dynamics features, such as visited local states, local policy, or interactions with other agents, by using various techniques (e.g., averaging over the temporal scale). Similarly, abstracted states of a cluster based capture its slow dynamics, which can be projected from abstracted states of its members.

A supervisor uses *rules* and *suggestions* to transmit its supervisory information to its subordinates. A rule is defined as a tuple $\langle c, F\rangle$, where

- c: a condition specifying a set of satisfied states
- F: a set of forbidden actions for states specified by c

A suggestion is defined as a tuple $\langle c, A, d\rangle$, where

- c: a condition specifying a set of satisfied states
- A: a set of actions
- d: the suggestion degree, whose range is $[-1, 1]$

Rules are "hard" constraints on subordinates' behavior. Suggestions are "soft" constraints and allow a supervisor to express its preference for subordinates' behavior. A suggestion with a negative degree, called a *negative suggestion*, urges a subordinate not to do the specified actions. In contrast, a suggestion with a positive degree, called a *positive suggestion*, encourages a subordinate to do the specified action. The greater the absolute value of the suggestion degree, the stronger the suggestion.

Each learning agent uses the framework's supervisory policy adaptation to integrate rules and suggestions into its policy learned by a normal multiagent learning algorithm and generate an adapted policy. This adapted policy is intended to coordinate the agent's exploration with others. Rules are used to prune the state-action space. Suggestions bias an agent's exploration. If an agent's local policy agrees with its supervisor's suggestions, it is going to change its local policy very little; otherwise, it follows the supervisor's suggestions and makes a more significant change to its local policy. More formally, the integration works as follows:

$$\pi^A(s,a) = \begin{cases} 0 & \text{if } R(s,a) \neq \emptyset \\ \pi(s,a) + \pi(s,a) * \eta(s) & \\ \qquad * \, deg(s,a) & \text{else if } deg(s,a) \leq 0 \\ \pi(s,a) + (1 - \pi(s,a)) & \\ \qquad * \, \eta(s) * deg(s,a) & \text{else if } deg(s,a) > 0 \end{cases}$$

where π^A is the adapted policy, π is the learning policy, $R(s,a)$ is a set of rules applicable to state s and action a, $deg(s,a)$ is the degree of the satisfied suggestion, and $\eta(s)$ ranges from $[0,1]$ and determines the suggestion receptivity.

This supervision framework utilizes a hierarchy of control and data abstraction, which is conceptually different from existing hierarchical multi-agent learning algorithms that use a hierarchy of task abstraction. Unlike conventional heuristic approaches, this framework dynamically generates distributed supervisory heuristics based on partially global views to coordinate agents' learning. Supervisory heuristics guides the learning exploration without affecting policy update. In principle, the framework can work with any multi-agent learning algorithms. However, the supervision framework in [8] did not specify how to automatically construct proper hierarchical supervision organizations, which is the specific limitation addressed by this paper.

3. SUPERVISORY ORGANIZATION FORMATION

This section describes our approach to dynamically evolving a hierarchical supervisory organization for better coordinating DRL when agents are concurrently learning their decision policies. Organization formation is best described via answering two questions: how agent clusters are formed, and how a cluster supervisor is selected. Our approach adopts a relatively simple strategy for supervisor selection. Each cluster selects an agent as its supervisor that minimizes the communication overhead between supervisors and their subordinates. A new supervisor then establishes connections to supervisors of neighboring clusters based on the connectivity of their subordinates.

Agent clustering is to decide what agents should be grouped together so that their learning exploration strategies can be better coordinated by one supervisor. Because of limited resources of computation and communication, it is usually not feasible to put all agents together and use a fully centralized coordination mechanism. To deal with bounded resources and maintain satisficing performance of coordination, our clustering strategy is to cluster highly interdependent agents together, whose interactions have a great impact on the system performance, and meanwhile to minimize interactions across clusters. Thus the resulting system has a nearly de-

composable, hierarchical structure, which reduces the complexity of coordinating DRL in a distributed way.

To measure the interdependency between agents, we characterize a type of interactions among agents, called *joint-event-driven interactions*, in a DEC-MDP model. We also define a measure for the strength of such interactions, called *gain of interactions*, and analyze how interactions between agents contribute to the system performance by using this measure. Based on this measure, we then propose a distributed, negotiation-based agent clustering algorithm to form a nearly decomposable organization structure. Finally, we discuss how to extend supervision framework proposed in [8] to integrate our self-organization approach. For clarity, this paper focuses the discussion on forming a two-level hierarchy. Our organization formation approach can be iteratively applied in order to form a multi-level hierarchy.

3.1 Joint-Event-Driven Interactions

DEFINITION 2. *A* **primitive event** $e_j = \langle s_j, a_j \rangle$ *occurs when agent j executes action a_j in state s_j. A* **joint event** $\vec{e}_X = \langle e_{j_1}, e_{j_2}, \ldots, e_{j_h} \rangle$ *contains a set of primitive events generated by agents $X = \{j_1, j_2, \ldots, j_h\}$. A joint event $\vec{e}_X$ occurs iff all of its primitive events occur.*

Note that our definition of a joint event is different from that of an event in [15], where an event occurs if any one of its primitive events occurs. For brevity, events discussed in this paper refer to joint events. An event is used to capture the fact that some agents did some specific activities. A primitive event can be generated by either an agent or the external environment. For convenience, we treat the external environment as an agent.

DEFINITION 3. *A* **joint-event-driven interaction** $i_j^X = \langle \vec{e}_X, e_j \rangle$ *from a set of agents X onto agent j is a tuple that includes a joint event $\vec{e}_X$ and a primitive event e_j. A joint-event-driven interaction i_j^X is* **effective** *iff the event $\vec{e}_X$ affects the distribution over the resulting state of event e_j, that is, $\exists s_j \in S_j$ such that $p(s_j^{t+1} = s_j | e_j^t = e_j) \neq p(s_j^{t+1} = s_j | e_j^t = e_j, \vec{e}_X^t = \vec{e}_X)$, where t is the time.*

Here we define an interaction between agents as an affecting relationship, which is uni-directional. An effective interaction on an agent basically changes its transition function. If there exists an effective interaction $\langle \langle e_X \rangle, e_j \rangle$, then we say that agents X effectively interact with agent j.

Now we define a measure for the strength of interactions among agents. Let $E_X^j = \{\vec{e}_X | \exists e_j \in S_j \times A_j$ such that interaction $\langle \vec{e}_X, e_j \rangle$ is effective$\}$ be all joint events generated by a set of agents X that effectively interact with agent j.Let $V_j(s_j|\pi) = \sum_{a_j} \pi_j(s_j, a_j) \mathbf{E}[r_j(s_j, a_j)|\pi]$ be the expected value of being in state s_j, where π_j is the policy of agent j, and $\mathbf{E}[r_j(s_j, a_j)|\pi]$ is the expected reward of executing action a_j in state s_j.

DEFINITION 4. *The* **gain of interactions** *from a set of agents X to agent j, given a joint policy π, is*

$$g(X, j|\pi) = \sum_{\vec{e}_X \in E_X^j} p(\vec{e}_X|\pi) \sum_{s_j} p(s_j|\vec{e}_X, \pi) V_j(s_j|\pi),$$

where $p(\vec{e}_X|\pi)$ is the probability that event $\vec{e}_X$ occurs and $p(s_j|\vec{e}_X)$ is the probability of being in state s_j after $\vec{e}_X$ occurs.

The value of the gain of interactions is affected by two factors: how frequently agents effectively interact (reflecting on $p(\vec{e}_X|\pi)$) and how well they are coordinated (reflecting on $\sum_{s_j} p(s_j|\vec{e}_X) V_j(s_j|\pi)$). For example, in our experiments of distributed task allocation, if agents X frequently interact with agent j but they are not well coordinated, then the value of $g(X, j)$ tends to be a large negative value (all expected rewards are negative). Here ill-coordination means that agents X frequently generate events that cause agent j to be in states with low expected rewards. For instance, they send tasks to agent j when it is overloaded.

Obviously, if agents X do not effectively interact with agent j, then $g(X, j|\pi) = 0$ (because $E_X^j = \emptyset$). Now let us consider a special type of interactions among agents, called mutually exclusive interactions.

DEFINITION 5. *Two nonempty disjoint agent sets X and Y are said to* **mutually exclusively interact** *with agent j, iff $E_X^j = \emptyset \vee E_Y^j = \emptyset \vee p(s_j^{t+1} = s_j, e_j^t = e_j, \vec{e}_X^t = \vec{e}_X, \vec{e}_Y^t = \vec{e}_Y) = 0$, for all $s_j \in S_j, e_j \in S_j \times A_j, \vec{e}_X \in E_X^j, \vec{e}_Y \in E_Y^j$.*

If X and Y mutually exclusively interact with agent j, then no two effective interactions generated by X and Y, respectively, will simultaneously occur to affect the state transition of agent j. In many applications [2, 4, 6], agents have such a type of interactions. For example, in network routing [2], the state space is defined by the destination of packages and each decision of an agent is triggered by one routing packet sent by one agent, so any two agents mutually exclusively interact with any third agent. Mutually exclusive interaction has the following property.

PROPOSITION 1. *If X and Y mutually exclusively interact with agent j, then $g(X \cup Y, j|\pi) = g(X, j|\pi) + g(Y, j|\pi)$.*

PROOF. Let E_X and E_Y be all events generated by X and Y, respectively.

$$\begin{aligned}
g(X \cup Y, j|\pi) &= \sum_{\vec{e}_{XY} \in E_{X \cup Y}^j} p(s_j, \vec{e}_{XY}|\pi) V_j(s_j|\pi) \\
&= \sum_{\vec{e}_X \in E_X^j} \sum_{\vec{e}_Y \in E_Y} \sum_{s_j} p(s_j, \vec{e}_X, \vec{e}_Y|\pi) V_j(s_j|\pi) \\
&\quad + \sum_{\vec{e}_X \in E_X} \sum_{\vec{e}_Y \in E_Y^j} \sum_{s_j} p(s_j, \vec{e}_X, \vec{e}_Y|\pi) V_j(s_j|\pi) \\
&\quad - \sum_{\vec{e}_X^j \in E_X} \sum_{\vec{e}_Y \in E_Y^j} \sum_{s_j} p(s_j, \vec{e}_X, \vec{e}_Y|\pi) V_j(s_j|\pi) \\
&= \sum_{\vec{e}_X \in E_X^j} \sum_{s_j} p(s_j, \vec{e}_X|\pi) V_j(s_j|\pi) \\
&\quad + \sum_{\vec{e}_Y \in E_Y^j} \sum_{s_j} p(s_j, \vec{e}_Y|\pi) V_j(s_j|\pi) \\
&= g(X, j|\pi) + g(Y, j|\pi)
\end{aligned}$$

□

Let $\mathcal{X}$ be all agents in a system and $\mathcal{X}_j \subseteq \mathcal{X}$ be a set of agents that effectively interact with agent j.

PROPOSITION 2. *If every two agents in $\mathcal{X}_j$ mutually exclusively interact with agent j, then*

$$\rho_j(\pi_j|\pi_{-j}) = \sum_{x \in \mathcal{X}_j} g(\{x\}, j|\pi).$$

PROOF.

$$\begin{aligned}\rho_j(\pi_j|\pi_{-j}) &= \sum_{s_j} p(s_j|\pi)V_j(s_j|\pi)|\pi) \\ &= \sum_{\vec{e}_X \in E^j_{\mathcal{X}_j}} p(\vec{e}_X|\pi) \sum_{s_j} p(s_j|\vec{e}_X)V_j(s_j|\pi) \\ &= g(\mathcal{X}_j, j|\pi) \\ &= \sum_{x \in \mathcal{X}_j} g(\{x\}, j|\pi)\end{aligned}$$

□

COROLLARY 1. *If every pair of agents in $\mathcal{X}$ mutually exclusively interact with any third agent, then*

$$\sum_{j \in \mathcal{X}} \sum_{x \in \mathcal{X}} w_j g(\{x\}, j|\pi) = \rho(\pi).$$

This corollary follows immediately from Lemma 3 and Proposition 2. Proposition 2 and Corollary 1 show how interactions contribute to the local and global performance, respectively, that is, the greater the absolute value of the weighted gain of interactions between two agents, the greater the (positive or negative) potential impact of their interactions on both the local and global performance. Although the properties of the gain of interactions we have just shown are valid in a restricted case, it can also be shown that the global performance measure can be tightly bounded by a weighted sum of gains of interactions among agents, which are approximately mutually exclusive. Therefore, the weighted gain can generally reflect the strength of interactions between agents, which is the basis of our self-organization approach.

3.2 Distributed Agent Clustering through Negotiation

Our clustering algorithm is intended to form a nearly decomposable organization structure, where interactions between clusters are generally weaker than interactions within clusters, to facilitate coordinating DRL. We assume all reward weights are equal and use the absolute value of the gain of interactions to measure the strength of interactions among agents. Supervisory organizations formed by using this measure will favorably generate rules and suggestions to improve ill-coordinated interactions (i.e. with a large negative gain) and maintain well-coordinated interactions (i.e., with a large positive gain), which potentially improve the performance of DRL. Our algorithm does not require interactions between agents to be mutually exclusive.

Due to bounded computational and communication resources, we limit the cluster size to control the quality and complexity of coordination. Our clustering problem is formulated as follows: given a set of agents $\mathcal{X}$ and the maximum cluster size θ, subdivide $\mathcal{X}$ into a set of clusters $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$, such that

1. $\forall i = 1, \ldots, m, |C_i| \leq \theta$,
2. $\cup C_i = \mathcal{X}$ and $\forall i \neq j, C_i \cap C_j = \emptyset$,
3. The total utility of clusters $U(\mathcal{C}) = \sum_{C_i \in \mathcal{C}} U(C_i)$ is maximal, where $U(C_i)$ is the utility of a cluster C_i defined as follows:

$$U(C_i) = \sum_{x_i, x_j \in C_i \text{ and } x_i \neq x_j} |g(\{x_i\}, x_j)| \qquad (7)$$

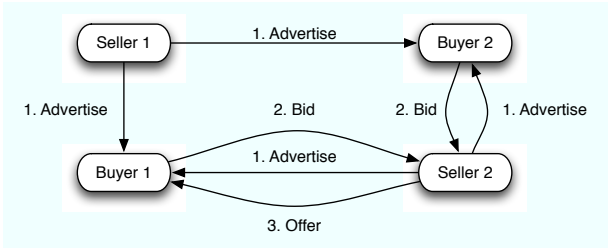

Figure 2: Self-organization negotiation protocol

Note that the total utility $U(\mathcal{C})$ has no direct relation to the system performance measure $\rho(\pi)$. The purpose of our clustering algorithm is not to directly improve the system performance, but to form proper supervisory organizations for coordinating learners that are ill-coordinated so as to potentially improve the learning performance.

Our clustering approach is distributed and based on an iterative negotiation process that involves a two roles: a buyer and a seller. A *buyer* is a supervisor who plans to expand its control and recruit additional agents into its cluster. A *seller* is a supervisor who has agents that the buyer would like to have. Supervisors can be buyers and sellers simultaneously. A *transaction* is to transfer a nonempty subset of boundary subordinates from a seller's cluster to a buyer's cluster. The *local marginal utility* is the difference between a cluster's utility before a transaction and the utility after the transaction. The *social marginal utility* is the sum of the local marginal utilities of both the buyer and the seller.

Based on these terms, our clustering problem can be translated into deciding which sellers the buyers should attempt to get agents from and which buyers the sellers should sell their agents to so that $U(\mathcal{C})$ is maximized.

The input to our clustering algorithm is an initial supervisory organization and the gain of interactions between agents. Figure 2 shows the dynamics of the negotiation protocol. Each supervisor only negotiates with its immediate supervisors. As our system is cooperative, our negotiation decisions are based on marginal social utility calculation. A round of negotiation consists of the following sub-stages:

1. Seller advertising: the supervisor of each cluster C_i sends an advertisement to each neighboring buyer. The advertisement contains local marginal utility $U^{lm}(C_i/X) = U(C_i) - U(C_i/X)$ of giving up each nonempty subset X of its subordinates adjacent to the buyer's cluster.

2. Buyer bidding: the supervisor of each cluster C_j waits for a period of time, collecting advertisements from neighboring supervisors. When the period is over, it calculates local marginal utility $U^{lm}(C_j \cup X) = U(C_j \cup X) - U(C_j)$ and then social marginal utility $U^{sm}(C_j, C_i, X) = U^{lm}(C_j \cup X) - U^{lm}(C_i/X)$ for introducing each nonempty subset X of subordinates of a seller of cluster C_i. If $U^{sm}(C_j, C_i, X)$ is the greatest social marginal utility and $U^{sm}(C_j, C_i, X) > 0$, then the buyer sends a bid to the supervisor of cluster C_i with the social marginal utility $U^{sm}(C_j, C_i, X)$; otherwise, do nothing.

3. Selling: given the multiple responses from buyers during a period time, the supervisor of cluster C_i chooses to transfer a subset of subordinates X to the cluster

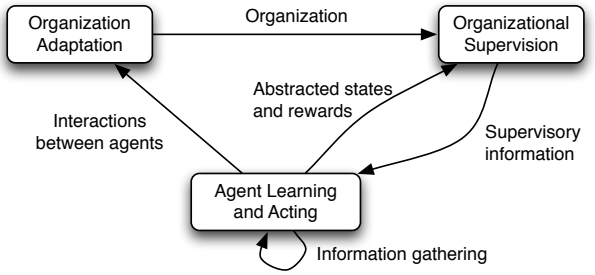

Figure 3: Extended supervision framework

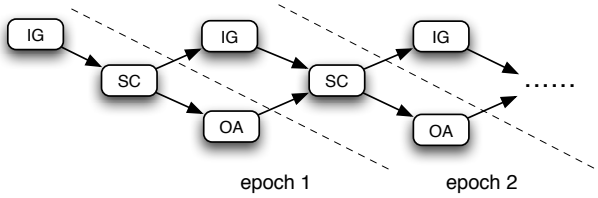

Figure 4: Iterations of three activities: information gathering (IG), supervisory control (SC), and organization adaptation (OA)

C_j if $U^{sm}(C_j, C_i, X)$ is the maximal social marginal utility that the seller receives during this round.

The basic idea of our approach is similar to the LID-JESP algorithm [16] and the distributed task allocation algorithm in [17]. LID-JESP is used to generate offline policies for agents in a special DEC-POMDP, called ND-POMDP. However, we focus on agent clustering. Our negotiation strategy is also similar to that in [12], but uses one less sub-stage in each round of negotiation.

PROPOSITION 3. *When our clustering algorithm is applied, the total utility $U(\mathcal{C})$ strictly increases until local optimum is reached.*

PROOF SKETCH. By construction, only non-neighboring supervisors can transfer some subordinates to their neighboring clusters and they will only do this if the social marginal utility is positive, which results in an increase of the total utility $U(\mathcal{C})$. In addition, a supervisor's transferring subordinates to a neighboring cluster will not affect the utility of other neighboring clusters and non-neighboring clusters. Thus with each cycle the total utility is strictly increasing until local optimum is reached. □

3.3 Extended Supervision Framework

The gain of interactions is defined on the transition function, the reward function, and a specific joint policy. However, as all agents are learning their decision policies, interactions between agents may change over the time. To deal with this issue, we decompose the system runtime into a sequence of epochs. The gain of interactions between agents is approximately estimated from their execution trace during an epoch. Each epoch contains three activities: *information gathering*, and *supervisory control* and *organization adaptation*. The supervision framework proposed in [8] is now extended to allow dynamically evolving supervisory organizations for better coordinating DRL when agents are concurrently learning their decision policies. As shown in Figure 3, the extended framework contains these three interacting activities. Three activities iterate in the way as shown in Figure 4 during the whole system runtime.

Both information gathering activity and supervisory control activity have been discussed in detail in Section 2.3. With this extended framework, during the information gathering phase, each agent collects information about interactions from its neighbors, in addition to its execution sequence and reward information. After a period of time, agents will move to supervisory control phase, at the beginning of which each agent will calculate the gain of interactions with its neighbors and report it along with other information (i.e., abstracted states and rewards) to its supervisor. To avoid interfering the DRL supervision, organization adaption only happens after the supervisory control phase. However, since there is no communication between learning agents and their supervisors during the information gathering stage, organization adaption can be conducted concurrently with the next phase of information gathering. During this phase, using information of subordinates' interactions with their neighbors, supervisors run our negotiation-based clustering algorithm and supervisor selection strategy to dynamically adapt the current supervisory organization. The resulting organization will be used for the next supervisory control activity. Initially, the system starts with a very simple supervisory organization, where each agent is its own supervisor. Then the supervisory organization is periodically evolving as agents are learning and acting.

4. EXPERIMENTS

We evaluated our approach in a distributed task allocation problem (DTAP) [8] with Poisson task arrival distribution and exponentially distributed service time. Agents are organized in a network. Each agent may receive tasks from either the environment or its neighbors. At each time unit, an agent makes a decision for each task received during this time unit whether to execute the task locally or send it to a neighbor for processing. A task to be executed locally will be added to the local queue. Agents interact via communication messages and communication delay between two agents is proportional to the distance between them. The main goal of DTAP is to minimize the average total service time (ATST) of all tasks, including routing time, queuing time, and execution time.

4.1 Experimental Setup

We chose one representative MARL algorithm, the Weighted Policy Learner (WPL) algorithm [18], for each worker to learn task allocation policies. WPL is a gradient ascent algorithm where the gradient is weighted by $\pi(s,a)$ if it is negative; otherwise, it will weighted by $(1-\pi(s,a))$. A worker's state is defined by a tuple $\langle l, f\rangle$, where l is the current work load (or total work units) in the local queue and f is a boolean flag indicating whether there is a task to be made a decision. Each neighbor corresponds to an action which forwards a task to that neighbor, and an agent itself corresponds to the action that put a task to the local queue. The reward $r(s,a)$ of doing an action a for an task is the negative value of the expected service time to complete the task after doing a in state s, which is estimated from previ-

ous finished tasks. All agents use WPL with learning rate 0.001.

The abstracted state of a worker is projected from its states and defined by its average work load over a period of time τ ($\tau = 500$ in our experiments). The abstracted state of a supervisor is defined by the average load of its cluster, which can be computed from the abstracted states of its subordinates. A subordinate sends a report, which contains its abstracted state, to its supervisor every τ time period. Supervisors use simple heuristics to generate rules and suggestions. With an abstracted state $\langle \bar{l} \rangle$, a supervisor generates a rule that specifies, for all states whose work load exceeds $\bar{l}$, a worker should not add a new task to the local queue. This rule helps balance load within the cluster. A supervisor also generates positive (or negative) suggestions for its subordinates to encourage (or discourage) them forwarding more tasks to a neighboring cluster that has a lower (or higher) average load. The suggestion degree for each subordinate depends on the difference between the average load of two clusters, the number of agents on the boundary, and the distance of the subordinate to the boundary. Therefore, suggestions are used to help balance the load across clusters. The implementation detail of generating supervisory information is discussed in [19]. Our experiments use the receptivity function $\eta(s) = 1000/(1000 + visits(s))$, where $visits(s)$ is the number of visits on state s.

To allow its supervisor to run our negotiation-based self-organization algorithm, each agent calculates the gain of interactions from other agents. As mentioned in Section 3.3, because of learning, each agent needs to approximately estimate each component in the definition of the gain of interactions from the history of its local executions and interactions with other agents in order to calculate it. In DTAP, one agent only interacts with its neighbors by forwarding tasks to them and its state does not affect states of its neighbors. Let $\vec{e}_k^j$ be the event of agent k, forwarding a task to agent j, that effectively interacts with agent j. To calculate $g(\{k\}, j|\pi)$, agent j estimates $p(\vec{e}_k^j|\pi)$ as the ratio of the number of tasks received from agent k to the total number of received tasks and $p(s_j|\vec{e}_k^j)$ as the ratio of the number of visits on state s_j resulting from $\vec{e}_k^j$ to the total number of visits on this state, and uses its current learned policy π_j and reward function r_j.

Three measurements are evaluated: average total service time (ATST), average number of messages (AMSG) per task, time of convergence (TOC), and average cluster size (ACS). ATST indicates the overall system performance. AMSG takes into account all messages for routing task, coordination, and self-organization negotiation. To calculate TOC, we take sequential ATST values with certain size. If the ratio of those values' deviation to their mean is less than a threshold (we use threshold of 0.025), we consider the system stable. TOC is the start time of the selected points. ACS is the average cluster size in the system at TOC.

Experiments were conducted using a 18x18 grid network with 324 agents. All agents have the same execution rate and tasks are not decomposable. The mean of task service time is $\mu = 10$. We tested two patterns of task arrival:

Side Load where agents in a 3x3 grid at the middle of each side receive tasks with rate $\lambda = 0.8$ and other agents receive no tasks from the external environment.

Corner Load where only agents in the 8x8 grid at the upper left corner receive tasks from the external environment. Within that grid, the 36 agents at the upper left corner has the task arrival rate $\lambda = 0.25$ and the rest agents has the rate $\lambda = 0.7$.

We compared the DRL performance under four cases: *None, Fixed-Small, Fixed-Large, and Self-Org.* In the *None* case, no supervision is used to coordinate DRL. Both *Fixed-Small* and *Fixed-Large* cases use a fixed organization, the former with 36 clusters, each of which is a 3x3 grid, and the latter with 9 clusters, each of which is a 6x6 grid. The *Self-Org* case uses our self-organization approach to dynamically evolving supervision organization.

In each simulation run, ATST and AMSG are computed every 500 time units to measure the progress of the system performance. Results are then averaged over 10 simulation runs and the variance is computed across the runs.

4.2 Experimental Results

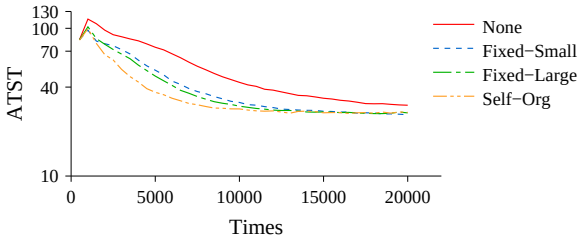

Figure 5: ATST under side load

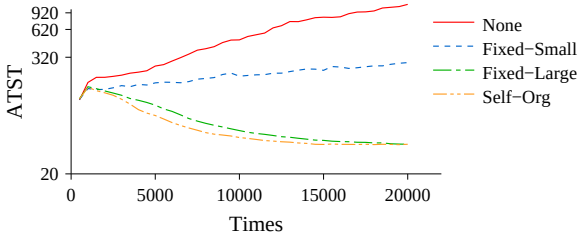

Figure 6: ATST under corner load

Figure 5 and 6 plot the trends of ATST, as agents learn, for different organization structures with different task arrival patterns. Note that the y axis in the plots is logarithmic. The supervision framework generally improves both the likelihood and speed of the learning convergence. Supervision with self-organized structure has a better learning curve than that with predefined organization structures. This is because our self-organization approach clusters highly interdependent agents together, and focused coordination on them tends to greatly improve the system performance. The *Fixed-Small* case has a small cluster size and consequently some highly interdependent agents are not coordinated well. In contrast, the *Fixed-Large* case has a large cluster size, which enlarges both the view and control of each supervisor and potentially improve the system performance. However,

with a large cluster size, an abstracted state of a cluster (generated by a supervisor) tends to lose detailed information about its subordinates, and also weakily interdependent agents are mixed with highly interdependent agents, both of which degrade the coordination quality.

Under corner load, the system with both *None* and *Fixed-Small* cases seems not to converge. For the *None* case, due to communication delay and limited views, agents in the top-left conner do not learn quickly enough knowledge about where light-loaded agents are. As a result, more and more tasks loop and reside in the top-left 8x8 grid. This makes the system load severely unbalanced and the system capability not well utilized, which causes the system load to monotonically increase. For the *Fixed-Small* case, because of a small cluster size, a supervisor's local view of the system may not be consistent with the global view. Some supervisors of overloaded clusters find their neighbors having even higher loads and consider their own clusters are "lightly" loaded. As a result, they generate incorrect directives for their subordinates, which degrade their normal learning.

Structure	ATST	AMSG	TOC	ACS
None	33.47 ± 1.67	5.81 ± 0.07	13000	0
Fixed-Small	29.09 ± 1.27	6.04 ± 0.11	10000	9
Fixed-Large	29.30 ± 1.46	6.16 ± 0.14	8500	36
Reorg	28.98 ± 1.15	6.59 ± 0.08	6500	14.50 ± 0.55

Table 1: Performance under side load

Structure	ATST	AMSG	TOC	ACS
None	N/A	N/A	N/A	0
Fixed-Small	N/A	N/A	N/A	9
Fixed-Large	44.94 ± 2.10	11.26 ± 0.10	12500	36
Self-Org	42.87 ± 2.06	11.41 ± 0.05	10500	25.33 ± 2.16

Table 2: Performance under corner load

Table 1 and 2 show different measures for each supervision structure at their respective convergence time points. Due to the system divergence, both the *None* and *Fixed-Small* cases have no data under corner load. In addition to improving the convergence rate, the supervision framework also decreases the system ATST. Self-organization further improves the coordination performance, as indicated by its ATST and TOC. Because of negotiations, the self-organization case has a slightly heavier communication overhead than those of fixed organizations.

5. CONCLUSION

In this paper, we formally define and analyze a type of interactions, called joint-event-driven interactions, among agents in a DEC-MDP. Based on this analysis, we develop a distributed self-organization approach that dynamically adapts hierarchical supervision organizations for coordinating DRL during the learning process. Experimental results demonstrate that dynamically evolving hierarchical organizations outperform predefined organizations in terms of both the probability and the quality of convergence.

6. REFERENCES

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

[2] Justin A. Boyan and Michael L. Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In *NIPS'94*, volume 6, pages 671–678, 1994.

[3] David H. Wolpert, Kagan Tumer, and Jeremy Frank. Using collective intelligence to route internet traffic. In *NIPS'99*, pages 952–958, 1999.

[4] Ying Zhang, Juan Liu, and Feng Zhao. Information-directed routing in sensor networks using real-time reinforcement learning. *Combinatorial Optimization in Communication Networks*, 18:259–288, 2006.

[5] Chongjie Zhang, Victor Lesser, and Prashant Shenoy. A Multi-Agent Learning Approach to Online Distributed Resource Allocation. In *IJCAI'09*, 2009.

[6] Haizheng Zhang and Victor Lesser. A reinforcement learning based distributed search algorithm for hierarchical content sharing systems. In *AAMAS'07*, 2007.

[7] Robert Crites and Andrew Barto. Improving elevator performance using reinforcement learning. In *NIPS'96*, pages 1017–1023, 1996.

[8] Chongjie Zhang, Sherief Abdallah, and Victor Lesser. Integrating organizational control into multi-agent learning. In *AAMAS'09*, 2009.

[9] Sherief Abdallah and Victor Lesser. Multiagent reinforcement learning and self-organization in a network of agents. In *AAMAS'07*, 2007.

[10] H. A. Simon. Nearly-decomposable systems. In *The Sciences of the Artificial*, pages 99–103, 1969.

[11] Bryan Horling and Victor Lesser. Using quantitative models to search for appropriate organizational designs. *Autonomous Agents and Multi-Agent Systems*, 16(2):95–149, 2008.

[12] Mark Sims, Claudia Goldman, and Victor Lesser. Self-Organization through Bottom-up Coalition Formation. In *AAMAS'03*, pages 867–874, 2003.

[13] Carlos Ernesto Guestrin. *Planning under uncertainty in complex structured environments*. PhD thesis, Stanford University, Stanford, CA, USA, 2003.

[14] Marek Petrik and Shlomo Zilberstein. Average-reward decentralized markov decision processes. In *IJCAI*, pages 1997–2002, 2007.

[15] Raphen Becker, Victor Lesser, and Shlomo Zilberstein. Decentralized Markov Decision Processes with Event-Driven Interactions. In *AAMAS'04*, volume 1, pages 302–309, 2004.

[16] Ranjit Nair, Pradeep Varakantham, Milind Tambe, and Makoto Yokoo. Networked distributed pomdps: a synthesis of distributed constraint optimization and pomdps. In *AAAI'05*, pages 133–139, 2005.

[17] Michael Krainin, Bo An, and Victor Lesser. An Application of Automated Negotiation to Distributed Task Allocation. In *IAT'07*, pages 138–145, 2007.

[18] Sherief Abdallah and Victor Lesser. Learning the task allocation game. In *AAMAS'06*, 2006.

[19] Chongjie Zhang, Sherief Abdallah, and Victor Lesser. MASPA: Multi-agent automated supervisory policy adaptation. In *University of Massachusetts Amherst Computer Science Technical Report #08-03*, 2008.