The Power of Suggestion

Nicholas Zerbel
Oregon State University
Corvallis, Oregon, USA
zerbeln@oregonstate.edu

Kagan Tumer
Oregon State University
Corvallis, Oregon, USA
kagan.tumer@oregonstate.edu

ABSTRACT

Multiagent teams have been shown to be effective in many domains that require coordination among team members. However, finding valuable joint-actions becomes increasingly difficult in tightly-coupled domains where each agent's performance depends on the actions of many other agents. Reward shaping partially addresses this challenge by deriving more "tuned" rewards to provide agents with additional feedback, but this approach still relies on agents randomly discovering suitable joint-actions. In this work, we introduce Counterfactual Agent Suggestions (CAS) as a method for injecting knowledge into an agent's learning process within the confines of existing reward structures. We show that CAS enables agent teams to converge towards desired behaviors more reliably. We also show that improvement in team performance in the presence of suggestions extends to large teams and tightly-coupled domains.

KEYWORDS

Multi-Agent Learning; Evolutionary Algorithms; Co-Evolutionary Algorithms; Reward Structures for Learning

1 INTRODUCTION

Multiagent teams have been effectively applied to many domains requiring coordination among team members such as robot soccer [23, 34, 35], the manipulation of large objects (such as boxes) [10, 38], and joint exploration tasks [4, 15, 17, 30]. However, achieving coordinated behavior in tightly-coupled domains —defined as domains requiring agents to take similar actions simultaneously to achieve a task— is a significantly more challenging learning problem. For example, in a task involving moving a large table, which requires at least three agents to move, one agent may try to move the table before other agents have discovered this action. Lifting the table is the correct action to take in this scenario; however, the agent will receive no reward feedback due to missing inputs from partner agents. In such situations, reward shaping is often used to tune agent feedback signals to allow agents to gain additional information from the reward signal [21, 29]. The additional feedback provided by reward shaping can be beneficial; however, exploration during the learning process still relies on the random discovery of actions. This often means that agent teams may not converge to an ideal behavior since reward shaping cannot tune feedback signals for actions agents never discover.

Injecting domain knowledge into an agent's learning process offers a potential solution to this problem. Potential Based Reward Shaping (PBRS) is a good example of incorporating knowledge at the reward level as it is a principled and theoretically correct method of incorporating heuristic knowledge into an agent [7] which is similar to nonzero Q-table initialization [21]. Although PBRS provides guarantees about preserving the Nash equilibria of a system, knowledge injection via the potential function requires enough knowledge to estimate the value of states in the domain which is not always available for certain problems.

In this paper we introduce Counterfactual Agent Suggestions (CAS) as a way to inject knowledge into a learning multiagent system as a low-level suggestion. These suggestions are provided to agents periodically during the learning process by a supervisor agent that is external to the system. The supervisor can be thought of as a human user guiding each agent's learning process by providing suggestions at critical moments when those suggestions may influence an agent's policy. The supervisor does not know what each agent's policy should be; however, the supervisor does have a preference for certain joint-actions over others.

CAS works within the framework of existing reward structures that leverage counterfactuals to produce shaped rewards. For example, D_{++} [28] allows agents to use hypothetical (counterfactual) partners, which are copies of themselves, to derive "stepping stone" rewards to reinforce tightly-coupled joint-actions in the absence of the required number of partners. This setup creates a shaped reward where agents compare a hypothetical state where they were as capable as n other agents with the actual global state where they were not. In this work, we explore how a supervisor can provide suggestions using the counterfactuals already present in D_{++}. These suggestions encourage agents to take similar exploratory actions while learning, enabling agent teams to converge to a desired behavior more reliably without changing the reward function.

The primary contributions of this work are to:

(1) Provide a method for injecting knowledge into the multiagent learning process using a low-level suggestion,
(2) Devise strategies for encoding knowledge into suggestions that induce desired behaviors from a team of agents.

In this work, we test the performance of CAS in a problem where agents must observe an n-sided polygon to watch for a mobile Point of Interest (PoI). If one side of the polygon does not have an agent observing it, a target can hide on the unobserved side; therefore, an n-sided polygon requires n agents to guarantee full coverage. Using this tightly-coupled PoI observation problem, we show that, in the presence of suggestions, agent teams converge to desired behaviors and joint-actions more reliably than agent teams that do not receive suggestions even when they use the same objective function for reward feedback. We also show that team performance improves in the presence of suggestions, and that this improvement in performance extends to larger agent teams operating in tightly-coupled domains.

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 2020, Auckland, New Zealand

https://doi.org/doi

2 BACKGROUND

There have been many successful applications of multiagent learning in coordination problems including search and rescue [37], mine collection [11], and coverage problems [1, 14, 22]. To address the coordination challenge in tightly-coupled domains, many works utilize non-learning methods such as direct agent-to-agent communication [10] or auction based methods [16] to establish coordination. Although communication based approaches are generally effective at addressing agent coordination, communication is often computationally expensive to implement as agents must learn what to communicate, when to communicate, and how to make use of the information received. Additionally, communication can be difficult to maintain across an entire team of agents particularly in tasks involving exploration and surveillance in remote areas where communication is limited. To reduce the complexity of learning with communication, behavior based approaches aim to determine agents' fitness based on internal motivators that are related to the task at hand [25, 26]. In this work, we assume that there is no direct agent-to-agent transfer of data; however, there is an implied communication between the supervisor and agents when suggestions are provided. This works focuses on how to utilize the information contained in suggestions; therefore, we assume that communication between the supervisor and agents requires only limited resources.

2.1 Reward Shaping

In cooperative multiagent systems, the design of agent reward functions has a significant impact on agent-to-agent interactions and the performance of the overall system [12, 33]. The act of creating these reward functions is known as reward shaping: a process that aims to provide agents with additional feedback by tuning reward signals to improve the quality of an agent's policy or the convergence speed towards a solution [21, 29]. The process of reward shaping is often represented as:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R + F(s,s') + \gamma \max_{a'} Q(s',a') - Q(s,a)] \quad (1)$$

where F(s, s') is a general term representing a shaped reward which is based on state transitions.

In many multiagent reinforcement learning problems, using heuristic knowledge has been shown to be effective in improving solution quality and decreasing solution convergence time [3, 19, 20, 32]. However, the mechanisms used to inject this knowledge differ. In this work, we discuss one method, Potential Based Reward Shaping, due to its similarity to the subject of this work.

2.2 Potential Based Reward Shaping

In single agent reinforcement learning problems, Potential Based Reward Shaping (PBRS) is a principled method for incorporating heuristic knowledge into an agent's learning process [21]. PBRS works by assigning a potential to every state, s, according to a potential function, $\Phi(s)$. The shaped reward is then evaluated by computing the change in potential due to the transition from state s to state s'. This leads to a shaped reward defined by:

$$F(s,s') = \gamma\Phi(s') - \Phi(s) \quad (2)$$

where F(s, s') is the same shaped reward used in Equation 1.

In the scope of Q-learning, PBRS has been shown to be equivalent to initializing the Q-table to $\Phi(s)$ [39]. It has also been shown that receiving the additional potential-based reward does not alter the optimal policy of an agent; however, it does influence agent exploration which can lead to agents learning different policies [21]. For applications in multiagent systems, PBRS has similar guarantees to single agent learning, and it does not affect the Nash equilibria of a system [5]. By incorporating suitable heuristics, PBRS can increase the probability that a team's performance will converge to a higher global utility while decreasing the overall convergence time [5].

Although knowledge injection via the potential function can provide agents with hints as to which states are potentially more beneficial than others, the information contained in these potentials does not necessarily provide agents with knowledge relating to specific actions [39]. This type of knowledge injection also works best in domains where the state-space can be discretized in a manner conducive to reinforcement learning methods such as Q-learning. It also requires enough domain knowledge to estimate the potential of states to influence agent exploration in a meaningful way. In this work, we develop a method for knowledge injection which is usable with non-reinforcement learning methods and which does not require state value estimates.

2.3 D++ Rewards

One of the challenges associated with distributed learning in cooperative multiagent systems is that agents with no prior knowledge of a task must randomly search through the joint state-action space, find actions that accomplish the objective, and learn to coordinate with teammates before any reward feedback is received. In loosely-coupled systems, difference rewards [2, 36] (defined in Equation 3) assist agents by comparing the actual team performance, $G(z)$, with a counterfactual team performance, $G(z_{-i} \cup c_i)$, where agent i's policy is substituted with a different policy or action, c_i. Often times, a null action is used to create a counterfactual global state where agent i took no actions (or executed a policy which had no impact on the system). This comparison with a hypothetical global state provides the agent with individualized feedback on how its policy affected the team's performance.

$$D_i = G(z) - G(z_{-i} \cup c_i) \quad (3)$$

In tightly-coupled systems, where tasks require complimentary actions from many agents, difference rewards are less effective because the reward feedback still relies on agents establishing and maintaining coordination before feedback is generated. In tightly-coupled domains, reward feedback is sparse and creates a difficult learning challenge as it is unlikely that a group of agents will simultaneously execute the same correct action when relying on random exploration. To address this difficult learning challenge, D_{++} rewards [28] have been shown to provide agents with "stepping-stone" rewards which evaluate the effect of introducing multiple counterfactual partners into the system identical to the agent receiving the reward. These counterfactual partners are used to create the hypothetical world state where an agent, i, was as capable as n other agents. This hypothetical world state is compared with the actual world state as defined in Equation 4.

$$D^n_{++}(i) = \frac{G(z_{+(\cup_{j=1,\ldots,n})i}) - G(z)}{n} \tag{4}$$

In the above equation, $G(z_{+(\cup_{j=1,\ldots,n})i})$ represents the hypothetical global state where n counterfactual partners were added to the system, $G(z)$ represents the actual global state without counterfactual partners, and n is used to discount the stepping stone reward with respect to the number of added counterfactual partners.

Discounting Equation 4 with respect to n allows agents to differentiate between the case where an objective needs only one more agent to be completed and the case where many more agents are required. Equation 4 is also formulated such that evaluating the case where $n = -1$ will produce the same difference reward as defined in Equation 3. This formulation is important for tightly-coupled tasks as difference rewards do not provide any feedback until coordination is established; however, D_{++} fails to provide gradient information where a sufficient number of agents have coordinated [28]. For these reasons, it is ideal to utilize both reward signals during the learning process. Algorithm 1 further summarizes how D_{++} balances difference rewards with stepping stone rewards.

Algorithm 1: Standard D_{++} Algorithm

```
1:  Calculate $D^{-1}_{++}$ using Eqn. 4
2:  Calculate $D^{N_A-1}_{++}$ using Eqn. 4
3:  if $D^{N_A-1}_{++} \leq D^{-1}_{++}$ then
4:      return $D^{-1}_{++}$
5:  else
6:      $n = 0$
7:      while $n < N_A - 1$ do
8:          $n = n + 1$
9:          Generate $n$ counterfactual partners
10:         Calculate $D_{++}(n)$ using Eqn. 4
11:         if $D^n_{++} > D^{n-1}_{++}$ then
12:             Return $D^n_{++}$
13:         end if
14:     end while
15: end if
16: Return $D^{-1}_{++}$
```

2.4 Related Works

Perhaps the most similar works to this one are works using PBRS to inject domain knowledge in the form of the potential function, $\Phi(s)$. Although PBRS can provide agents with hints as to which states are potentially more beneficial than others, the information contained in these potentials does not provide agents with specific insights over actions. To extend PBRS to provide more specific knowledge to agents, Wiewiora et al. propose a method for providing generalized advice to reinforcement learning agents known as Potential-Based Advice [39]. This advice function changes the shaping function to include an additional parameter relating to the policy an agent is currently evaluating. Essentially, this changes $F(s, s')$ to $F(s, a, s', a')$. Wiewiora et al. demonstrate two applications of this concept as Look-Ahead Advice and Look-Back Advice. Both potential-based advice and PBRS require enough domain knowledge to assign potential value estimates to states within a system. Potential based approaches are also designed to work within the context of Q-learning or other reinforcement learning methods. In this work, we define a method for injecting knowledge using CAS which does not require state value estimates and that is usable in non-reinforcement learning based approaches.

Similar works have also shown that the idea of counterfactuals used in difference rewards and D_{++} can be modified in a variety of ways to provide more useful reward feedback for agents. Devlin et al. demonstrated that PBRS and difference rewards can be combined to use counterfactuals as potential (CaP) [7]. CaP rewards agents for high-performing global evaluations while simultaneously encouraging agents to move towards states where other agents in the system are performing well. In this application $\Phi(s)$ depends on the state of all the agents in the system; therefore CaP can assign a different potential for the same local state which makes CaP an instance of dynamic PBRS [6]. Although this application does not explicitly deal with injecting knowledge into the multiagent learning process, it does show that counterfactuals can be augmented to provide additional insights during the learning process.

In another related work, Dixit et al. show that the counterfactual partners used in D_{++} can be modified to provide agents with additional insights in tightly-coupled, heterogeneous multiagent systems [8]. In standard D_{++}, agents infer counterfactual partners which are identical to themselves. This is a suitable implementation for homogeneous multiagent systems but not heterogeneous systems where agents may have different capabilities. Dixit et al. show that, if agents know the models for different partner types, agents can learn which partners to select to receive more insightful stepping stone rewards in these systems [8]. In this work, we use the counterfactuals present in D_{++} to inject knowledge into an agent's learning process by encoding that knowledge as a partner. The partners suggested by the supervisor do not need to represent a type of agent currently available in the system.

3 COUNTERFACTUAL AGENT SUGGESTIONS

In this work, we introduce Counterfactual Agent Suggestions (CAS) as a method for injecting knowledge into the multiagent learning process. These suggestions are delivered to agents periodically by another agent, referred to as the supervisor, which is external to the system. Conceptually, the supervisor can be thought of as a human user guiding a team of learning agents by providing them with suggestions at critical moments (defined as the discovery a possible joint-action) where those suggestions may influence an agent's policy. The supervisor does not know what an individual agent's policy should be, and the supervisor does not have access to an agent's internal state. The supervisor also does not need to know the joint-state of the system to make suggestions. However, the supervisor does have a preference over which joint-actions an agent should learn and creates suggestions designed to impart this knowledge. This enables agent policies to converge towards specific joint-actions or behaviors without comparing an agent's policy with a target policy distinguishing supervisors from critics in actor-critic methods [18].

CAS is designed to inject knowledge into an agent's learning process by using counterfactuals such as those already present in D_{++} (described in Section 2.3). In D_{++}, the counterfactual partners used by agents are generated by introducing multiple identical agents into the system to generate a hypothetical joint-state. Instead of relying on these agent-inferred partners, the supervisor suggests counterfactual partners to agents periodically throughout the learning process. These suggestions are delivered in Step 9 of Algorithm 1 which is the step where agents would typically generate partners for themselves. Although suggestions influence the rewards received by an agent, suggesting partners in D_{++} does not alter the calculation of the reward as defined by Equation 4.

In this work, we define two partner archetypes that the supervisor can use to encode knowledge within D_{++}: helpful partners and null partners. Helpful partners are counterfactual partners which are capable of satisfying the coupling requirements of a preferred joint-action once it is discovered by an agent. These partners can be used to provide agents with a positive stepping stone reward which will encourage agents to select that joint-action in the future. Null partners are built upon the concept of null actions which are often used in difference rewards to compute the hypothetical global state where an agent took no actions. Conceptually, a null partner exists within the system but does nothing to contribute towards the team's objectives or to detract from the team's overall performance. By suggesting null partners to an agent, the agent receives a stepping stone reward of zero.

To illustrate the application of these counterfactual suggestions, we use the n-sided polygon coverage problem discussed in Section 1 and illustrated in Figure 1. In the left panel, an agent has found a hexagonal PoI to observe; however, it is missing the five additional partners required to fully cover this structure. In the right panel, an agent has found a triangular PoI, but it is only missing two additional partners. Relying on standard D_{++}, the agent in the left panel would discover a positive stepping stone reward after inferring five counterfactual partners, and the agent in the right panel would discover a positive stepping stone reward after inferring two additional partners. It is much less likely that five additional agents will learn to cover the hexagonal structure compared to the triangular structure which only needs two additional agents. In this case, the supervisor can suggest null partners to the agent in the left panel to inject the knowledge that this action should be ignored. The agent on the right receives partners that satisfy the joint-action providing the agent with the knowledge that this action is desirable.

3.1 Cooperative CoEvolutionary Algorithms

To train agent control policies, encoded as neural networks (NNs) in this work, we use a standard Cooperative CoEvolutionary Algorithm (CCEA) [27]. CCEAs are an extension of Evolutionary Algorithms which have been shown to perform well in cooperative multiagent domains [9, 24]. The standard CCEA algorithm is described in Algorithm 2. The CCEA starts by creating a population of k neural networks for the total number of agents, N_A, operating in the system (one population for each agent). Within each generation, the CCEA generates k successor neural networks by mutating the weights contained within each existing neural network, bringing the total population of neural networks to a size of $2k$. Then, from each population, NNs are selected without replacement and placed on a team, T_i. Each team is simulated within the domain, and fitness values are assigned to each member of team T_i using a fitness function, $F(z)$. In this work, the fitness function, $F(z)$, is D_{++} described in Equation 4. Finally, k NNs are selected from the population of $2k$ individuals using ϵ-greedy selection to proceed to the next generation.

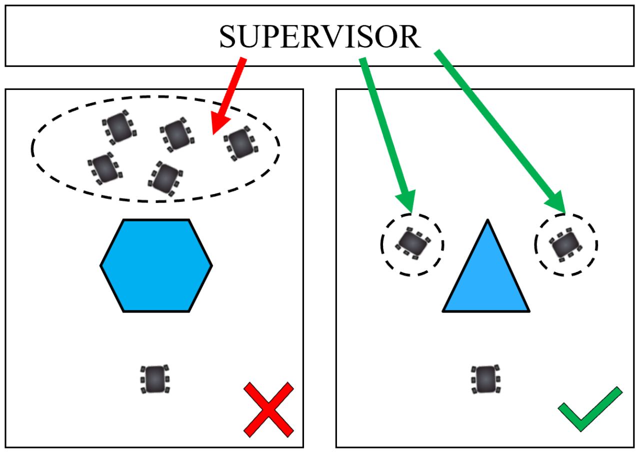


Figure (1) A supervisor provides counterfactual suggestions (contained within dashed circles) to agents in an n-sided polygon coverage problem. On the left, an agent discovers a hexagonal PoI but requires five additional partners for full coverage. On the right, an agent discovers a triangular PoI but only requires two more partners. It is far less likely that five more partners will learn to fully cover the hexagonal PoI, so the supervisor suggests null partners to discourage the agent from taking this action. Similarly, the supervisor suggests partners which satisfy the requirements for the triangular PoI to encourage learning this action.

Algorithm 2: Standard CCEA

```
1:  Initialize $N_A$ populations of $k$ neural network weights
2:  for $N_{Generations}$ do
3:      for Each population do
4:          Create $k$ successor NN weights
5:          Mutate successors
6:      end for
7:      for $i = 1- > 2k$ do
8:          Select NN weights from each population without
            replacement
9:          Add agents to team $T_i$
10:         Simulate $T_i$ in domain
11:         Evaluate fitness of each agent in $T_i$ using $F(z)$
12:     end for
13:     for Each population do
14:         Select $k$ solutions using $\epsilon$-greedy selection
15:     end for
16: end for
```

4 PROBLEM FORMULATION

In this work, we explore multiagent coordination in a monitoring task similar to the continuous rover problem where a set of homogeneous rovers are tasked with observing PoI located in a two-dimensional plane [2]. Rovers have no prior knowledge of the number of PoI in the region, the locations of the PoI, or the utility associated with observing a given PoI. To make this problem tightly-coupled, we modify the problem so that agents must observe an n-sided polygon to watch for a mobile PoI. A target can hide along an unobserved side; therefore, an n-sided polygon requires n agents to guarantee full coverage. The utility function for this environment is given by Equation 5.

$$G(z) = \sum_i \frac{(\prod_j^n N_{(i,j)})V_i}{\frac{1}{n}\sum_j^n \delta_{(i,j)}} \tag{5}$$

In Eqn. 5, z refers to the joint state-action of the rover team, V_i represents the value associated with PoI i, and n refers to the coupling requirement associated with PoI i. Term $N_{(i,j)}$ is a binary term which is 1 if rover j is within observational range of PoI i, and it is 0 otherwise. Term $\delta_{(i,j)}$ is the linear distance between PoI i and rover j. This makes the denominator of the term the averaged distance between PoI i and the n closest observing rovers.

Each rover is equipped with two sensors which provide it with a state input. One sensor is used to detect PoI and the other sensor is used to detect other rovers. Each sensor's inputs are discretized into four quadrants, Q, with respect to the body and heading of the rover. This sensor configuration produces eight state variables (four for each sensor type). The state variable representing rover detections is defined below as:

$$S_{ROV_{(j,Q)}} = \sum_{j' \in Q} \frac{1}{\delta_{(j,j')}} \tag{6}$$

where j represents the rover, Q represents the quadrant, and $\delta_{(j,j')}$ represents the linear distance between rover j and rover j' located within quadrant Q. The state variable representing PoI detections is defined as:

$$S_{PoI_{(j,Q)}} = \sum_{i \in Q} \frac{V_i}{\delta_{(i,j)}} \tag{7}$$

where V_i represents the value of PoI i, and $\delta_{(i,j)}$ represents the linear distance between PoI i in quadrant Q and rover j.

Each rover on the team is represented as a neural network whose inputs are the state variables described by equations 6 and 7. Neural networks are often referred to as universal approximators [13], and they are selected as the control model for rovers due to their ability to model continuous state-action control policies with only a rough estimate of the current state [2, 31].

5 EXPERIMENTAL SETUP

To investigate how CAS influences an agent's learning process, several experiments are presented comparing rover team performance in the presence of suggestions to rover team performance without suggestions. At each timestep in a simulation, each rover executes an action based on a policy encoded within a neural network. The outputs from this neural network are control signals that determines each rover's movement in the x and y directions. Each neural network is a single hidden layer, feed-forward network with 8 input nodes, 9 hidden nodes, and 2 output nodes. The network weights are trained using the CCEA defined by Algorithm 2. For each agent, there is a population of 40 networks that have weights initialized using a normal distribution, $\mathcal{N}(0, 1)$.

In the first experiment, three rovers must explore a region (size $30x30$) with two PoI. Each PoI has a coupling requirement of three, and each PoI is located on opposite sides of the region. The rovers only have enough time to explore one PoI. PoI 1 has a value of 10 and is located on the left side of the region, and PoI 2, located on the right, has value of 4. With the coupling requirement matching the number of rovers in the system, each rover must choose the same joint-action to satisfy the objective. Using this world setup, we run tests using two different suggestions. In one test, the supervisor encourages rovers to explore PoI 1 by suggesting helpful partners when a rover discovers PoI 1 and by providing null partners when a rover discovers PoI 2. A similar setup is used to encourage rovers to explore PoI 2 instead of PoI 1. In the second experiment, we use the same PoI setup as experiment 1; however, there are now six rovers and the coupling requirement of each PoI is now six. The same two suggestions are tested in this experiment.

In the third experiment, we investigate how rover teams perform in the presence of suggestions in a more complex system. In this system, there are six rovers, five PoI, and a coupling requirement of six. Three of the PoI are lower in value (less than or equal to 5), while the remaining PoI have values greater than 5. The region containing the rovers and the PoI is of size $40x40$. Tight coordination between all six rovers must be maintained for PoI to be observed in this world; however, the number of joint-actions to choose from is much greater making coordination more difficult to establish.

For each experiment, each PoI has an observability radius of 3, and the data presented is collected over 30 statistical runs. Error in figures is reported as the standard error of the mean. In each experiment, it is assumed that the world is fully observable to a rover's sensors. With full observability, each rover can detect the other rovers on the map, and the rovers can detect the presence of a PoI. However, the rovers cannot observe the PoI to collect rewards until they are within its observability radius.

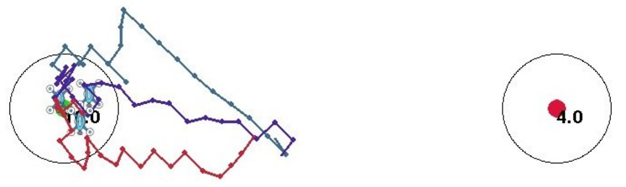
6 RESULTS

In the following experiments, we compare the performance of rover teams that learned without CAS, and those that learned with CAS. Each experiment compares how two different suggestion types from the supervisor influence the behaviors of the team.

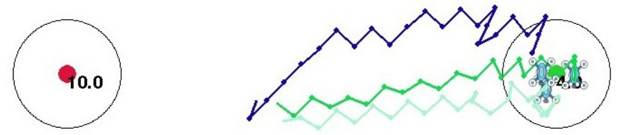
6.1 Two PoI, Coupling of 3

In experiment 1, three rovers must choose to either explore PoI 1 (worth 10.0) located on the left side of the region, or PoI 2 (worth 4.0) located on the right side of the region. Although there are only three rovers, each rover must learn the same joint-action to satisfy the coupling requirement of three for a complete PoI observation.

To encourage exploration of PoI 1, supervisors suggested rover partners capable of satisfying the PoI's coupling requirements once a rover's policy led them to PoI 1, and the supervisor suggested



(a) S1



(b) S2

Figure (2) **Examples of rover team behaviors induced by supervisor suggestions for a 2 PoI system with a coupling requirement of 3. In (a) suggested partners encouraged rovers to explore PoI 1 (denoted as S1), and in (b) suggested partners encouraged rovers to explore PoI 2 (denoted as S2). These behaviors are induced by using different suggestions, and the reward functions used in these tests are the same.**

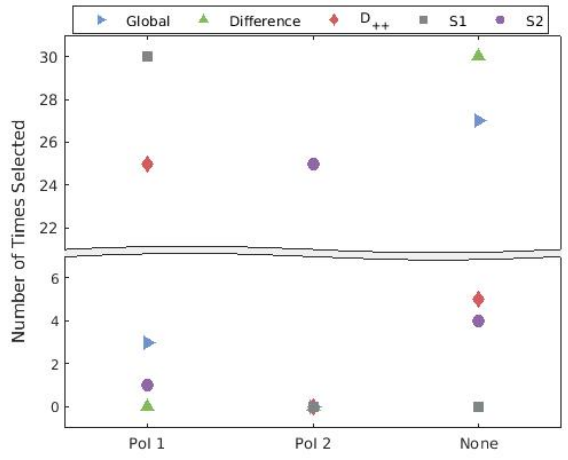


Figure (3) **This figure shows the number of times rover teams converged to PoI 1, PoI 2, or failed to learn a coordinated behavior by learning with suggestions and by learning without suggestions. Each PoI had a coupling requirement of 3 and the rover team size was 3. The convergence towards each behavior is counted with respect to the number of statistical trials (30).**

null partners to rovers exploring PoI 2. In the second test, the supervisor encourages rovers to explore PoI 2 by providing similar suggestions; rovers received helpful partners when their policies led them to PoI 2 and null partners if their policy led them to PoI 1. Suggestions encouraging the exploration of PoI 1 are denoted by S1, and suggestions encouraging the exploration of PoI 2 are denoted as S2. Examples of rover teams that learned the desired behaviors using S1 and S2 are illustrated in Figure 2.

Using CAS, the results of this experiment show that rover team behaviors can be modified even though the reward function remains unchanged. In fact, in the presence of suggestions, rover teams reliably learn behaviors that converge towards the desired joint-action (PoI 1 or PoI 2). Figure 3 illustrates the consistency of these behaviors by comparing the number of times teams learned a desired behavior with CAS across 30 statistical runs with the behaviors of rovers learning without CAS. In this figure we see that rovers learning with the global reward or difference rewards fail to establish consistent coordination even within this relatively simple system. The rover teams learning without CAS performed well; however, the rover teams either failed to learn proper coordination, or their behaviors always converged towards PoI 1. From a reward function optimization standpoint, this behavior is optimal; however, Figure 3 also shows that, with suggestions, the optimal behavior is learned more consistently using S1. In fact, using S1 allows the rover team to learn the desired behavior 100% of the time in this experiment. Figure 3 also shows that rover teams learning with S2 learn policies that select PoI 2 83% of the time. Although this behavior is sub-optimal, this behavior can be induced simply by providing a certain type of suggestion even though the reward function itself is identical to the one used by the rover teams learning with D_{++} and S1.

The results of this experiment also demonstrate that rover team performance also improves when the team learns with suggestions provided by the supervisor. In Figure 4, we see that the average team performance converges to the optimal value of 10.0 when the rover team learns using S1. Although D_{++} without CAS is capable of learning this behavior, the rovers converge more readily towards PoI 1 using S1 creating a more reliable outcome. Figure 4 also shows that the team score also converges towards the value of PoI 2 (which is 4.0) when rover teams learn using S2. Note that, although this behavior produces a sub-optimal outcome compared to D_{++} and S1, this behavior is being targeted through S2 and is achieved solely based on the suggestion as the reward function used to evaluate stepping stone rewards is the same in D_{++}, S1, and S2.

Although the setup of this experiment is relatively simple, the influence of suggestions on rover team behavior is clearly illustrated. Implementing a supervisor agent which provides agents with counterfactual agent suggestions enables agent policies to converge towards a desired outcome more reliably. This behavior modification occurs without any modification to the reward function, and it enables rover teams to coordinate more consistently, particularly in settings where every team member is required to complete an objective.

6.2 Two PoI, Coupling of Six

In this experiment, the number of rovers in the world is increased to six, and the coupling requirement of each PoI is also increased to six. Although the PoI configuration is unchanged, this setup provides a more difficult learning challenge as tight coordination must be established and maintained between all six rovers. Once again, we investigate the performance of the rover teams learning with two different suggestions provided by the supervisor. Suggestions

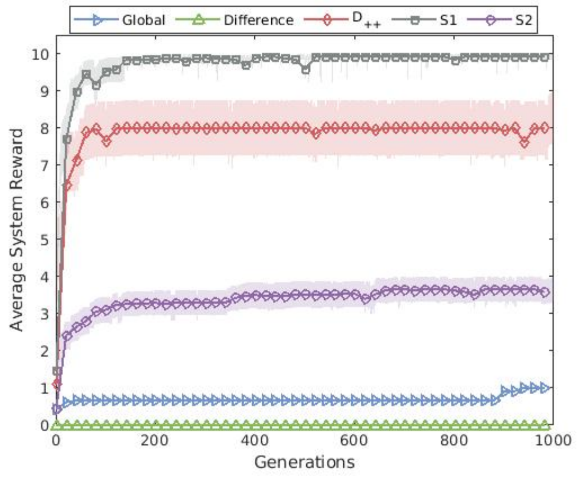


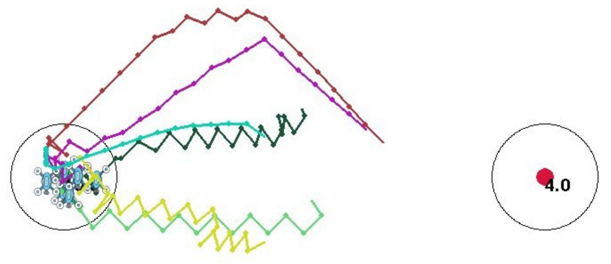
Figure (4) Rover team performance for a 2 PoI system with a coupling requirement of 3. Rover teams provided with suggestions encouraging the exploration of PoI 1 (denoted S1) learn this behavior more reliably than rover teams learning with D_{++}. Although sub-optimal, suggestions can be used to induce a behavior which encourages rovers to explore PoI 2 (denoted S2) even though the same reward function is used.

encouraging the exploration of PoI 1 or PoI 2 are denoted as S1 and S2, respectively. The performance of rover teams learning with CAS is compared to those learning without CAS.

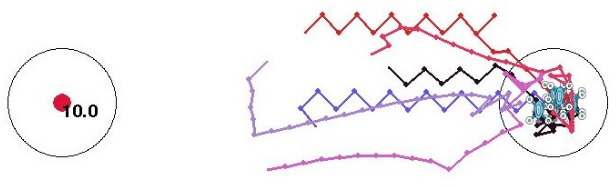
With the increased learning complexity associated with the higher agent-to-agent coupling requirement, rover teams have more difficulty establishing the tight coordination needed in this system. However, rover teams learning with suggestions manage to overcome some of this difficulty, enabling the team to learn coordinated behaviors more consistently. Figure 5 illustrates the rover teams which learned the desired behavior of observing PoI 1 or PoI 2 using S1 or S2, respectively.

Figure 6 further details the difficulty associated with learning a tightly-coupled task in this system. Rover teams learning with the global reward or difference rewards fail to establish coordination 100% of the time which is consistent with similar results presented in [28]. Although the rover team using D_{++} does learn to coordinate some of the time, the performance is extremely unreliable across all statistical runs. With six rovers and two PoI choices, the team ends up being divided among the two PoI most of the time. The rover teams learning with S1 learn the same optimal behavior as the team learning with D_{++}; however, the team converges towards this behavior more consistently leading to a more reliable outcome. In fact, the improved coordination in the presence of suggestions allows rover teams learning with S2 to achieve an average team score which is slightly better than the D_{++} without CAS even though though the team behavior is converging towards PoI 2 (worth 4.0). This result is illustrated in Figure 7.

Although there are only two options to choose from, establishing coordination between six agents is difficult when the performance of any one agent depends on the performance of five others. This experiment shows that, in the presence of suggestions, agent teams



(a) S1



(b) S2

Figure (5) Examples of rover team behaviors induced by supervisor suggestions for a 2 PoI system with a coupling requirement of 6. In (a) the supervisor suggested partners to rovers encouraging them to explore PoI 1 (denoted as S1). In (b) the supervisor suggested partners to rovers encouraging them to explore PoI 2 (denoted as S2). These behaviors are induced by using different suggestions, and the reward functions used in these tests are the same.

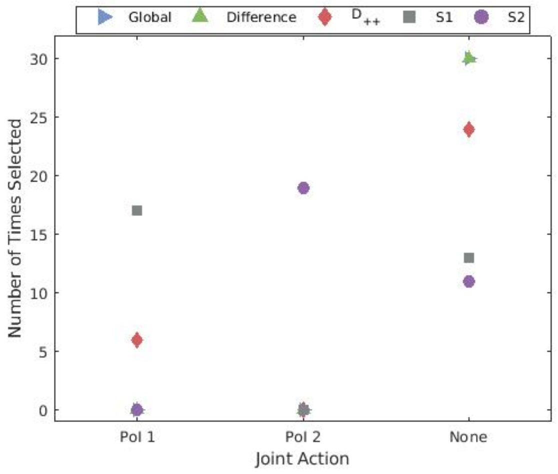


Figure (6) This figure shows the number of times rover teams learned to explore PoI 1, PoI 2, or failed to learn a coordinated behavior. Each PoI had a coupling requirement of 6 and the rover team size was 6. The convergence towards each behavior is counted with respect to the number of statistical trials (30).

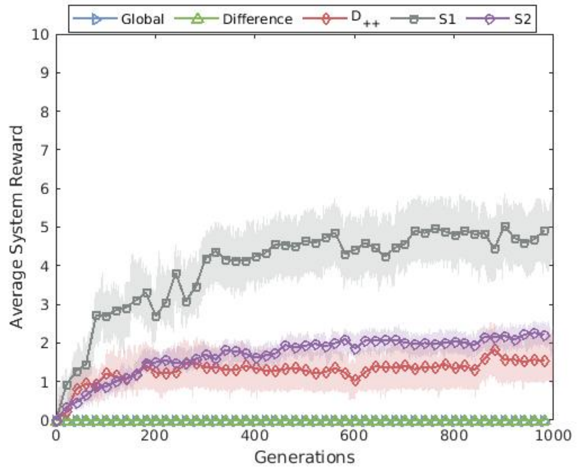


Figure (7) Rover team performance for a 2 PoI system with a coupling requirement of 6. Rovers receiving suggestions to explore PoI 1 (denoted S1) outperform teams learning with D_{++}. Although sub-optimal, rover teams provided with suggestions encouraging the exploration of PoI 2 (denoted S2) also reliably learn this behavior and achieved a better performance than the rover teams using D_{++} resulting from more consistent coordination.

are able to establish more consistent coordinated behavior even in this more difficult learning environment. To illustrate this effect further, the next experiment shows the performance of a six rover team learning in a larger world where there are five different joint-actions to choose from.

6.3 Five PoI, Coupling of Six

In the third experiment, we explore the performance of a six rover team in a world with five PoI scattered across a region that is of size $40x40$. The coupling requirement for each PoI is six; therefore, rover teams will need to establish strict coordination between all six individuals to successfully observe a PoI. As was the case in the previous experiments, two suggestion types are tested using the supervisor. Using suggestion type 1 (denoted S1) supervisors suggested helpful partners when a rover's policy led them to observe a PoI with a value greater than 5, and the supervisor suggested a null partner when a rover's policy led them to observe a PoI with a value less than or equal to 5. Similarly, using suggestion type 2 (denoted S2), supervisors suggested helpful partners when a rover's policy led them to explore a PoI with value less than or equal to 5 and null partners for PoI with values greater than 5.

Due to the greater number of joint-actions to choose from in this system, rover teams using D_{++} without CAS fail to establish effective coordination throughout the learning process. However, rover teams that receive suggestions manage to overcome some of the difficulty associated with this system. Figure 8 shows that using either S1 or S2, rovers are able to learn policies which allow them to establish coordinated behavior among all six individuals.

The performance of rover teams S1 and S2 are extremely similar even though these suggestions encourage rovers to explore PoI of different values. The similarity in performance is a symptom of

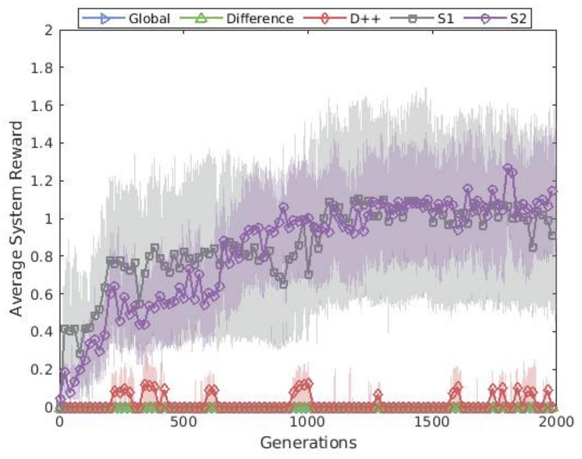


Figure (8) Rover team performance for a 5 PoI system with a coupling requirement of 6. Rover teams either received suggestions encouraging the exploration of PoI with values greater than 5 (denoted S1), or they received suggestions encouraging the exploration of PoI with values less than or equal to 5 (denoted S2). Agents using Global and Difference rewards did not learn resulting in an average reward of zero.

this learning environment where coordination is difficult to establish and maintain among six independent rovers. However, these results show that suggestions inject enough knowledge into each rover's learning process to establish some coordinated behavior whereas rovers relying on D_{++} alone fail to maintain coordination throughout the learning process.

7 CONCLUSIONS

In this work, we introduce a method for injecting knowledge into the multiagent learning process by using Counterfactual Agent Suggestions (CAS) that are delivered to agents using a supervisor agent. We showed that the supervisor can encode knowledge in the form of a counterfactual partner and then deliver those suggestions using existing reward structures such as D_{++}. Furthermore, we showed that CAS can be used to induce completely different behaviors in agents without modifying an agent's reward function.

This work illustrated the effectiveness of counterfactuals as a mechanism for providing periodic suggestions to agents throughout the learning process without explicitly modifying the agent's reward function or providing extensive examples. In our current formulation, these counterfactuals are partner agents. In the future, we will: (i) explore more fine-tuned counterfactuals that are combinations of counterfactual partners and counterfactual actions; ii) investigate the use of suggestions and agent-generated counterfactuals to induce more complex team behaviors that can not be achieved in the absence of suggestions.

8 ACKNOWLEDGEMENTS

This work was partially supported by the National Science Foundation under grant No. IIS-1815886 and by Scientific Systems Company Inc. under contract No. SC-1656-02. The authors thank Stephane Airiau for his thoughtful feedback on this paper.

REFERENCES

[1] Adrian Agogino, Chris HolmesParker, and Kagan Tumer. 2012. Evolving large scale UAV communication system. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*. ACM, Philadelphia, PA, USA, 1023–1030. https://doi.org/10.1145/2330163.2330306

[2] Adrian Agogino and Kagan Tumer. 2004. Efficient evaluation functions for multi-rover systems. In *Genetic and evolutionary computation conference*, Vol. 3102. Springer, Seattle, WA, USA, 1–11.

[3] Monica Babes, Enrique Munoz De Cote, and Michael L. Littman. 2008. Social reward shaping in the Prisoner's dilemma. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*, Vol. 3. International Foundation for Autonomous Agents and Multiagent Systems, Estoril, Portugal, 1389–1392.

[4] Mitchell Colby, Logan Yliniemi, and Kagan Tumer. 2016. Autonomous multiagent space exploration with high-level human feedback. *Journal of Aerospace Information Systems* 13, 8 (2016), 301–315. https://doi.org/10.2514/1.I010379

[5] Sam Devlin and Daniel Kudenko. 2011. Theoretical considerations of potential-based reward shaping for multi-agent systems. In *The 10th International Conference on Autonomous Agents and Multiagent Systems*, Vol. 1. International Foundation for Autonomous Agents and Multiagent Systems, Taipei, Taiwan, 225–232.

[6] Sam Devlin and Daniel Kudenko. 2012. Dynamic potential-based reward shaping. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, Vol. 1. International Foundation for Autonomous Agents and Multiagent Systems, Valencia, Spain, 433–440.

[7] Sam Devlin, Logan Yliniemi, Daniel Kudenko, and Kagan Tumer. 2014. Potential-based difference rewards for multiagent reinforcement learning. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multiagent Systems*, Vol. 1. ACM, Paris, France, 165–172.

[8] Gaurav Dixit, Nicholas Zerbel, and Kagan Tumer. 2019. Dirichlet-Multinomial Counterfactual Rewards for Heterogeneous Multiagent Systems. In *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE, New Brunswick, NJ, USA, 209–215.

[9] Sevan G. Ficici, Ofer Melnik, and Jordan B. Pollack. 2005. A game-theoretic and dynamical-systems analysis of selection methods in coevolution. *IEEE Transactions on Evolutionary Computation* 9, 6 (2005), 580–602. https://doi.org/10.1109/TEVC.2005.856203

[10] Brian P Gerkey and Maja J Mataric. 2002. Pusher-watcher: An approach to fault-tolerant tightly-coupled robot coordination. In *Proceedings 2002 IEEE International Conference on Robotics and Automation*. IEEE, Washington, DC, USA, 464–469.

[11] Dani Goldberg and Maja J. Matarić. 2003. Maximizing Reward in a Non-Stationary Mobile Robot Environment. *Autonomous Agents and Multi-Agent Systems* 6, 3 (2003), 287–316. https://doi.org/10.1023/A:1022935725296

[12] Pieter Jan't Hoen, Karl Tuyls, Liviu Panait, Sean Luke, and Johannes A La Poutre. 2005. An overview of cooperative and competitive multiagent learning. In *Proceedings of the First international conference on Learning and Adaption in Multi-Agent Systems*. Springer-Verlag, Utrecht, The Netherlands, 1–46.

[13] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 1989. Presentation on Multilayer Feedforward Networks are Universal Approximators. (1989), 359–366 pages. https://www.sciencedirect.com/science/article/pii/0893608089900208

[14] Andrew Howard, Maja J. Matarić, and Gaurav S. Sukhatme. 2002. An incremental self-deployment algorithm for mobile sensor networks. *Autonomous Robots* 13, 2 (2002), 113–126. https://doi.org/10.1023/A:1019625207705

[15] Athanasios Ch Kapoutsis, Savvas A. Chatzichristofis, Lefteris doitsidis, João Borges de Sousa, Jose Pinto, Jose Braga, and Elias B. Kosmatopoulos. 2016. Real-time adaptive multi-robot exploration with application to underwater map construction. *Autonomous Robots* 40, 6 (2016), 987–1015. https://doi.org/10.1007/s10514-015-9510-8

[16] Sven Koenig, Pinar Keskinocak, and Craig Tovey. 2010. Progress on agent coordination with cooperative auctions. In *Twenty-fourth aaai conference on artificial intelligence*, Vol. 3. Association for the Advancement of Artificial Intelligence, Atlanta, GA, USA, 1713–1717.

[17] Shih Yun Lo, Shiqi Zhang, and Peter Stone. 2018. PETLON: Planning efficiently for task-level-optimal navigation. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, Vol. 1. International Foundation for Autonomous Agents and Multiagent Systems, Stockholm, Sweden, 220–228.

[18] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, Igor Mordatch, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems 30*, I Guyon, U V Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, and R Garnett (Eds.). Curran Associates, Inc., Long Beach, CA, USA, 6379–6390. https://doi.org/10.1007/BF01744832

[19] Bhaskara Marthi. 2007. Automatic Shaping and Decomposition of Reward Functions. In *Proceedings of the 24th International Conference on Machine Learning (ICML '07)*. Association for Computing Machinery, New York, NY, USA, 601–608. https://doi.org/10.1145/1273496.1273572

[20] Maja J. Matariundefined. 1997. Reinforcement Learning in the Multi-Robot Domain. In *Robot Colonies*. Vol. 4. Kluwer Academic Publishers, USA, 73–83. https://doi.org/10.1023/A:1008819414322

[21] Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. 1999. Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML '99)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 278–287.

[22] Mohammad Javad Noroozoliaee, Bechir Hamdaoui, and Kagan Tumer. 2013. Efficient objective functions for coordinated learning in large-scale distributed OSA systems. *IEEE Transactions on Mobile Computing* 12, 5 (2013), 931–944. https://doi.org/10.1109/TMC.2012.67

[23] Enrico Pagello, Antonio D'Angelo, Federico Montesello, Francesco Garelli, and Carlo Ferrari. 1999. Cooperative behaviors in multi-robot systems through implicit communication. *Robotics and Autonomous Systems* 29, 1 (1999), 65–77. https://doi.org/10.1016/S0921-8890(99)00039-1

[24] Liviu Panait and Sean Luke. 2005. Cooperative Multi-Agent Learning: The State of the Art. *Autonomous Agents and Multi-Agent Systems* 11, 3 (2005), 387–434. https://doi.org/10.3109/00365529108998595

[25] Lynne E. Parker. 1998. ALLIANCE: An architecture for fault tolerant multirobot cooperation. *IEEE Transactions on Robotics and Automation* 14, 2 (1998), 220–240. https://doi.org/10.1109/70.681242

[26] Lynne E. Parker. 2000. Lifelong adaptation in heterogeneous multi-robot teams: response to continual variation in individual robot performance. *Autonomous Robots* 8, 3 (2000), 239–267. https://doi.org/10.1023/A:1008977508664

[27] Mitchell A. Potter and Kenneth A. De Jong. 1994. A cooperative coevolutionary approach to function optimization. In *Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature: Parallel Problem Solving from Nature*. Springer-Verlag, Berlin, Heidelberg, 249–257.

[28] Aida Rahmattalabi, Jen Jen Chung, Mitchell Colby, and Kagan Tumer. 2016. D++: Structural credit assignment in tightly coupled multiagent domains. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Daejeon, South Korea, 4424–4429. https://doi.org/10.1109/IROS.2016.7759651

[29] Jette Randløv and Preben Alstrøm. 1998. Learning to Drive a Bicycle Using Reinforcement Learning and Shaping. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML '98)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 463–471.

[30] João Machado Santos, Tomáš Krajník, and Tom Duckett. 2017. Spatio-temporal exploration strategies for long-term autonomy of mobile robots. *Robotics and Autonomous Systems* 88 (2017), 116–126. https://doi.org/10.1016/j.robot.2016.11.016

[31] Jack F. Shepherd and Kagan Turner. 2010. Robust neuro-control for a micro quadrotor. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*. ACM, New York, NY, USA, 1131–1138. https://doi.org/10.1145/1830483.1830693

[32] Peter Stone and Manuela Veloso. 1999. Team-partitioned, opaque-transition reinforcement learning. In *In Proceedings of the Third Annual Conference on Autonomous Agents*, Vol. 1604. Springer-Verlag, Berlin, Heidelberg, 261–272. https://doi.org/10.1145/301136.301195

[33] Peter Stone and Manuela Veloso. 2000. Multiagent systems: a survey from a machine learning perspective. *Autonomous Robots* 8, 3 (2000), 345–383. https://doi.org/10.1023/A:1008942012299

[34] Matthew E. Taylor, Peter Stone, and Yaxin Liu. 2007. Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research* 8 (2007), 2125–2167.

[35] Lisa Torrey, Trevor Walker, Jude Shavlik, and Richard Maclin. 2005. Using advice to transfer knowledge acquired in one reinforcement learning task to another. In *European Conference on Machine Learning*. Springer, Berlin, Heidelberg, 412–424.

[36] Kagan Tumer and Adrian Agogino. 2009. Improving air traffic management with a learning multiagent system. *IEEE Intelligent Systems* 24, 1 (2009), 18–21. https://doi.org/10.1109/MIS.2009.10

[37] J Wang, M Lewis, and P Scerri. 2004. Cooperating Robots for Search and Rescue. In *Proceedings of the AAMAS 1st International Workshop on Agent Technology for Disaster Management*. ACM, Hakodate, Japan, 92–99. https://doi.org/10.1.1.119.8181

[38] Ying Wang, Clarence W De Silva, and Fellow Ieee. 2006. Multi-robot Box-pushing : Single-Agent Q-Learning vs. Team Q-Learning. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Beijing, China, 3694–3699.

[39] Eric Wiewiora, Garrison Cottrell, and Charles Elkan. 2003. Principled Methods for Advising Reinforcement Learning Agents. In *Proceedings, Twentieth International Conference on Machine Learning*, Vol. 2. AAAI Press, Washington, DC, USA, 792–799.