Integrating Organizational Control into Multi-Agent Learning

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

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

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

ABSTRACT

Multi-Agent Reinforcement Learning (MARL) algorithms suffer from slow convergence and even divergence, especially in large-scale systems. In this work, we develop an organization-based control framework to speed up the convergence of MARL algorithms in a network of agents. Our framework defines a multi-level organizational structure for automated supervision and a communication protocol for exchanging information between lower-level agents and higher-level supervising agents. The abstracted states of lower-level agents travel upwards so that higher-level supervising agents generate a broader view of the state of the network. This broader view is used in creating supervisory information which is passed down the hierarchy. The supervisory policy adaptation then integrates supervisory information into existing MARL algorithms, guiding agents' exploration of their state-action space. The generality of our framework is verified by its applications on different domains (distributed task allocation and network routing) with different MARL algorithms. Experimental results show that our framework improves both the speed and likelihood of MARL convergence.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation

Keywords

Multi-Agent Learning, Organization Control, Policy Adaptation, Coordinated Learning, Supervision

1. INTRODUCTION

A central challenge in multi-agent systems (MAS) research is to design distributed coordination mechanisms for agents that have only partial views of the whole system in order to generate efficient solutions to complex, distributed problems. To effectively coordinate their actions, agents need to estimate the unobserved states of the system and adapt their actions to the dynamics of the environment. Multi-agent reinforcement learning (MARL) techniques have been extensively explored in such a setting.

In order to achieve scalability in conventional approaches [2, 4, 15] to MARL, the learning of each agent has been restricted to using information received only from its immediate neighbors to update its estimates of the world states (i.e., Q-values for state-action pairs). However, this constraint results in long latency as state information propagates to agents further away. Such latency can result in neighborhood information being outdated, leading to mutually inconsistent views among agents. As a result, such a limited view for each agent and the non-stationarity of the environment (all agents are simultaneously learning their own policies) causes MARL algorithms to converge slowly and even diverge in some cases. The slowness of MARL convergence is further degraded by the large policy search space of each agent. Each agent's policy not only includes its local state and actions but also some characteristics of the states and actions of its neighboring agents [2], or the state size of each agent may be proportional to the number of agents in the system [4].

Two paradigms have been studied to speed up the multi-agent learning process. The first paradigm is to reduce the policy search space. For example, the TPOT-RL [11] reduced the state space by mapping states onto a limited number of action-dependent features. Another approach is hierarchical multi-agent reinforcement learning [5], where the explicit task structure was used to restrict the space of policies. Each agent learned joint abstract action-values by communicating with others only the state of high-level subtasks. The second paradigm is to employ heuristics to guide the policy search. Heuristically Accelerated Minimax-Q (HAMMQ) [3] incorporated heuristics into the Minimax-Q algorithm to speed up its convergence rate. HAMMQ shared the convergence property with Minimax-Q. However, HAMMQ was intended for use only in a two-agent configuration. Its authors used hand-coded domain heuristics, which did not capture the dynamics of other learning agents. Another approach [12] used both local and global heuristics to accelerate the learning process in a decentralized multirobot system. The local heuristic was derived from a robot's local information, while the global heuristic was derived from the global data obtained from other robots. The global data needed to be exactly the same among robots. This consistency was maintained by broadcasting messages among all robots, which incurred heavy communication overhead and did not scale well. In addition, this approach was developed specifically for the multirobot patrolling problem.

In this paper, we present a different approach, called Multi-Agent Supervisory Policy Adaptation (MASPA), that employs organizational control to guide multi-agent learning and accelerate its convergence. MASPA is composed of three components: a multi-

Cite as: Integrating Organizational Control into Multi-Agent Learning, Chongjie Zhang, Sherief Abdallah and Victor Lesser, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. XXX-XXX.

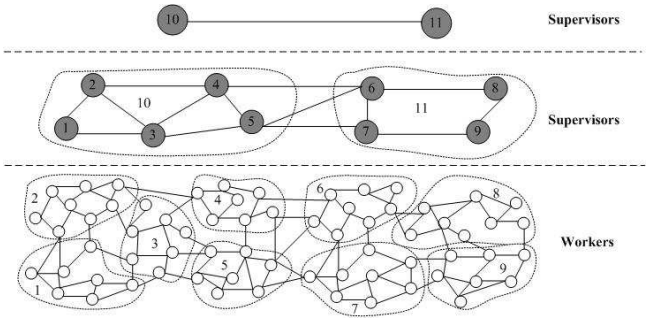

Figure 1: An organizational structure for multi-level supervision

level supervision organization (a meta-organization built on top of the agents' overlay network), a communication protocol for exchanging information between lower-level agents and higher-level supervising agents, and a policy adaptation mechanism that integrates organizational control information into MARL algorithms (e.g., GIGA [16], WPL [1], etc.) to bias the exploration process of each learning agent.

The key idea of MASPA is as follows. Each level in the supervision organization is an overlay network in itself. For example, Figure 1 shows a three-level supervision organizational structure. The abstracted states of lower-level agents travel upwards so that higher-level supervising agents can generate a broader view of the state of the network. This broader view comes from not only information about the states of lower-level agents but also information from neighboring supervising agents. In turn, this broader view results in creating supervisory information which is passed down the hierarchy. This supervisory information guides the learning of agents in collectively exploring their state-action spaces more efficiently, and consequently results in faster convergence. To provide up-to-date supervisory information, the process above is periodically repeated.

In this way, MASPA deals with scalability issues by using approximate partial global views that can be acquired with relatively low overhead. The use of these dynamic views does not increase the state space of individual agent, but rather are used to generate directives for each agent so that its exploration is both more informed and more coordinated with other agents. To our knowledge, MASPA is the first framework that surrounds and coordinates multi-agent learning with organizational control. It has a hierarchy of control and data abstraction, which is conceptually different from existing hierarchical multi-agent learning algorithms that uses a hierarchy of task abstraction. In addition, MASPA can be used together with approaches that reduces the policy search space to further speed up the learning.

As other approaches to improving MARL algorithms, the use of MASPA requires some additional knowledge. This knowledge is used to decide what organizational structure needs to be formed, what abstracted state information is useful, and how to convert this information into supervisory information. However, MASPA itself is a general framework that dynamically guides the learning of agents. We verified the generality of MASPA with its applications in different domains (distributed task allocation and network routing) with different MARL algorithms. Experimental results show that it not only dramatically speeds up the rate of MARL convergence, but also increases its likelihood of convergence.

MASPA assumes agents will voluntarily share their state information. It also implicitly assumes the original multi-agent system can be formed into a nearly decomposable hierarchy [9] of at least one level. This assumption implies that if agents in the original MAS are far apart in spatial terms, their behaviors are also far apart in causal terms. For example, in Figure 1, knowing detailed information about agents in cluster 6 will not significantly affect the behaviors of agents in cluster 1. Our assumptions hold in many real cooperative systems. Sensor network is one example, where the whole system is designed to cooperate and usually decomposable according to proximity. Other examples include package routing in the Internet, peer-to-peer file sharing or information retrieval, and resource sharing in grid computing.

To focus on the essence of MASPA coordinating multiagent learning and isolate its impact on the system performance, this paper uses pre-defined supervision organization structures. Supervision organizations can be dynamically formed during the learning through a bottom-up self-organization approach [14]. For simplicity, this paper limits the discussion to the case where learning only happens at the bottom level and supervising agents use pre-specified heuristics to make decisions, but, in principle, MASPA does not restrict supervising agents learning their supervision policies.

The rest of the paper is organized as follows: Section 2 presents a multi-level organizational structure for automated supervision mechanism. Section 3 defines a communication protocol for agents at different levels. Section 4 describe the supervisory policy adaptation that integrates supervisory information into MARL algorithms. Section 5 empirically evaluates our framework on distributed task allocation problem and network routing. Finally, Section 6 concludes this work and discusses some future work.

2. ORGANIZATIONAL SUPERVISION

Supervision mechanisms commonly exist in human organizations, such as enterprises and governments. The purpose of these mechanisms is to run an organization effectively and efficiently to fulfill the organization goals. Supervision involves gathering information, making decisions, and providing directions to regulate and coordinate actions of organization members. The practical effectiveness of supervision mechanisms in human organizations, especially in large organizations, inspired us to introduce a similar mechanism into multi-agent systems in order to improve the efficiency of MARL algorithms.

To add a supervision mechanism to a MAS with an overlay structure, MASPA adopts a multi-level, clustered organizational structure. Agents in the original overlay network, called workers, are clustered based on some measure (e.g., geographical distance). Each cluster is supervised by one agent, called the supervisor, and its member agents are called subordinates (note that subordinates at the lowest level are workers). The supervisor role can be played by a dedicated agent or one of the workers. If the number of supervisors is large, a group of higher-level supervisors can be added, and so on, forming a multi-level supervision structure.[1] In this paper, our discussion focuses on the situation where each agent belongs to only one cluster.

Two supervisors at the same level are adjacent if and only if at least one subordinate of one supervisor is adjacent to at least one subordinate of the other. Communication links, which can be physical or logical, exist between adjacent workers, between adjacent supervisors, and between subordinates and their supervisors. Figure 1 shows a three-level organizational structure. The bottom level is the overlay network of workers which forms 9 clusters. A shaded circle represents a supervisor, which is responsible for a corresponding cluster. Note that links between subordinates and

[1]The top supervision level can have multiple supervisors.

their supervisors are omitted in this figure.

3. COMMUNICATION PROTOCOL

Each agent can demonstrate both fast and slow dynamics in how its features change. Fast dynamics of an agent are exhibited by the changes of such features as those that represent interactions with other agents, its local state, and its policy (or value function). Slow dynamics are exhibited by the changes of an agent's *abstracted state*. The abstract state is defined by a vector of features, which can be projected from features with fast dynamics by using such techniques as:

- Using partial components of a feature and ignoring other components that do not affect slow dynamics
- Using some statistics (e.g., mean, mode, etc.) of a feature generated over the temporal or spatial scale
- Replacing a fast-changing feature with its distribution parameters if its changes follow some statistical distribution

Similarly, each cluster also has fast and slow dynamics. Fast dynamics of a cluster are exhibited by that of its members. Slow dynamics of a cluster are captured by the changes of its supervisor's abstracted state. The abstracted state of a supervisor is projected either from the abstracted states of its subordinates or directly from features with fast dynamics of its subordinates. MASPA assumes that a supervisor can make rational decisions based on its own and neighbors' abstracted states.

MASPA uses three types of communication messages: *report*, *suggestion*, and *rule*. A report is used by a subordinate to pass its abstracted state upwards to provide its supervisor with a broader view. A supervisor also sends its report to its adjacent supervisors at the same level in addition to its immediate supervisor (if any). The supervisor's view is based on not only the agents that it supervises (directly or indirectly) but also its neighboring supervisors. This peer-supervisor communication allows each supervisor to make rational local decisions when directions from its immediate supervisor are unavailable.

Based upon this information, a supervisor employs its expertise, integrates directions from its superordinate supervisor, and provides supervisory information to its subordinates. Rules and suggestions are used to transmit supervisory information. We define a *rule* 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 $[-1, 1]$

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 impact of the suggestion on the supervised agent.

Each rule (or suggestion) contains a condition specifying states where it can be applied. Subordinates are required to obey rules from their supervisors. Rules are "hard" constraints on subordinates' behavior. In contrast, suggestions are "soft" constraints and allow a supervisor to express its preference for subordinates' behavior. A supervisor has a more global view but may lack detailed information about its subordinates' local policies and its own surrounding environment. Using suggestions, the supervisor is able to affect a subordinate's policy yet allow the subordinate to override its directives when needed. The implicit assumption is that a supervisor's suggestions will be correct most of the time so that the penalty of bad suggestions is outweighed by good suggestions. Therefore, a subordinate does not rigidly adopt suggestions. The effect of a suggestion on a subordinate's local decision making may vary, depending on its current policy and state. A supervisor will refine or cancel rules and suggestions as new or updated information becomes available.

A set of rules are in conflict if they forbid all possible actions on some state(s). Two suggestions are in conflict if one is positive and the other is negative and they share some state(s) and action(s). A rule conflicts with a suggestion if a state-action pair is forbidden by the rule but is encouraged by the suggestion. In our supervision mechanism, we assume each supervisor is rational and will not generate rules and suggestions that are in conflict. However, in a multi-level supervision structure, a supervisor's local decision may conflict with its superordinate (the supervisor's supervisor) direction. Rules have higher priority than suggestions. There are several strategies for resolving conflicts between rules or between suggestions, such as always taking its superordinate or local rule, stochastically selecting a rule, or requesting additional information to make a decision. The strategy choice depends on the application domain. Note that it may not always be wise to select the superordinate decision, because, although the superordinate supervisor has a broader view, its decision is based on abstracted information. The strategy used here for resolving conflicts picks the most constraining rule and combines suggestions by summing the degrees of the strongest positive suggestion and the strongest negative suggestion.

4. SUPERVISORY POLICY ADAPTATION

Using MARL, each agent gradually improves its action policy as it interacts with other agents and the environment. A *pure* policy deterministically chooses one action for each state. A *mixed or stochastic* policy specifies a probability distribution over the available actions for each state. A policy can be represented as a function $\pi(s, a)$, which specifies the probability that an agent will execute action a at state s. As argued in [10], mixed policies can work better than pure policies in partially observable environments, if both are limited to act based on the current percept. Due to partial observability, most MARL algorithms are designed to learn mixed policies. The rest of this section shows how mixed policy MARL algorithms can take advantage of higher-level information specified by rules and suggestions to speed up convergence.

As shown in Figure 2 (a), a typical MARL algorithm contains two components: policy (or action-value function) update and action selection based on the learned policy. One common method to speed up learning is to supply an agent with additional reward to encourage some particular actions, which is called reward shaping [6]. This use of the special reward affects both policy update and action selection. In a single-agent setting, there are potential function forms of reward shaping that leave the optimal policy/value-function unchanged [6]. However, due to the non-stationary learning environment in a multi-agent setting, reward shaping may generate a policy that is undesirable in that they may distract from the main goal, which is supported by the normal reward.

MASPA directly biases the action selection for exploration without changing the policy update process. As shown in Figure 2 (b), MASPA' supervisory policy adaptation integrates rules and sugges-

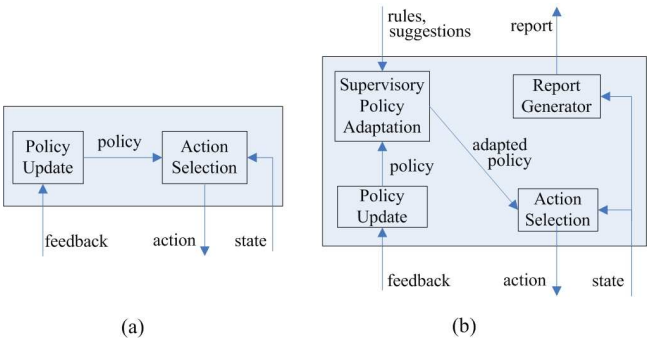

Figure 2: Unsupervised MARL vs. Supervised MARL with MASPA

tions into the policy learned by an unsupervised MARL algorithm and then outputs an adapted policy. This adapted policy is intended to control exploration. Our integration assumes policies learned by an unsupervised MARL are stochastic. The report generator computes the abstract state of the agent.

Let R and G be the rule set and suggestion set, respectively, that a worker received and π be its learned policy. We define $R(s,a) = \{\langle c, F\rangle \in R|$ state s satisfies the condition c and $a \in F\}$ and $G(s,a) = \{\langle c, A, d\rangle \in G|$ state s satisfies the condition c and $a \in A\}$. As we assume a supervisor is rational, it will not generate more than one suggestion for a subordinate that satisfies a state-action pair. Thus, $|G(s,a)| \leq 1$. The function $deg(s,a)$ that returns the degree of the satisfied suggestion is defined as following:

$$deg(s,a) = \begin{cases} 0 & \text{if } |G(s,a)| = 0 \\ d & \text{if } |G(s,a)| = 1 \text{ and } \langle c, A, d\rangle \in G(s,a) \end{cases}$$

Then the adapted policy π^A for the action selection is generated by the supervisory policy adaptation:

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

The state-dependent function $\eta(s)$ ranges from $[0,1]$. As similarly defined in the supervised actor-critic architecture [8], it determines the receptivity for suggestions and allows the agent to selectively accept suggestions based on its current state. For instance, if an agent becomes more confident in the effectiveness of its local policy on state s because it has more experience with it, then $\eta(s)$ decreases as learning progresses. In our experiments, we set $\eta(s) = k/(k + visits(s))$ where k is a constant and $visits(s)$ returns the number of visits on the state s.

With the supervisory policy adaptation, a rule explicitly specifies undesirable actions for some states and is used to prune the state-action space. Suggestions, on the other hand, are used to bias agent exploration. To integrate suggestions into MARL, MASPA uses the strategy that the lower the probability of a state-action pair, the greater the effect a positive suggestion has on the pair and the less the effect a negative suggestion has on it. The underlying idea is intuitive. If the agent's local policy already agrees with the supervisor's suggestions, as indicated by the policy having high (or low) probabilities for state-action pairs from the positive (or negative) suggestions, it is going to change its local policy very little (if at all); otherwise, the agent follows the supervisor's suggestions and makes a more significant change to its local policy.

To normalize π^A such that it sums to 1 for each state, the *limit* function from GIGA [16] is applied with minor modifications so that every action is explored with minimum probability ϵ:

$$\pi^A = limit(\pi^A) = argmin_{x:valid(x)}|\pi^A - x|$$

i.e., $limit(\pi^A)$ returns a valid policy that is closest to π^A.

Our normalization also implicitly solves the issue of rules in conflict. If a set of rules forbids all actions on a state, then the probability of each action is set to 0. After normalization, the probabilities of all actions are equal, that is, the action choice becomes completely random. This strategy is reasonable when the agent does not know the consequence of violating each rule.

5. EXPERIMENTAL RESULTS

We have tested MASPA in two different domains: distributed task allocation problem (DTAP) and network routing. In the following experiments, we manually cluster agents in the overlay network using Manhattan distance. The agent closest to the center of each cluster is elected as the supervisor. Supervisors also play the worker role. We assume there are links that allows direct communication between subordinates and their supervisors and between adjacent supervisors.

5.1 Distributed Task Allocation

We evaluated MASPA in a simplified DTAP [2] with Poisson task arrival and exponential service time. Agents are organized in an overlay network. Each agent receives tasks from the environment at a certain rate. 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 neighboring agent for processing. A task to be executed locally will be added to the local queue with unlimited queue length, where tasks are executed on a first-come-first-serve basis. Agents interact via communication messages and communication delay between two agents is proportional to the distance between them, one time unit per distance unit. The main goal of DTAP is to minimize the total service time of all tasks, averaged by the number of tasks, $ATST = \frac{\sum_{T \in \bar{T}_\tau} TST(T)}{|\bar{T}_\tau|}$, where $\bar{T}_\tau$ is the set of tasks received during a time period τ and $TST(T)$ is the total service time that task T spends in the system, which includes the routing time, queuing time, and execution time.

5.1.1 Experimental Setup

We chose one representative MARL algorithm, the Weighted Policy Learner (WPL) algorithm [1], for each worker to learn task allocation policies, and compared its performance with and without MASPA. WPL is a gradient ascent algorithm where the gradient is weighted by $\pi(a)$ if it is negative; otherwise, it will weighted by $(1 - \pi(a))$. So effectively, the probability of choosing a good action increases by a rate that decreases when the probability approaches to 1. Similarly, the probability of choosing a bad action decreases by a rate that decreases when the probability approaches to 0. A worker's state is defined by the current work load (or total work units) in the local queue.

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 gener-

ates 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.

Three measurements are evaluated: the average total service time (ATST), the average number of messages (AMSG) per task, and the time of convergence (TOC). ATST indicates the overall system performance, which can reflect the effectiveness of learning and supervision mechanism and can also be used to verify system stability (convergence) by showing a monotonic decrease in ATST as agents gain more experiences. AMSG shows the overall communication overhead for finishing one task, which including both for task routing and MASPA supervision. To calculate TOC, we take sequential ATST values with certain size and then calculate the ratio of those values' deviation to their mean. If the ratio is less than a threshold (e.g., 0.025), then we consider the system stable. TOC is the start time of the selected points.

Experiments were conducted using uniform two-dimension grid networks of agents with different sizes: 6x6, 10x10, and 27x27, all of which show similar results. But as the size of the system increases, the MASPA impact on the system performance becomes greater. For brevity, we only present here the results for the 27x27 grid (with 729 agents). For simplicity, we assume that all agents have the same execution rate and that tasks are not decomposable. The mean of task service time is $\mu = 10$. We tested three patterns of task arrival rates:

Uneven Center Load where 121 agents in the centric 11x11 grid receive tasks and other agents receive no tasks from the external environment. In the centric 11x11 grid, the task arrival rate of agents on the outermost 6 columns is $\lambda = 0.8$ and the rate of the rest agents is $\lambda = 0.2$.

Corner Load where only agents in the 12x12 grid at the up-left corner receive tasks from the external environment. In that 12x12 grid, the agents in the 9x9 grid at the up-left corner has the task arrive rate $\lambda = 0.2$ and the rest agents has the rate $\lambda = 0.7$.

Boundary Load where the 200 outermost agents receive tasks with rate $\lambda = 0.33$ and other agents receive no tasks from the external environment.

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. All agents use WPL with learning rate 0.001. Our experiments use the parameter $\eta(s) = 1000/(1000 + visits(s))$.

We compared four structures: *no supervision, local supervision, one-level supervision, and two-level supervision*. In the *local supervision* structure, agents are their own supervisors. With this structure, each agent gains a view only about itself and its neighbors, which is not much different from its view in the organization without supervision. We use the *local supervision* structure to evaluate whether domain knowledge combined with a limited view, which is used to create rules and suggestions, still improves the system performance. In contrast, the performance of the two following structures with supervision show the benefits of having a broader view combined with domain knowledge. The *one-level supervision* structure has 81 clusters, each of which is a 3x3 grid and the agent at each cluster center is elected as the supervisor. The *two-level supervision* structure forms from the *one-level supervision* structure by grouping 81 supervisors into 9 clusters, each of which is a 3x3 grid. The supervision structures with three or more levels did not show further improvement over the two-level supervision in our DTAP experiments. This is because a wide-range task transfer causes a long routing time which offsets the reduction of the queuing time in each agent.

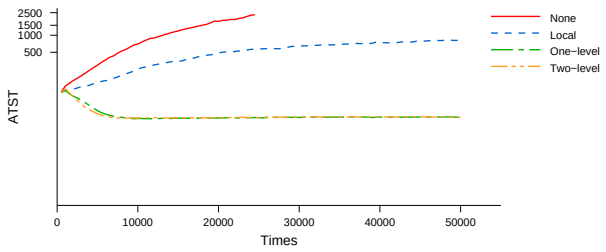

Figure 3: ATST for different structures with uneven center load

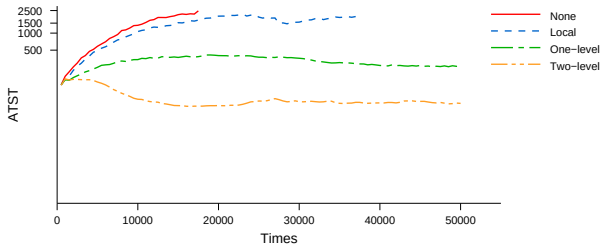

Figure 4: ATST for different structures with corner load

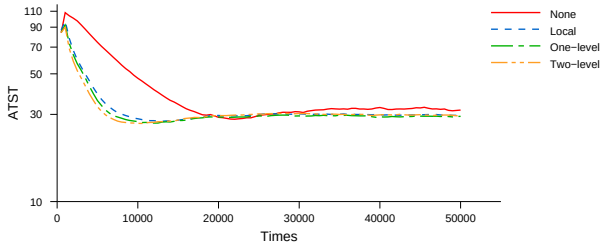

Figure 5: ATST for different structures with boundary load

5.1.2 *Results & Discussions*

Figure 3, 4 and 5 plot the trend of ATST, as agents learn, for different organization structures with different task arrival patterns. Note that the y axis in the plots is logarithmic. As expected, MASPA improves both the likelihood and speed of the learning convergence. The broader the view MASPA observes, the greater the system performance it improves. In addition, several other observations are also noted.

Under both uneven center load and corner load, the system without MASPA does not seem to converge. From Figure 3 and 4, we see that both simulations ends before 50000 time units. This

happens because, using random exploration, agents in the grid do not learn and propagate quickly enough knowledge about where light-loaded agents are. As a result, for example, under uneven center load patter, more and more tasks loop and reside in the center 11x11 grid where agents receive external tasks. This makes the system load severely unbalanced and the system capability not well utilized, which causes the system load to monotonically increase. Our simulations ran out of all computing resources and terminated before showing any signs of convergence. In contrast, observing broader views, MASPA guides and coordinates the exploration of agents and allows them to learn quickly to effectively route tasks.

Under both uneven center load and corner load, *local supervision* does not prevent system divergence. This is because uneven task arrival rates in both patterns cause many agent's local view of the system to become inconsistent with the global system view. For example, under uneven center load pattern, many overloaded agents at the center columns find their neighbors having even higher loads. As a result, *local supervision* generates incorrect directives for them to explore their actions. For similar reasons, explained at a cluster level instead of a worker level, the system with *one-level supervision* doesn't perform well under corner load pattern.

Broader views for MASPA do not necessarily significantly improve the system performance. For example, under uneven center load, *one-level supervision* and *two-level supervision* show similar performance, and, under boundary load pattern, all supervision structures demonstrate similar performance. This is because, in both cases, broader views do not provide much additional information for MASPA. For example, under the boundary load pattern, local work loads in the whole network quickly form some pattern, where an agent farther away from the network boundary usually has a lighter local load. Then, based on their local view, most agents generate suggestions for themselves to forward tasks to neighbors closer to the network center, which are coincidentally similar to suggestions generated from a broader view (e.g., one-level or two-level supervision).

Supervision	ATST	AMSG	TOC
None	N/A	N/A	N/A
Local	N/A	N/A	N/A
One-level	33.41 ± 0.66	10.21 ± 0.25	7500
Two-level	34.08 ± 0.62	10.60 ± 0.22	6000

Table 1: Performance of different structures with uneven center load

Supervision	ATST	AMSG	TOC
None	N/A	N/A	N/A
Local	N/A	N/A	N/A
One-level	265.50 ± 6.59	24.83 ± 1.34	38500
Two-level	51.37 ± 0.88	16.33 ± 0.26	14000

Table 2: Performance of different structures with corner load

Table 1, Table 2, and Table 3 show the different measures for each supervision structure at their own convergence time point. In addition to increasing the convergence rate, MASPA also decreases the system ATST. In most cases, the broader the views MASPA observes, the lower the ATST the system generates. We can also observe that MASPA does not incur heavy communication overhead. For example, with the boundary load pattern, *one-level supervision* has less than 0.6% communication overhead. With the corner load pattern, *two-level supervision* actually produces lower AMSG than *one-level supervision*. This is because *two-level supervision* leads workers to learn more quickly and effectively to forward tasks to the right workers, which dramatically reduces the number of messages for routing tasks and offsets the overhead from an additional level of supervision.

Supervision	ATST	AMSG	TOC
None	29.26 ± 0.71	6.90 ± 0.21	17500
Local	28.21 ± 0.59	7.02 ± 0.09	8500
One-level	27.64 ± 0.60	6.94 ± 0.16	7500
Two-level	27.49 ± 0.60	7.14 ± 0.14	6500

Table 3: Performance of different structures with boundary load

During the experiments, we observed that supervisory information corresponding to coarse-grained control tend to be more helpful than that corresponding to fine-grained control in improving the system performance. Moreover, fine-grained may even decrease system performance. Coarse-grained control considers and operates on the whole cluster as one entity, while fine-grained control operates on individual cluster members. "Moving more tasks from my cluster to one of neighboring clusters" and "balancing the load within the cluster" are examples of coarse-grained control . "Moving more tasks from a high-loaded agent to a low-loaded agent along the shortest path" is an example of fine-grained control. One explanation for this observation is that supervisory information corresponding to coarse-grained control results in more coordination among agents' exploration, speeding up the learning convergence. In contrast, in our simulation, due to lack of detailed information of each cluster member, fine-grained control for some individual members is not able to fully evaluate the impact on and from other agents. As a result, the fine-grained control may interfere with the normal learning process of other agents and the dynamics of other agents may degrade the fine-grained control.

We have explored different values of cluster size and found that system performance decreases with cluster size that are either too small (e.g., ≤ 5) or too large (e.g., ≥ 100). This is because, with too small a cluster size, supervisors do not collect enough information to create correct rules and suggestions. When a cluster size is too large, the representation of cluster abstracted states for DTAP (i.e. averaging loads of subordinates) ignores the variance among subordinates. As a result, supervisors are not able to create proper rules and suggestions for every subordinate. Therefore, there is a trade-off for the cluster size. In addition, cluster sizes that produce the best performance vary in different environments (e.g., different task arrival patterns).

Similarly, there is a trade-off in the length of the report period. A too short report period causes a large variance of the abstracted state (also increases communication overhead) and results in oscillating suggestions and rules. A too long report period causes the supervisory information received by workers to be out-dated and as a result, decreases the convergence rate.

5.2 Network Routing

We also evaluated our framework using a network routing simulator adopted from Boyan and Littman [4]. It is a discrete time simulator of communication networks with various topologies. A communication network consists of a homogeneous set of nodes (or agents) and links between them. Packets are periodically introduced into the network under a Poisson distribution with a random origin and destination. No two packets have the same agent as their origin and destination. When a packet arrives at an agent, the agent

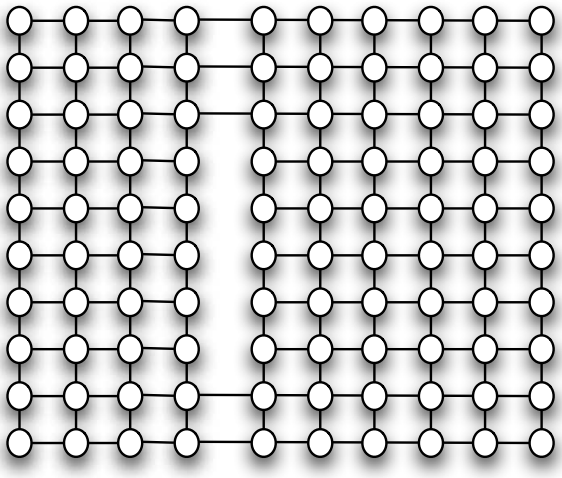

Figure 6: The 10 x 10 grid topology

puts it into the local FIFO (first in first out) queue. At each time step, an agent makes its routing decision to forward the top packet in the queue to one of its neighbors. Once a packet reaches its destination, it is removed from the network. In our experiments, we set the time cost of sending a packet down a link as a unit cost. So the delivery time of packet consists of its transmission cost and its waiting time in queues. The main goal of a network routing algorithm for this problem is to minimize the Average Delivery Time (ADT) of all packets.

5.2.1 *Experimental Setup*

Each agent uses a Policy Gradient Descent (PGD) algorithm to learn its routing policies. The PGD algorithm is a variant of the GIGA algorithm [16], which minimizes the total discounted cost and approximates the policy gradient of each state-action pair with the normalized difference of its Q-value and the expected Q-value on that state. PGD learns stochastic policies, but, unlike multi-agent OLPOMDP [13] and GAPS [7] that were also applied to network routing problem, it does not require a global reward signal. The state s is defined by the destination of the packet that an agent is forwarding. We define $Q_x(s,a)$ as the estimated time that an agent x takes to deliver a packet to the destination s through its neighbor a, including any time that the packet would have to spend in the agent x's queue. The "cost signal" $r(s,a)$ for forwarding a packet with destination s to its neighbor a is $q_a + w + t$, where w is the waiting time of the packet in x's queue and t is the transmission time between agent x and a. The Q-learning algorithm is used to update x's estimates.

The MASPA implementation in network routing is similar to that in DTAP. The main difference is the way that MASPA messages are generated. In the network routing problem, we do not use rules. The abstracted state of a worker (or supervisor) is defined a vector $\langle t_1, t_2, \ldots, t_m \rangle$, where t_i is the average estimated time that the worker (or the supervisor's cluster) takes to deliver a packet to destination agents in cluster i. So, by using statistic *mean*, the abstract state of a worker can be computed from its Q-value table and a supervisor's abstracted state can be projected from its subordinates' abstracted states. A simple heuristic is used for generating suggestions. A supervisor always produces positive (or negative) suggestions for its subordinates to encourage (or discourage) them

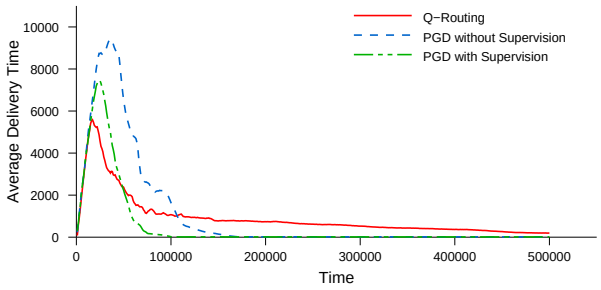

Figure 7: Performance under network load = 7.0

forwarding packets to clusters with lower (or higher) estimated delivery time to some destination cluster. The suggestion degree for each subordinate depends on the difference between the average estimated delivery time of neighboring clusters and the distance of the subordinate to the boundary.

We have tested the PGD algorithm with and without MASPA on several network topologies with various number of nodes, all of which show similar results. For brevity, we concentrate on the result analysis for the 10 x 10 grid network pictured in Figure 6. The Q-routing [4] algorithm is used as baseline, which learns deterministic policies. Two measurements are evaluated: the average delivery time (ADT) and the time of convergence (TOC). The ADT is computed every 1000 time units. To calculate TOC, we take 50 sequential ADT values and then calculate the ratio of those values' deviation to their mean. If their mean is less than the maximum expected ADT (we use 300) and the ratio is less than a threshold (we use 0.05), then we consider the system stable. TOC is the start time of the selected points.

Results are then averaged over 10 simulation runs. All agents use the PGD algorithm with a learning rate $\zeta = 0.1$. Workers send reports to their supervisors every 500 time units. Our experiments use the parameter $\eta(s) = 20000/(20000 + visits(s))$.

5.2.2 *Results & Discussions*

Figure 7 shows the performance trend as agents learn under network load= 7.0. The network load is the average number of packages entering the network at each time unit. All three algorithms, after initial periods of inefficiency during which they randomly explore the environment, gradually improve their performance and stabilize. At the very early period, MASPA does not improve the performance much. This is because, due to almost complete random exploration, subordinates do not provide accurate environment information to their supervisor, which may result in some improper suggestions. As information accuracy increases, MASPA properly biases the policy search of the PGD algorithm and speeds up the convergence. Due to policy oscillation, Q-routing shows slow convergence.

Figure 8 shows the TOC of three algorithms under various network loads. As expected, MASPA consistently speeds up the convergence of the PGD algorithm. The higher the network load, the greater the speed improvement. For example, when load ≥ 5.5, MASPA decreases the TOC by around 40% or more. Under low network loads, optimal policies usually follows shortest paths, so they are deterministic. The PGD algorithms use gradient update and gradually converge to deterministic policies, slower than Q-routing that directly learns deterministic policies. However, under high loads, where optimal policies are usually stochastic, the Q-routing policies show oscillation during the learning and the PGD

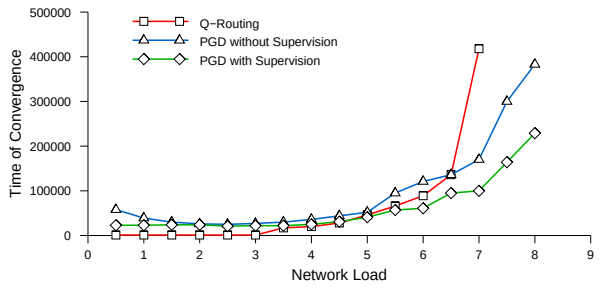

Figure 8: Time of Convergence at various loads

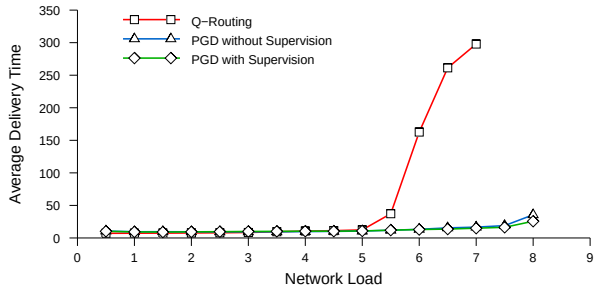

Figure 9: Delivery time at various loads

algorithm with MASPA converges faster to stochastic policies.

Figure 9 shows the ADT at the convergence time point under various network loads. Under low loads, as both PGD algorithms, with and without MASPA, converge to deterministic policies, they show almost the same performance. Due to random exploration with some probability, they perform slightly worse than Q-routing. However, under high loads, MASPA improves the PGD performance. For example, when load ≥ 6.5, MASPA decreases the ADT by at least 10%, and when load $= 8.0$, MASPA reduces the ADT by around 30%. As both PGD algorithms converge to stochastic policies, which allows agents to simultaneously exploit multiple paths to deliver packets to a single destination, they perform much better than Q-routing under high loads.

6. CONCLUSION

This work presents MASPA, a decentralized supervision framework, that enables efficient learning in large-scale multi-agent systems. In MASPA, the automated supervision mechanism fuses activity information of lower-level agents and generates supervisory information that guides and coordinates agents' learning process. This supervision mechanism continuously interacts with and dynamically controls the learning process. Simulation results obtained in two different domains with different MARL algorithms verify the generality of MASPA and demonstrate that MASPA significantly accelerates the learning process with relatively low communication overhead.

As mentioned, the use of MASPA may require some additional knowledge. Future work includes developing techniques and algorithms to relax these requirements. We are developing a distributed algorithm for dynamically forming supervision organizations (addressing agent clustering and supervisor election) [14]. We are also interested in developing techniques for automatically learning projection functions to construct abstracted states, and investigating learning algorithms for supervisors to automate the generation of rules and suggestions.

7. REFERENCES

[1] S. Abdallah and V. Lesser. Learning the task allocation game. In *AAMAS'06*, 2006.

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

[3] R. A. C. Bianchi, C. H. C. Ribeiro, and A. H. R. Costa. Heuristic selection of actions in multiagent reinforcement learning. In *IJCAI'07*, Hyderabad, India, 2007.

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

[5] R. Makar, S. Mahadevan, and M. Ghavamzadeh. Hierarchical multi-agent reinforcement learning. In *Autonomous Agents'01*, pages 246–253, 2001.

[6] A. Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: theory and application to reward shaping. In *ICML'99*, pages 278–287, 1999.

[7] L. Peshkin and V. Savova. Reinforcement learning for adaptive routing. In *International Joint Conference on Neural Networks (IJCNN)*, 2002.

[8] M. T. Rosenstein and A. G. Barto. Supervised actor-critic reinforcement learning. In J. Si, A. Barto, W. Powell, and D. Wunsch, editors, *Learning and Approximate Dynamic Programming: Scaling Up to the Real World*, pages 359–380. John Wiley and Sons, 2004.

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

[10] S. P. Singh, T. Jaakkola, M. L. Littman, and C. Szepesvari. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, 38(3):287–308, 2000.

[11] P. Stone and M. Veloso. Team-partitioned, opaque-transition reinforcement learning. In *Autonomous Agents'99*, pages 206–212, 1999.

[12] P. Tangamchit, J. Dolan, and P. Khosla. Learning-based task allocation in decentralized multirobot systems. In *DARS'00*, pages 381–390, 2000.

[13] N. Tao, J. Baxter, and L. Weaver. A multi-agent policy-gradient approach to network routing. In *ICML '01*, pages 553–560, 2001.

[14] C. Zhang, V. Lesser, and S. Abdallah. Self-organization for dynamically supervising distributed learning. In *University of Massachusetts Amherst Computer Science Technical Report UM-CS-2009-007*, 2009.

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

[16] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *ICML'03*, pages 928–936, 2003.