Combining Dynamic Reward Shaping and Action Shaping for Coordinating Multi-Agent Learning

Xiangbin Zhu
College of Mathematics, Physics and Information Engineering
Zhejiang Normal University,
Jinhua Zhejiang, 321004, China
zhuxb@zjnu.cn

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

***Abstract*—Coordinating multi-agent reinforcement learning provides a promising approach to scaling learning in large cooperative multi-agent systems. It allows agents to learn local decision policies based on their local observations and rewards, and, meanwhile, coordinates agents' learning processes to ensure the global learning performance. One key question is that how coordination mechanisms impact learning algorithms so that agents' learning processes are guided and coordinated. This paper presents a new shaping approach that effectively integrates coordination mechanisms into local learning processes. This shaping approach uses two agents levels and combines reward shaping and action shaping. The higher-level agents dynamically and periodically produce the shaping heuristic knowledge based on the learning status of lower-level agents. The lower-level agents then uses this knowledge to coordinate their local multi-agent learning processes. Experimental results show our approach effectively speeds up the convergence of multi-agent learning in large systems.**

***Index Terms*—Multi-agent Learning; Organization Control; Supervision; Reward Shaping; Action Shaping**

I. Introduction

A central question in developing cooperative multi-agent systems is to design distributed coordination policies for agents so that they work together to optimize the global performance. Multi-agent reinforcement learning (MARL) provides an attractive approach to this question. MARL allows agents to explore environment through trial and error, adapt their behaviors to the dynamics of the uncertain and evolving environment, and gradually improve their performance through experiences.

One of key research challenges is to scale MARL to large cooperative systems. Coordinating MARL [7, 8, 13, 16, 15] provides a promising direction to address this challenge. Using coordinating MARL, agents learn their policies based on their local observations and interactions, while their learning processes are coordinated and guided by exploiting non-local information to improve the overall learning performance. One important problem of coordinating MARL is how agents' learning processes need to be modified in order to integrate non-local knowledge. Existing approaches for coordinating MARL use a technique, called *action shaping*, i.e., biasing action selection by directly manipulating learned policies [7, 8, 13, 16, 15]. Action shaping can prohibit an agent from taking some actions in specified states, and can encourage or discourage an agent to take some actions in specified states. Action shaping is immediately effective on the specified states, but only limited to these specified states. However, it is difficult and complex to use action shaping to exploit common situations where neighboring states of "bad" states (i.e., with low expected rewards) are more likely bad and neighboring states of "good" states are more likely good.

In this paper, we demonstrate that an alternative technique, called *reward shaping*, can potentially address this issue in coordinating MARL. Reward shaping [2, 10, 11] has been extensively studied for single agent reinforcement learning. It exploits heuristic knowledge by providing an agent with additional reward signals to accelerate its learning process. By utilizing the backup operation of reinforcement learning (updating the value of a state using values of future states), reward shaping implicitly exploits situations where neighboring states of "bad" states (i.e., with low expected rewards) are more likely bad and neighboring states of "good" states are more likely good. Unlike other work on multi-agent reward shaping [3, 6, 4, 5], our reward shaping approach dynamically generates additional reward signals for agents based on their current learning status and is used to coordinate agents' learning processes. Moreover, this paper illustrates that reward shaping and action shaping are complementary to each other and combining them for coordinating MARL can further improve the learning performance.

In this paper, we illustrate our approach using a coordinating MARL framework [13] (see Figure 1 and 5), called Multi-Agent Supervisory Policy Adaptation (MASPA). This framework employs low-overhead, periodic organizational control to coordinate multi-agent reinforcement learning to ensure the global learning performance. MASPA is general and extensible and can work with most existing MARL algorithms. MASPA provides an action shaping technique, which will be used as our evaluation baseline in a distributed task allocation application domain.

The rest of the paper is organized as follows: Section 2 introduces MASPA and its action shaping. Section 3 presents reward shaping in multi-agent learning. Section 4 discusses in detail the advantages and disadvantages of action shaping and reward shaping and how they are complementary to each other. Section 5 illustrates how to dynamically generate shaping

rewards in a distributed task allocation problem. Section 6 shows the empirical results and analyzes these results. Finally, Section 7 concludes the paper.

II. MASPA and Action Shaping

Many realistic settings have a large number of agents and communication delay between agents. To achieve scalability, each agent can only interact with its neighboring agents and has a limited and outdated view of the system (due to communication delay). In addition, using MARL, agents learn concurrently and the environment becomes non-stationary from the perspective of an individual agent. As shown in [13], MARL may converge slowly, converge to inferior equilibria, or even diverge in realistic settings. To address these issues, a supervision framework was proposed in [13]. This framework employed low-overhead, periodic organizational control to coordinate and guide agents' exploration during the learning process.

The supervisory organization has a multi-level structure. Each level is an overlay network. Agents are clustered and each cluster is supervised by one supervisor. Two supervisors are linked if their clusters are adjacent. Figure 1 shows a two-level organization, where the low-level is the network of learning agent and the high-level is the supervisor network.

The supervision process 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. As shown in Figure 1, 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.

A supervisor uses *rules* and *suggestions* to transmit its supervisory information to its subordinates. A rule is defined as a tuple $< c, F >$, 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 $< c, A, d >$, 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.

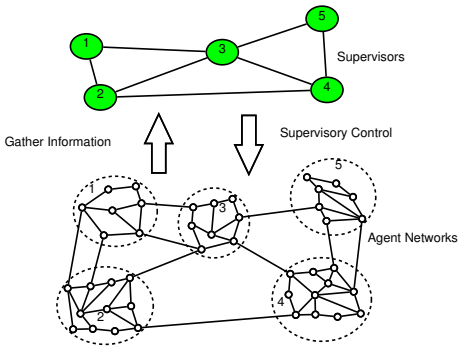

Fig. 1. The two-level hierarchical learning structure

Each learning agent integrates rules and suggestions into its policy which is learned by a local learning algorithm to generate an adapted exploration policy. Let *R* be the rule set. Let G be the suggestion set. $G(s,a)$=$\{< c, A, d >\in G|$state s satisfies the condition c and $a \in A\}$ is defined. The function $deg(s,a)$ returns the degree of suggestion, which is defined as following:

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

So the adapted policy π^A can be gotten as following:

$$\pi^A(s,a) = \begin{cases} 0 & \text{if } R(s,a) \neq \emptyset \\ \pi(s,a) + \pi(s,a) * \eta(s) & \\ \quad * deg(s,a) & \text{else if } deg(s,a) \leq 0 \\ \pi(s,a) + (1 - \pi(s,a)) * \eta(s) & \\ \quad * deg(s,a) & \text{else if } deg(s,a) > 0 \end{cases} \quad (2)$$

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] determines the suggestion receptivity. The $\eta(s)$ function decreases as learning progresses.

Because this type of integration method use supervisory information to directly bias the action selection for exploration without changing the policy update process, we refer to it as *action shaping*.

III. Reward Shaping in Multi-Agent Learning

An alternative form of integrating supervisory information into local learning processes could potentially involve *reward shaping*. Reward shaping has been shown to be a beneficial in single-agent reinforcement learning and in limited multi-agent settings [3, 9, 11]. In this section, in the context of our two-level coordination approach to MARL, we will present a reward shaping approach to integrating dynamic supervisory

information into local learning algorithm of multiple agents so that their learning processes are coordinated. In the later section, we will discuss how to periodically and dynamically compute appropriate shaping rewards.

MARL can use a model-free multi-agent learning method that is based on Q-learning, such as PGA-APP [14]. Therefore, the reward shaping technology of single-agent systems can be directly integrated in the MARL. The reward shaping of Q-learning is to provide an additional reward in order to accelerate the convergence of Q-learning [11, 12]. The one-step Q-learning with reward shaping is defined by [11]:

$$Q^{t+1}(s,a) \leftarrow (1-\alpha)Q^t(s,a) + \alpha[r(s,a) + F^t(s,a,s') + \gamma \max_{a'} Q^t(s',a')]$$

where $F^t(s,a,s')$ is the general form of the shaping reward and $r(s,a)$ is the immediate reward. The reward shaping presented here can be thought of as dynamic advice because it is generated online.

In the context of our two-level coordination approach to MARL, we can convert suggestions and rules to shaping rewards. So, we can use functions f_r and f_s for mapping rules and suggestions to reward respectively. So shaping reward can be described as follows:

$$F(s,a,t,s',a',t') = f_s(deg(s,a)) + f_r(R(s,a)) \quad (3)$$

Based on the equation (1), $f_s(deg(s,a))$ can be defined by:

$$f_s(deg(s,a)) = \begin{cases} r(s,a) * \eta(s) \\ \quad * \ deg(s,a) & \text{if } deg(s,a) \leq 0 \\ (1 - r(s,a)) * \eta(s) \\ \quad * \ deg(s,a) & \text{else if } deg(s,a) > 0 \end{cases} \quad (4)$$

where $r(s,a)$ is the immediate reward for the action a.

Let $r_{rule}(s,a)$ be the shaping reward that an agent receives for rules. For the state s and the action a, if there is a $R(s,a)$, the shaping reward $r_{rule}(s,a)$ can be defined as:

$$r_{rule}(s,a) = \begin{cases} \alpha r(s,a) & \text{if} \quad r(s,a) < 0 \\ -\alpha r(s,a) & \text{else} \end{cases} \quad (5)$$

where $r(s,a)$ is the immediate reward for the action a and α is an adjustment parameter.

Based on the equation (5), $f_r(R(s,a))$ can be defined by:

$$f_r(R(s,a)) = \begin{cases} 0 & \text{if } R(s,a) = \emptyset \\ r_{rule}(s,a) & \text{else} \end{cases} \quad (6)$$

IV. Combining Reward Shaping and Action Shaping

In this section, we show action shaping and reward shaping are complementary and the advantages of combining them for coordinating multi-agent learning.

Zhang et al. [13] have empirically verified that the action shaping method is effective for coordinating MARL. As mentioned early, the action shaping can accelerate the local Q-learning process via avoiding some bad actions or selecting some good actions. Thus, it can improve the system performance by directly changing the local policy. Rules are used to prune the state-action space. Suggestions bias an agent's exploration. But action shaping can not affect more states temporally and spatially than that of reward shaping.

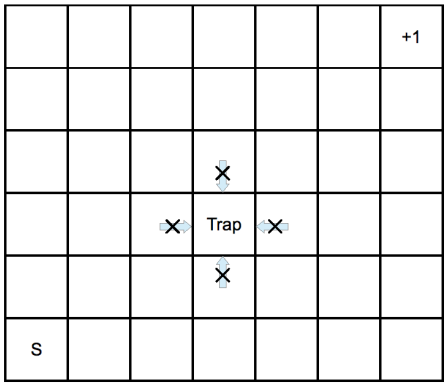

Fig. 2. The grid world with action shaping

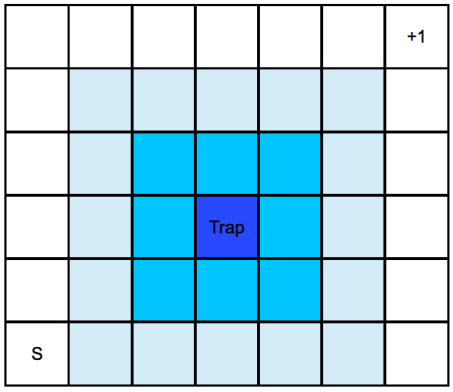

Fig. 3. The grid world with reward shaping

For example, consider a grid world, shown in Figure 2. This grid world has a start state denoted by 'S' and a goal state with reward '+1'. This grid world also contains a trap with reward -0.1. Each state has four actions: $Right, Left, Up$ and $Down$. The actions are stochastic motion. For example, if an agent takes the action Up, it will move up with probability 0.8, but with probability 0.1, it will move right, and with probability 0.1. it will move left. The goal for this grid world is to find an optimal policy for an agent to travel from the start state to the goal state. If we use action shaping, there will be some rules for neighboring states of the trap, which prohibit selecting the action that leads the agent to the trap with probability 0.8. But these rules do not change the reward of the trap. So action shaping almost doesn't change the Q-values of the neighboring states. It only cuts the trap from the state-action space. However, because the move is a stochastic move, the neighboring states of the trap are actually dangerous states. It is difficult to express these states with rules and suggestions. In contrast, by exploiting the backup operation of reinforcement learning, reward shaping can implicitly affect the Q-values of the neighboring states, which is showed in the Fig.3. This is because reward shaping makes the negative reward of the trap

to be greater and thus the agent will less likely explore the neighboring state of the trap.

The explanation above is from the spatial perspective. Suppose that the trap could be moving. After the trap has moved, the effect of reward shaping will be temporary and, in contrast, the effect of action shaping almost goes off instantly. So action shaping is more flexibility and reward shaping has longer-lasting effect. Thus, reward shaping and action shaping will bring different impacts from the temporal point. For example, in the distributed task allocation problem (DTAP) [1], action shaping will bring more benefits for adjusting the load balance in a cluster. The Fig.4 shows a simply state transition diagram for an agent in DTAP. An agent has five states based on its loads, e.g., s_0 indicating the lightest load and s_4 representing the heaviest load. If an agent takes the action a_0, the agent's load will increase. If an agent takes the action a_1, the agent's load will be reduced. At an early stage of the learning, the state s_2 may have a rule with regard to its queue length. So the Q-value of action a_0 in state s_2 will be reduced if we use reward shaping. But at the latter stage, the agent has a rule if its state is the state s_3. But the Q-value of action a_0 in state s_2 will still be low for some time. This situation will benefit the balance of queue lengths in a cluster. But if we only use action shaping, the agent will select the action a_0 with high probability in state s_2. So the state of the agent will change to state s_3 more quickly.

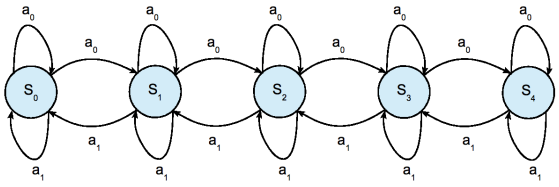

Fig. 4. The state transition diagram of DTAP

In general, action and reward shaping can both speed up the convergence of the MARL by making the exploration phase of reinforcement learning more effective. Nevertheless, at the beginning of learning, action shaping almost always provides better performance than that of reward shaping because action shaping guides immediately the exploration of MARL while reward shaping needs more time to improve Q-values. Therefore, combining action shaping and reward shaping will be a good idea.

The key issue for combining the two shaping is how to combine the two shaping in practice. Given the above reasons, not only the function $\eta(s)$ is used in action shaping of suggestions, but also it should be used in action shaping of rules. Intuitively, at the beginning, let action shaping take a leading role. Later, as the local policy has sufficiently learnt to be reasonable, the impact of action shaping should be decreased via $\eta(s)$. Therefore, we have a function $\eta(s)$ for rules and The adapted policy π^A can be changed as following:

$$\pi^A(s,a) = \begin{cases} (1-\eta(s)) * \pi(s,a) & \text{if } R(s,a) \neq \emptyset \\ \pi(s,a) + \pi(s,a) * \eta(s) & \\ \qquad * \, deg(s,a) & \text{else if } deg(s,a) \leq 0 \\ \pi(s,a) + (1-\pi(s,a)) * \eta(s) & \\ \qquad * \, deg(s,a) & \text{else if } deg(s,a) > 0 \end{cases} \tag{7}$$

where $\eta(s)$ is defined as following:

$$\eta(s) = k/(k + visit(s)) \tag{8}$$

where $visit(s)$ is the number of visiting the state and k is a constant.

V. Dynamically Computing Supervisory Information

One important issue of our two-level approach to MARL is how to generate supervisory information for action shaping and reward shaping in order to coordinate learning processes of multiple agents. In this section, we will first introduce the idea of a *cluster value*, which is periodically computed for each cluster to provide an evolving non-local view and then discuss how to dynamically compute supervisory information using the cluster value.

We use DTAP as a domain-dependent example. In DTAP, each agent receives tasks which arrive based on a Poisson distribution at a certain rate with exponential execution time. At each time, when an agent receives a task, the agent must make a decision whether it executes the task locally or transmits the task to one of its neighbors. So if an agent has 2 neighbors, it can choose one of three actions when it has received a task. As mentioned before, we have a hierarchical structure for multi-agent learning. Fig. 5 shows a 2-layer multi-agent system.

A. Cluster Value

The cluster value V_c is employed to evaluate how good a cluster has learned. A cluster evaluation function $C(z)$ is designed to compute the cluster value, where z is the argument vector with regards to a specific cluster. Let E be the set of all agents in a cluster. Let S_i be the state space of agent i. Let A_i be the action space of the agent i. $p_i(s)$ is the expected probability distribution of state s of agent i. Then, V_c can be calculated by the equation (9).

$$V_c = \sum_{i \in E} \sum_{s \in S_i} R_i(s) p_i(s) \tag{9}$$

where $R_i(s) = \sum_{a \in A_i} \pi_i(s,a) r_i(s,a)$, $\pi_i(s,a)$ is the policy value when the agent i selects the action a at the state s based on the policy π_i and $r_i(s,a)$ is the reward received when the agent i selects the action a at the state s. We will later discuss how this value can be approximated for a real application.

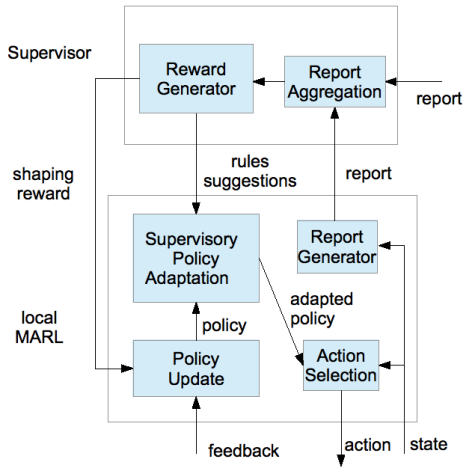

Fig. 5. Supervised MARL

B. Action shaping and Reward Shaping

After we get the cluster value and the policy, we can compute the shaping reward. Firstly, let d_v be the difference between two neighbor clusters' values, which express some measure of the difference between learning processes of these two clusters. The goal of the supervisory control implemented action and reward shaping is trying to increase the learning performance of the cluster with a lower cluster value without significantly affecting the learning performance of the cluster with a higher cluster value. To achieve this goal, we need address following questions:

- How to compute the cluster-level shaping reward with d_v, which can express the quantitative goodness of distribute reinforcement learning. The cluster-level shaping reward is called as $r_{suggestion}$, which is used for action shaping and reward shaping. So there is a function f_c for mapping d_v to $r_{suggestion}$. It is defined as follows:

$$r_{suggestion} = f_c(d_v) \tag{10}$$

- Another problem is how to convert $r_{suggestion}$ to the shaping reward of a specific agent, that is, how to map a non-local shaping reward into local shaping rewards.

We assume that there are two clusters: cluster c_1 and cluster c_2. Based on the policy of cluster c_1, cluster c_1 interacts with one of its neighboring clusters, which is c_2 for example. let V_{c1} and V_{c2} is the cluster values of cluster c_1 and its adjacent cluster c_2, respectively. So we have:

$$d_v = V_{c2} - V_{c1} \tag{11}$$

Based on the equation (9) and equation (11), the equation (10) can be changed as fellows:

$$r_{suggestion} = \alpha(V_{c2} - V_{c1}) \tag{12}$$

where α is an adjustment coefficient.

For DTAP, cluster value V_c is computed based on reports received from its cluster agents. In each report, the queue length of each agent is the important argument. The supervisor receives reports from its subordinates at fixed intervals. After getting all reports, the supervisor can calculate the average queue length of its cluster, which is also called the average load. Thus, V_c is equal to the average queue length.

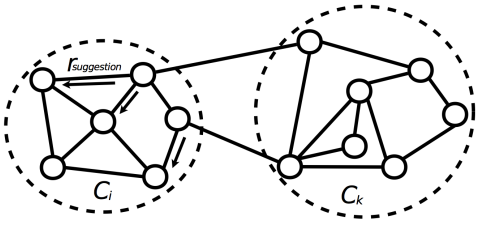

Fig. 6. Propagating shaping reward

Based on its cluster value, cluster c_i chooses one of its neighboring clusters, e.g., cluster c_k. Let V_{ci} and V_{ck} be the average load of cluster c_i and its adjacent cluster c_k respectively. So based on the equation (11), we have:

$$d_v = V_{ck} - V_{ci}$$

If $d_v < 0$, cluster c_i considers cluster c_k has a lower average load. Then, cluster c_i will encourage its members forwarding tasks to cluster c_k according to the following reward:

$$r_{suggestion} = -d_v/V_{ci}$$

The reward is a positive reward, which means to encourage forwarding tasks to cluster c_k.

If $d_v > 0$, cluster c_i considers cluster c_k has a higher average load. Cluster c_i encourages the subordinates to process its tasks by themselves. In other words, we discourage the subordinates to forward tasks to these neighboring agents which have a higher average load. Thus, the cluster-level shaping reward is given by:

$$r_{suggestion} = -d_v/V_{ck}$$

Therefore, the next step is how to convert $r_{suggestion}$ to the shaping reward for a specific agent. Our method is to transfer the cluster-level reward to subordinates adjacent to cluster c_k, which is showed in Fig. 6. For the $r_{suggestion}$ is based on average loads of clusters, we need more subordinates to encourage or discourage forwarding tasks to cluster c_k. The cluster-level reward will also be transferred to subordinates that are not on the boundary to other clusters. But the reward will attenuate for agents further away from the boundary. If a cluster-level reward $r_{suggestion}$ is transferred to subordinate j and its neighbor n doesn't receive the same shaping reward, then a shaping reward with value $\xi r_{suggestion}$ and action j will be transferred to n, where ξ is the reward decay rate. To reduce network overhead, if a shaping reward is less than a threshold(e.g., 0.05), it will not be transferred.

An agent may need to combine two shaping rewards. Let $R_{suggestion}$ be the shaping reward that an agent receives. Our combination strategy is showed as follows:

$$R_{suggestion} = \begin{cases} max(r_1, r_2) & \text{if} \quad r * r_2 > 0 \\ r_1 + r_2 & \text{else} \end{cases}$$

where r_1 and r_2 are two shaping rewards that are received by the agent. This strategy can be generalized to combine more than two shaping rewards.

Another source of shaping reward is from the rules in the supervisory information. Rules indicates that the agent should not choose some specific actions because the action will cause very bad performance. Our empirical results show the rules usually can have a large effect on learning performance because a rule can reduce the state-action space for local multi-agent reinforcement learning's exploration. Similarly, the reward shaping associated with rules has an important impact on learning performance.

For DTAP, when an agent has too long a queue, it should forward any tasks that it received to other agents. So we need a measure to judge whether an agent is overloaded. The measure is a rule, which is also called as load limit rule. The key idea is to get a limit L_{limit} for queues. When the length l_{queue} is larger than the limit L_{limit}, an agent should not add a new task to its local queue. The limit L_{limit} is set to the cluster value V_c for a cluster. In essence, this rule helps balance load within the cluster. In detail, a rule-message, which includes the limit L_{limit}, will be transferred to all subordinates within a cluster. When the current queue length is already greater than the limit L_{limit} , the probability of the action that locally execute tasks will be set to zero. However, the length l_{queue} of the local queue of an agent may be greater than the load limit so that the average load of the cluster could be higher than before.

C. Combining Action shaping and Reward Shaping

According to the above analysis, we can get suggestions, rules and shaping reward. For suggestions, we can change the equation (1) as follows:

$$deg(s,a) = \begin{cases} 0 & \text{if } |G(s,a)| = 0 \\ R_{suggestion} & \text{if } |G(s,a)| = 1 \\ & \text{and } < c, A, d > \in G(s,a) \end{cases} \quad (13)$$

Based on the equation (13), we can compute $f_s(deg(s,a))$. When the length l_{queue} is larger than the limit L_{limit}, an agent have a rule. So we can use the equation (6) to compute the $f_r(R(s,a))$. Therefore, we can do the combining work based on the equation (3) and the equation (7)

In our experiments, the adjustment parameter α in the equation (5) is 1 and the constant k in the equation (8) is 1000.

VI. Evaluation

We use DTAP [1] to evaluate our algorithms. The main goal of DTAP is to minimize the average time of service time(ATST). The service time means the interval time which is from the task's arriving to the end of the task's execution. The communication cost among agents is proportional to the distance between them, one time unit per distance unit.

A. Experimental Design

The experimental setup is almost based on the method in [13]. But we choose PGA-APP as the MARL algorithm [14]. PGA-APP is a gradient-based algorithm that arguments a basic gradient ascent algorithm with policy prediction. The characteristics of PGA-APP is that an agent adjusts its strategy in response to forecasted strategies of the other agents, instead of their current ones. The agent chooses an action only based on its reward.

The state of an agent is mapped from its average work load over a period of time $\tau(\tau = 500)$. There are three measurements:

- ATST(average total service time): The measurement is used to evaluate the overall system performance. Thus, it is the main measurement for evaluating MARL.
- AMSG(average number of message per task): The measurement indicates the overall communication overhead for finishing one task.
- TOC(time of convergence): The value of TOC is calculated based on the ratio of ATST's deviation to its means.

The two-dimension grid networks of agents are 27*27 grids for experiments. All agents have a same execution rate. The mean of task service time μ is 10. We tested three patterns of task arrival rates: boundary load, center load and corner load.

In each simulation run, ATST and AMSG are computed every 500 time units to measure the progress of system performance. The simulation ran over 10 times to get average values. We compared four structures: no supervision, action shaping, reward shaping and combined shaping that combines action shaping and reward shaping. For three structures with supervision, there are 81 clusters and each cluster has 3*3 agents.

B. Results

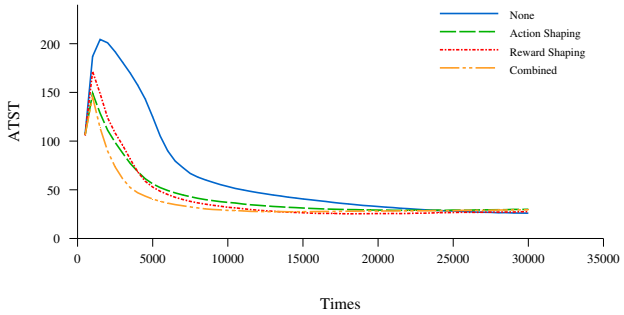

Fig. 7. ATST with boundary load for different structures

Fig. 7 shows the result of ATST for the pattern of boundary load. Fig. 8 shows the result of ATST for the pattern of center load. Fig. 9 shows the result of ATST for the pattern of corner load. All pattern structures produced similar results. As expected, reward shaping has a higher ATST than that of action shaping at the early stage of learning. This is because action shaping can immediately guide the agent to choose good

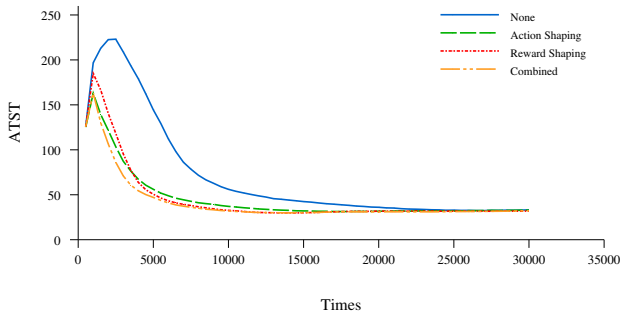

Fig. 8. ATST with center load for different structures

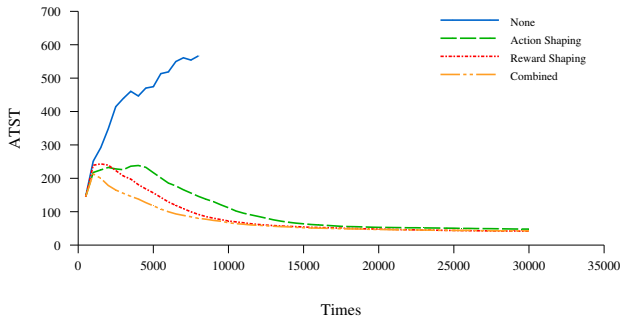

Fig. 9. ATST with corner load for different structures

actions and to avoid bad actions. But for reward shaping, although it can also affect the exploration of the agents to accelerate the convergence of learning process, the agents may still explore some bad states so that it needs more time compared to action shaping. Nevertheless, the ATST with reward shaping is still better than that with no supervision at the early stage because the supervisor can provide a partial global view. As time goes by, the algorithm with reward shaping converges more quickly than that with action shaping. The reason, as mentioned before, is the agent with reward shaping may avoid more bad states.

When we combine with reward shaping and action shaping, the algorithm shows significantly improved performance, which is because the combined algorithm improves the performance of reward shaping at the early stage.

TABLE I
PERFORMANCE OF DIFFERENT STRUCTURES WITH BOUNDARY LOAD

Supervision	ATST	ATST(20500)	AMSG	TOC
None	32.2±0.4	32.2±0.4	5.9±0.1	20500
Action Shaping	31.3±0.6	29.1±0.5	6.8±0.2	15000
Reward Shaping	28.4±0.3	25.6±0.7	6.5±0.2	12500
Combined	30.4±0.5	28.3±0.5	7.2±0.1	8500

Table I, Table II and Table III show the different measures, including ATST, AMSG and TOC, where the columns of ATST and AMSG are calculated at the time of convergence

TABLE II
PERFORMANCE OF DIFFERENT STRUCTURES WITH CENTER LOAD

Supervision	ATST	ATST(18500)	AMSG	TOC
None	37.5±2.0	37.5±2.0	9.0±0.1	18500
Action Shaping	35.6±0.7	31.5±0.5	9.6±0.2	11000
Reward Shaping	32.7±0.5	31.6±0.5	9.8±0.2	10000
Combined	33.3±0.9	31.0±0.8	9.5±0.2	9000

TABLE III
PERFORMANCE OF DIFFERENT STRUCTURES WITH CORNER LOAD

Supervision	ATST	ATST(16500)	AMSG	TOC
None	N/A	N/A	N/A	N/A
Action Shaping	59.1±5.1	59.1±5.1	14.2±0.7	16500
Reward Shaping	54.2±6.7	52.1±5.5	15.5±0.9	15000
Combined	53.1±7.8	50.2±5.6	13.7±1.4	14500

and ATST(*) is the performance at the latest time of convergence for all four cases. We can see that the algorithms with shaping technology also decreases system ATSTs while speeding up the convergence. This is because shaping technology can alter an agent's exploration. Moreover, in multi-agent systems, this altering will also change other agents's learning processes. Many multi-agent applications have multiple points of equilibrium. So these agents may explore more state-action space so that MARL will converge to a better equilibrium, especially when the shaping technology that agents take is based on supervisory information.

From these tables, we can observe that the algorithm with shaping technology do not produce heavy communication overhead. We can also see that the system without supervision almost always has lower AMSG than that of the others. This is because these systems with an organizational structure increase the communication overhead for sending reports, rules and suggestions.

In experiments, we actually found that the systems with reward shaping, including only reward shaping and combined shaping have higher AMSG than that of the systems with only action shaping at the middle stage. This is because the systems with reward shaping encourage their agents to do more exploration so that they have more messages to transfer. But at the later stage, the systems with reward shaping almost always have the same AMSG as that of the systems with only action shaping because the systems with reward shaping have reduced the exploration in the later stage.

VII. CONCLUSION

Acting shaping has been used for coordinating multi-agent reinforcement learning (MARL). In this paper, we presented a reward shaping method for coordinating MARL. Furthermore, we show action shaping and reward shaping are complementary and present a two-level new shaping approach that combine reward shaping and action shaping for coordinating MARL. To dynamically generate supervisory information for supporting reward and action shaping, our approach employs

an two-level organizational structure. The higher-level agents gather information from the lower-level agents and their neighboring supervisory agents and then dynamically generates supervisory information. This supervisory information is then integrated into the local learning processes of lower-level agents by using our new shaping approach so that their learning processes are coordinated. Experiments show that our two-level shaping approach effectively speeds up MARL and improves the learning quality.

VIII. Acknowledgment

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

References

[1] S. Abdallah and V. Lesser. Multiagent reinforcement learning and self-organization in a network of agents. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems* (AAMAS-07), pages 172–179, Honolulu, Hawaii, May 2007. (Best Paper Award nominee).

[2] J. Asmuth, M. L. Littman, and R. Zinkov. Potential-based shaping in model-based reinforcement learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pages 604–609, 2008.

[3] M. Babes, E. M. De Cote, and M. L. Littman. Social reward shaping in the prisoner's dilemma. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3*, pages 1389–1392. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[4] S. Devlin and D. Kudenko. Theoretical considerations of potential-based reward shaping for multi-agent systems. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 225–232. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[5] S. Devlin and D. Kudenko. Dynamic potential-based reward shaping. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 433–440. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[6] S. Devlin, D. Kudenko, and M. Grześ. An empirical study of potential-based reward shaping and advice in complex, multi-agent systems. *Advances in Complex Systems*, 14(02):251–278, 2011.

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

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

[9] A. D. Laud. *Theory and application of reward shaping in reinforcement learning*. PhD thesis, University of Illinois, 2004.

[10] B. Marthi. Automatic shaping and decomposition of reward functions. In *Proceedings of the 24th international conference on Machine learning*, pages 601–608. ACM, 2007.

[11] A. Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the 16th International Conference on Machine Learning,*, pages 278–287, 1999.

[12] J. Randlov and P. Alstrom. Learning to drive a bicycle using reinforcement learning and shaping. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 463–471, 1998.

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

[14] C. Zhang and V. Lesser. Multi-agent learning with policy prediction. In *Proceedings of the 24th National Conference on Artificial Intelligence (AAAI10)*, 2010.

[15] C. Zhang and V. Lesser. Coordinating multi-agent reinforcement learning with limited communication. In *AAMAS'13*, 2013.

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