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

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

Chongjie Zhang
School of Computer Science
University of Massachusetts
chongjie@cs.umass.edu

Victor Lesser
School of Computer Science
University of Massachusetts
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-level agent organization structures 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 the lower-level agents. The lower-level agents then uses this knowledge to coordinate their local learning processes with other agents. Experimental results show our approach effectively speeds up the convergence of multi-agent learning in large systems.**

***Keywords*-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 system performance. Multi-agent reinforcement learning (MARL) provides an attractive approach to this question. MARL allows agents to explore the environment through trial and error, to adapt their behaviors to the dynamics of the uncertain and evolving environment, and to gradually improve their performance through experiences.

One of key research challenges for MARL is to scale learning to large cooperative systems. Coordinating MARL [6, 7, 12, 15, 14] provides a promising direction to address this challenge. Using coordinated 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 [6, 7, 12, 15, 14]. 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 reward shaping can potentially address this issue in coordinating MARL. Reward shaping [1, 9, 10] 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. Moreover, reward shaping can expand this effect temporally and spatially. These will be explained in detail later. Unlike other work on multi-agent reward shaping [2, 5, 3, 4], 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.

However, coordinating MARL with reward shaping cannot generate a reasonable policy early in the learning process because it needs more time for exploration than that with action shaping. In this paper, we proposes a method which combines reward shaping and action shaping. Empirical results show that reward shaping and action shaping can be complementary to each other and combining them for coordinating MARL can further improve learning performance.

In this paper, we illustrate our approach using a coordinating MARL framework [12] (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 [12], MARL may converge slowly, converge to inferior equilibria, or even diverge in realistic settings. To address these issues, a supervision framework was proposed in [12]. 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.

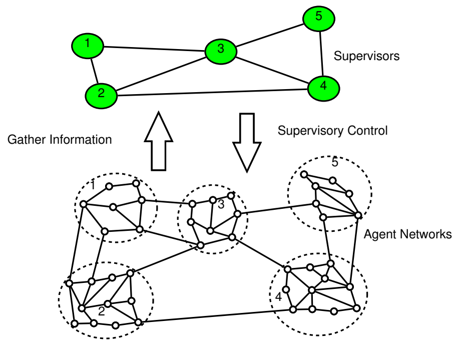

Figure 1. The two-level hierarchical learning structure

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.

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 *RL* 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:

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

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

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

where π^A is the adapted policy, π is the learning policy, $RL(s,a)$ 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 [2, 8, 10]. 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. We will then discuss how to periodically and dynamically compute appropriate shaping rewards.

MARL can be model-free, such as PGA-APP [13] that is built upon Q-learning. 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 [10, 11]. The one-step Q-learning with reward shaping is defined by [10]:

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

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:

$$F^t(s,a,s',a') = f_s(deg(s,a)) + f_r(RL(s,a)) \tag{3}$$

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

$$f_s(deg(s,a)) = \begin{cases} r(s,a) * \eta(s) \\ \qquad * \; deg(s,a) \text{ if } r(s,a) > 0 \\ -r(s,a) * \eta(s) \\ \qquad * \; deg(s,a) \text{ else if } r(s,a) \le 0 \end{cases} \tag{4}$$

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 the associated rule set $RL(s,a)$ is not empty, the shaping reward $r_{rule}(s,a)$ can be defined as:

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

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

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

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

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. [12] 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 encouraging 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. However, action shaping affect fewer states temporally and spatially than that of reward shaping.

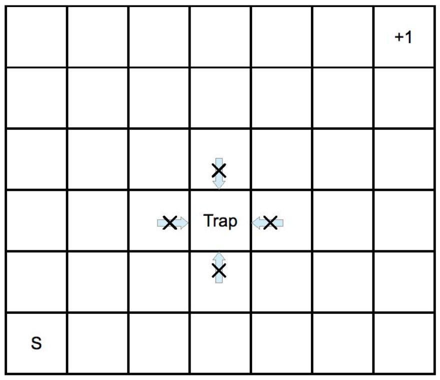

Figure 2. The grid world with action shaping

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 $Down$. Actions are stochastic motions. For example, if an agent takes 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

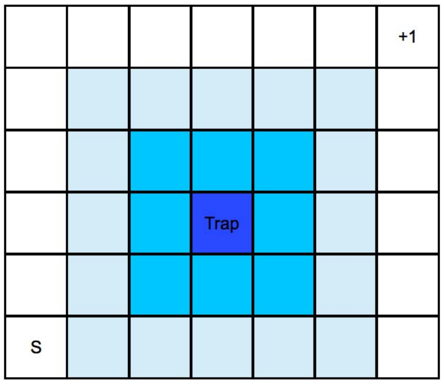

Figure 3. The grid world with reward shaping

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 does not directly affect the Q-values of the neighboring states, but 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 shown in Figure 3. This is because reward shaping makes the negative reward of the trap greater and thus the agent will less likely explore neighboring states of the trap.

The explanation above is from the spatial perspective. Suppose that agents are in a non-stationary environment where the trap could be moving. After the trap has moved, the effect of reward shaping will take more time to die out since it has already affected neighboring states, but, in contrast, the effect of action shaping will be immediately adjusted to the new state. Thus action shaping is more responsive but local in character, whereas reward shaping is less responsive but non-local in character, and thus they will bring different impacts from the temporal point. For example, in the distributed task allocation problem (DTAP) [12], action shaping will bring more benefits for adjusting the load balance in a cluster. Figure 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 action a_0, the agent's load increase. If an agent takes action a_1, the agent's load may be reduced because of completed tasks. Assume that, at an early stage of the learning, the agent has a rule on state s_2 for limiting its local queue length (i.e., preventing from taking action a_0 on state s_2). As a result,if we use reward shaping, the Q-value of action a_0 in state s_2 will be reduced. Due to the non-stationary environment, the agent now has an overwritten rule for limiting its local queue on state s_3 at the later stage. Then using reward shaping, the agent will still have the low Q-value of action a_0 in state s_2 for some time. However, in contract, if the agent only uses action shaping, then it will select action a_0 with high probability in state s_2 and visit state s_3 more quickly and frequently.

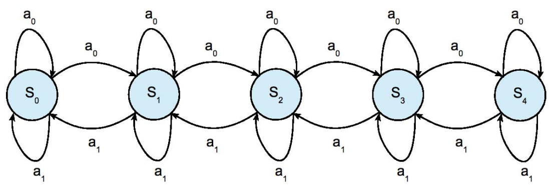

Figure 4. The state transition diagram of DTAP

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 strategy of MARL while reward shaping needs more time to improve Q-values. Therefore, combining action shaping and reward shaping can potentially be beneficial.

To combine these two shaping methods, the receptivity function $\eta(s)$ is used on both suggestions and rules of action shaping. 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:

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

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 coordination 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

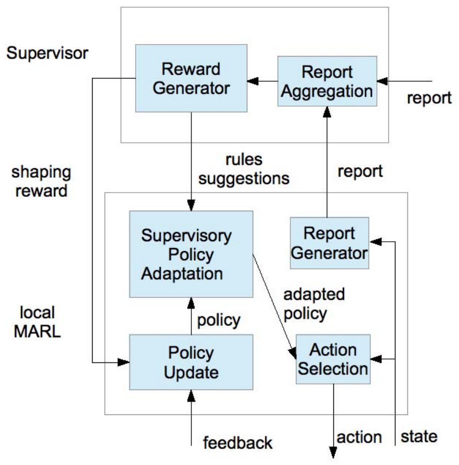

Figure 5. Supervised MARL

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 that arrive according to 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. Figure 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. Let $p_i(s)$ be the probability that agent i visits state s. We define $R_i(s) = \sum_{a\in A_i} \pi_i(s,a)r_i(s,a)$, where $\pi_i(s,a)$ 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. Then, V_c can be calculated by the equation (9).

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

B. Action Shaping and Reward Shaping

As mentioned before, the shaping reward for each agent can be calculated from its suggestion degree, which is from its supervisor. In DTAP, the suggestion degree is computed using the cluster values. 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 improve learning performance of the cluster with a lower cluster value without significantly affecting learning performance of the cluster with a higher cluster value. To achieve this goal, we need to compute the cluster-level suggestion degree $r_{suggestion}$ using d_v values, which can express the quantitative goodness of distributed reinforcement learning, and then to map such a non-local suggestion degree to local suggestion degrees.

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:

$$d_v = V_{c2} - V_{c1} \tag{10}$$

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

$$r_{suggestion} = \alpha(V_{c2} - V_{c1}) \tag{11}$$

where α is an adjustment coefficient.

For DTAP, cluster value V_c is approximately 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 approximated by the average queue length.

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 (10), 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 to forward tasks to cluster c_k according to the following suggestion degree:

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

A positive suggestion degree 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 suggestion degree is given by:

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

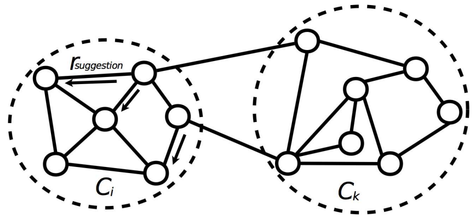

Figure 6. Propagating shaping reward

Using the cluster-level suggestion degree $r_{suggestion}$, we now consider generating the local suggestion degree for specific agents. Our method is to transfer the cluster-level suggestion degree to subordinates adjacent to cluster c_k, which is showed in Figure 6. The cluster-level suggestion degree will also be transferred to subordinates that are not on the boundary to other clusters, but with a discount factor based on their distance to the boundary. So shaping rewards will attenuate for agents further away from the boundary.

An agent may need to combine two or more suggestions on the same state-action pairs from its cluster manager based on its cluster being potentially connected to more than one cluster. Let $D_{suggestion}$ be the combined suggestion degree that combine two suggestions that an agent receives. Our combination strategy is showed as follows:

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

where r_1 and r_2 are two suggestion degrees that are received by the agent. This strategy can be generalized to combine more than two suggestion degrees. Once an agent gets their suggestion degrees, it computes its shaping rewards based on equation (4).

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

For DTAP, when an agent has too long a queue, it should forward any tasks that it received to other agents. We create a load limit rule to limit the local queue length. When the local queue 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.

When an agent's local queue length l_{queue} is larger than the limit L_{limit}, this load limit rule will be activated and the agent will then use equation (6) to compute shaping reward $f_r(RL(s, a))$ associated with this rule.

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 [12] to evaluate our approaches. The main goal of DTAP is to minimize the average time of service time(ATST) of all tasks received by the system. The service time of a task refers to the interval between its arrival time and the end time of its 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 the same as in [12], except that we choose PGA-APP [13] as the local learning algorithm. 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), which is used to evaluate the overall system performance. Thus, it is the main measurement for evaluating MARL.
- AMSG (average number of message per task), which indicates the overall communication overhead for finishing one task.
- TOC (time of convergence), which is used to evaluate the learning speed. 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. Note that after the systems reach the TOC point computed by our method, the learning performance may still continue improving but with a small rate.

The two-dimension grid networks of agents are 27*27 grids for experiments. All agents have the 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 integrates action shaping and reward shaping. For three structures with supervision, there are 81 clusters and each cluster has 3*3 agents.

B. Results

Figure 7, 8, and 9 show results of ATST under different task load patterns. All pattern structures produced similar

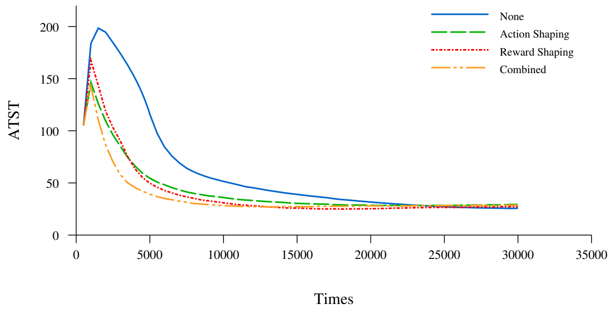

Figure 7. ATST with boundary load for different structures

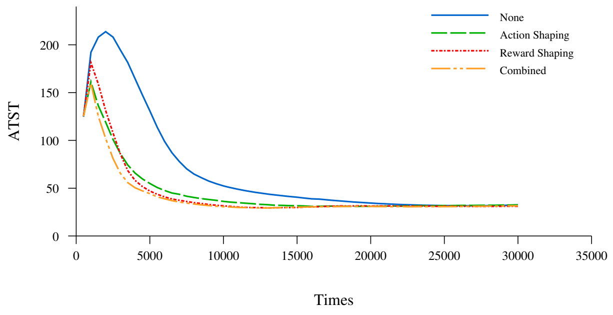

Figure 8. ATST with center load for different structures

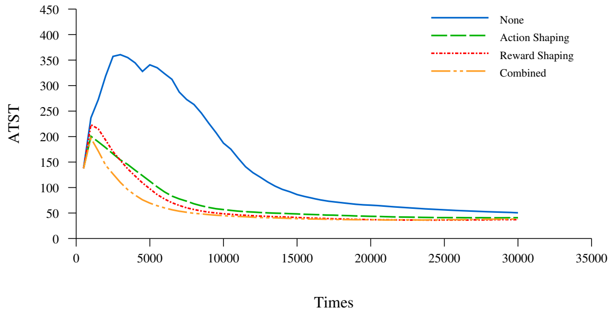

Figure 9. ATST with corner load for different structures

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 agents to choose good actions and to avoid bad actions, while, using reward shaping, agents still need to explore some bad states. Nevertheless, reward shaping outperform the case with no supervision at the early stage because shaping rewards implicitly provide a partial global view and coordinate agents' learning processes. As time goes by, learning with reward shaping converges more quickly than that with action shaping. The reason, as mentioned before, is that agents with reward shaping can avoid more bad states. Learning with combined reward shaping and action shaping shows further improved performance, which is because the combined method improves learning performance of reward shaping at the early stage.

Table I
PERFORMANCE OF DIFFERENT STRUCTURES WITH BOUNDARY LOAD

Supervision	ATST(12500)	ATST(8500)	AMSG	TOC
None	45.9±0.6	60.2±0.8	5.9±0.1	20500
Action Shaping	33.5±0.6	39.9±0.9	6.8±0.2	15000
Reward Shaping	28.4±0.3	35.0±0.3	6.5±0.2	12500
Combined	27.6±0.4	30.4±0.5	7.2±0.1	8500

Table II
PERFORMANCE OF DIFFERENT STRUCTURES WITH CENTER LOAD

Supervision	ATST(10000)	ATST(9000)	AMSG	TOC
None	56.1±5.5	62.8±7.5	9.0±0.1	18500
Action Shaping	37.1±1.0	39.3±1.0	9.6±0.2	11000
Reward Shaping	32.7±0.5	34.7±0.3	9.8±0.2	10000
Combined	32.0±0.9	33.3±0.9	9.5±0.2	9000

Table III
PERFORMANCE OF DIFFERENT STRUCTURES WITH CORNER LOAD

Supervision	ATST(15500)	ATST(12000)	AMSG	TOC
None	82.5±18.1	129.2±42.6	11.7±0.4	29000
Action Shaping	47.6±1.6	52.0±2.0	11.8±0.3	16000
Reward Shaping	40.8±1.7	44.7±1.8	12.5±0.4	15500
Combined	38.5±0.7	41.8±1.1	11.6±0.2	12000

Table I, Table II and Table III show different measures, including ATST, AMSG and TOC under different task load patterns. AMSG are calculated at the time of convergence and ATST(*) is computed at the time step as specified in the parentheses. As we mentioned, the system's ATST may still continue decreasing after reaching TOC computed by our method, but with a small rate. To better illustrate the results, we compare learning performance of all four cases at both time of convergence of reward shaping and combined shaping. We can see that learning with shaping technologies can decreases system ATSTs while speeding up the convergence. In addition, reward shaping performs better than action shaping and combined shaping outperforms reward shaping in terms of ATST and learning convergence. We can also observe that learning shaping technologies do not produce heavy communication overhead.

To further analyze results, we conducted two pairwise hypothesis tests for comparing the performance of combined shaping and reward shaping with different loads.

Hypothesis 1: learning with combined shaping converges faster than that with reward shaping. The p-value of this hypothesis test is 2.385E-040 for the boundary load, 3.595E-019 for the center load, and 1.722E-019 for the corner load, all of which are less than 0.05 and statistically confirm Hypothesis 1.

From our experimental results, we observe that combined shaping increases convergence speed by between 10% and 30%.

Hypothesis 2: learning with combined shaping produces a better (or lower) ATST than learning with reward shaping at the earliest convergence time among four cases, i.e., the TOC of combined shaping. The p-value of this hypothesis test is 5.508E-040 for the boundary load, 6.508E-022 for the center load, and 5.319E-011 for the corner load, all of which are less than 0.05 and statistically confirm Hypothesis 2. We observe that, at the earliest convergence, combined shaping improves overall performance in ATST between 4% and 13% over reward shaping.

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 new shaping approach that combines reward shaping and action shaping for coordinating MARL. To dynamically generate supervisory information for supporting reward and action shaping, our approach employs a 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 local learning processes of the lower-level agents by using our new shaping approaches so that their learning processes are coordinated. Experiments show that our two-level shaping approach effectively speeds up MARL and improves the learning quality, and our combined shaping method outperforms both action shaping and reward shaping when applied alone.

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 the views of the National Science Foundation.

References

[1] 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.

[2] 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, 2008.

[3] 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, 2011.

[4] 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.

[5] 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.

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

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

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

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

[10] 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.

[11] 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.

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

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

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

[15] 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.